Noname manuscript No.
(will be inserted by the editor)

A certified Branch & Bound approach for Reliability-Based Optimization problems

Benjamin Martin · Marco Correia · Jorge Cruz

the date of receipt and acceptance should be inserted later

Abstract Reliability-based Optimization problems are optimization problems considering a constraint that measures reliability of the modelled system: the probability of a safety event with respect to controllable decision variables and uncertain random variables. Most solving approaches use approximate techniques for evaluating this reliability constraint. As a consequence, the reliability of the computed optimal decision is not guaranteed.

In this paper, we investigate an interval-based Branch & Bound for solving globally reliability-based optimization problems with numerical guarantee. It combines an interval Branch & Bound framework with a certified reliability analysis technique. This technique considers the reliability constraint and induced safety region modelled within Probabilistic Continuous Constraint Programming paradigm. The certified reliability analysis is numerically handled by an interval quadrature algorithm. In addition, a new interval quadrature function for two random variables, based on linear models of the safety region is described. Two implementations of the Branch & Bound, which differ on how the certified reliability analysis is handled throughout the optimization process, are presented. A numerical study of these two variants shows the relevance of the interval linear model-based quadrature function.

Keywords Reliability-based optimization, Branch & Bound, Interval analysis, Probabilistic constraints

Benjamin Martin
NOVA-LINCS, Universidade Nova de Lisboa, Lisbon, Portugal;
LIX, École Polytechnique, Paris-Saclay, France
E-mail: bmartin@lix.polytechnique.fr,

Marco Correia
NOVA-LINCS, Universidade Nova de Lisboa, Lisbon, Portugal
E-mail: mvc@fct.unl.pt,

Jorge Cruz
NOVA-LINCS, Universidade Nova de Lisboa, Lisbon, Portugal
E-mail: jcrc@fct.unl.pt.

1 Introduction

Reliability-Based Optimization (RBO) is the problem of finding the best reliable decision of a problem containing controllable decision variables and uncontrollable uncertain variables. Reliability analysis is used to assess the probability of safety of a decision considering the uncertainty. From the point of view of the optimization problem, this give rise to a particular reliability constraint. The range of applications is wide, in particular in engineering design and structural optimization, see e.g. [30, 29, 25]. Note that a RBO problem is equivalent to a chance constrained problem in Stochastic Programming.

Knowing the probability distribution of the uncertain variables, evaluating the reliability of a decision requires computing the probability of a safety event, viewed as the probability mass of a safety region. A safety region is classically represented as a conjunction of (non-linear) inequality constraints, of both the decision and random variables, each of them modelling a safety condition of a critical element of the system. The computation of the reliability requires the integration of the joint probability distribution function of the random variables over the safety region, which is a challenging numerical problem. In addition to the reliability analysis, the decision must be optimized with respect to a given objective, typically a function of the decision variables. Thus an RBO problem is two-fold: reliability assessment and optimization.

This problem has been tackled with different techniques. Most of these approaches base their reliability analysis on approximate techniques. For example *Monte Carlo* estimation, which is usually subsumed by more advanced techniques when requiring high reliability such as *First Order Reliability Method* (FORM) or *Second Order Reliability Method* (SORM), see e.g. [1, 27] for a survey of recent RBO methods. The use of approximate techniques is motivated by their relatively low computational cost and ease of use. However, when problems are highly nonlinear, they are subject to numerical errors, causing them to under or overestimate the reliability of decisions [6]. This can be a great issue when assessing the reliability is of critical importance.

In this paper, we investigate a global RBO approach based on interval analysis and the recently proposed *Probabilistic Continuous Constraint Programming* (PCCP) paradigm [5, 6]. This approach performs global optimization with rigorous reliability analysis, i.e asserting certainly the reliability of decisions. To the authors knowledge, no similar approach exists in the literature. The reliability analysis presented here can be considered as an interval version of the robust bounding approach [24] in robust optimization, whose idea is to bound the safety region with easier to integrate sets. Note also that although the standard version of the reliability analysis described here can be extended to more than two random variables, the other version with linear models is currently only dedicated to two random variables. The paper is constructed as follows. Section 2 describes the problem and a brief overview of the literature is provided. Section 3 provides the necessary background on interval analysis, and in particular on interval quadrature used with PCCP. In Section 4, we show several results for performing reliability analysis over box region of the decision space, which are embedded in the Branch & Bound algorithm for solving RBO problems described in Section 5. Some experimental results are analysed in Section 6 and the paper is concluded in Section 7.

2 Reliability-Based Optimization

In this paper, we consider RBO problems of the form:

$$\begin{bmatrix} \min\ f(y) \\ \text{s.t.}\ \ P(g(x,y) \leq 0) \geq r \\ \qquad \underline{y}_i \leq y_i \leq \overline{y}_i,\ \forall i = 1,\ldots,m \end{bmatrix} \tag{1}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ is the objective function depending on the decision variables $y \in \mathbb{R}^m$; $g(x,y) : \mathbb{R}^{n+m} \to \mathbb{R}^q$ is a conjunction of q inequality constraints considering the random variables $x \in \mathbb{R}^n$ that models the safety region, $r \in [0,1]$ is the required reliability and $\underline{y}_i, \overline{y}_i$ represents respectively lower and upper bounds on the decision variable y_i. In this paper, we consider f and g to be continuously differentiable.

We will denote by $\mathcal{H}(y) := \{x \in \mathbb{R}^n : g(x,y) \leq 0\}$ the safety region corresponding to a decision y and $\boldsymbol{y}^{\text{init}} = \{y \in \mathbb{R}^m : \underline{y} \leq y \leq \overline{y}\}$ the bound on the decision variables[1]. We consider the vector of random variables $x = (x_1,\ldots,x_n)$ with joint probability distribution function (PDF) $\Phi : \mathbb{R}^n \to \mathbb{R}_+$ (which we will consider continuously differentiable as well) defined on a probability space $(\Omega, \mathcal{B}, P)$ where $\mathcal{H}(y) \subseteq \Omega, \forall y \in \boldsymbol{y}^{\text{init}}$, $\mathcal{B}$ being a Borel σ-algebra and P the probability measure. Therefore the reliability of a decision y is given by

$$P(x \in \mathcal{H}(y)) := \int_{x \in \mathcal{H}(y)} \Phi(x)\,\mathrm{d}x. \tag{2}$$

Evaluating (2) is numerically challenging as the set $\mathcal{H}(y)$ is defined by a set of nonlinear inequalities defining the safety region. As simulations techniques such as Monte Carlo are in general too computationally expensive, in particular when r is high, most approaches in the RBO literature use the method FORM or SORM to evaluate (2). These two methods are based on a transformation of the random space into a standard centered normally distributed random space, and the computation of the *Most Probable Failure Point* and induced reliability index. This latter part is typically done via solving an optimization sub-problem. Then a first or second order approximation of the safety region in the new space is built and used to evaluate the probability given the standard (multivariate) centered normally distributed law. Other approximate methods that uses the reliability index are the Dimension Reduction Method (DRM) [25], in which a Taylor series expansion is used to better estimate the safety region.

Most RBO solving methods are based on one of these evaluations of reliability but differ on how the reliability index is computed, or how a targeted reliability index is attained. In order to perform optimization and reliability analysis, these methods consider different transformations of the problem (1). Three main categories of methods can be distinguished [1,27]: *two-level* approaches, in which the main optimization loop contains a sub-process performing the reliability analysis; *single-loop* approaches, in which the reliability analysis is replaced by equivalent complementarity constraints leading to a single optimization loop; and *decoupled* approaches, in which the RBO problem is transformed into a sequence of simpler deterministic problems in which the reliability analysis is performed throughout the sequence. For instance, see [9,17,7] for an example of each approach. However,

[1] Vector comparison must be understood component-wise.

as the reliability analysis of the optimal decision returned by these methods, including via Monte Carlo estimation, is not done in a rigorous way, its reliability is not strictly guaranteed, see e.g [6].

In [10], the convergence of a verified quadrature method, based on interval analysis, is proposed. This method computes a verified enclosure of an integral as given by (2). This method has been used in [5,6] within the probabilistic continuous constraint paradigm for producing enclosures of the probability of an event modelled as a constraint system, as $\mathcal{H}(y)$ [6]. However, this paradigm has not been used in an optimization context for which, as can be seen in the literature, the way the reliability analysis is performed impacts the design of the optimization method. We introduce in the following section the probabilistic continuous constraint paradigm and necessary background on interval analysis.

3 Preliminaries on Interval Analysis

Interval analysis (IA) is a branch of numerical analysis born in the 1960's [21]. It considers using interval of reals instead of reals, and gives a framework for handling uncertainties and verified computations. It has been sucessfully applied in numerical constraint satisfaction problems, and in particular in the recent development on probabilistic constraints [5,6], verified quadrature [10] and in nonlinear global optimization [13,16,23]. We refer to [22,14,16,15] for a broad overview of IA.

3.1 Notations and definitions

An interval $\boldsymbol{x}$ is a closed connected subset of $\mathbb{R}$ and is defined by a lower and an upper bound $\underline{x}, \overline{x} \in \mathbb{R}$, i.e. $\boldsymbol{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} : \underline{x} \leq x \leq \overline{x}\}$. We denote by $\mathbb{IR}$ the set of all bounded real intervals. A n-dimensional box $\boldsymbol{x}$ is a vector of n intervals $(\boldsymbol{x}_i)_{1 \leq i \leq n}$, defined by a lower and an upper bound vector $\underline{x}$ and $\overline{x}$. The hull operation of any arbitrary subset $\mathcal{U} \subset \mathbb{R}^n$ is $\square\mathcal{U} = [\underline{u}, \overline{u}]$ such that $\forall i \in \{1, \ldots, n\}$, $\underline{u}_i = \inf\{u_i : u \in \mathcal{U}\}$ and $\overline{u}_i = \sup\{u_i : u \in \mathcal{U}\}$. Given an interval $\boldsymbol{x}$, $\text{mid}(\boldsymbol{x}) := 0.5(\underline{x} + \overline{x})$ is its *center*, $\text{wid}(\boldsymbol{x}) := \overline{x} - \underline{x}$ is its *width*. The width of a box is the maximum of its component-wise widths. The volume of a n-dimensional box $\boldsymbol{x}$ is $\text{vol}(\boldsymbol{x}) := \prod_{1 \leq i \leq n} \text{wid}(\boldsymbol{x}_i)$. A convergent sequence of boxes $(\boldsymbol{x}^k)_{k \in \mathbb{N}}$ is a sequence satisfying $\lim_{k \to \infty} \text{wid}(\boldsymbol{x}^k) = 0$ and $\exists x \in \boldsymbol{x}^k, \forall k \in \mathbb{N}$.

An *interval extension* $\boldsymbol{f} : \mathbb{IR}^n \to \mathbb{IR}$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is an interval function satisfying the containment principle, i.e. $f(\boldsymbol{x}) := \{f(x) : x \in \boldsymbol{x}\} \subseteq \boldsymbol{f}(\boldsymbol{x})$ for any box $\boldsymbol{x} \in \mathbb{IR}^n$. A similar component-wise definition holds for vector-valued functions. An interval extension is *convergent* if for any sequence of boxes $(\boldsymbol{x}^k)_{k \in \mathbb{N}} \subseteq \mathcal{U}$, $\lim_{k \to \infty} \text{wid}(\boldsymbol{x}^k) = 0 \implies \lim_{k \to \infty} \text{wid}(\boldsymbol{f}(\boldsymbol{x}^k)) = 0$. The *natural extension* $\boldsymbol{f}$ of f, which will be considered the default interval extension in the paper, consist of replacing all arithmetic and unary operations in the function f by their interval arithmetic counterparts. A Taylor model of $f : \mathbb{R}^n \to \mathbb{R}$ inside a box $\boldsymbol{x} \in \mathbb{IR}^n$ is a pair $\langle p, \boldsymbol{R} \rangle$ where p is a polynomial and $\boldsymbol{R}$ an interval such that $f(x) \in p(x) + \boldsymbol{R}$ for all $x \in \boldsymbol{x}$.

Integration over a box domain can be performed with an interval extension:

$$\int_{\boldsymbol{x}} f(x)\,\mathrm{d}x \in \boldsymbol{f}(\boldsymbol{x})\mathrm{vol}(\boldsymbol{x}). \tag{3}$$

Considering a Taylor model extension, this can be simplified to

$$\int_{\boldsymbol{x}} f(x)\,\mathrm{d}x \in \boldsymbol{T}_f(\boldsymbol{x}) = \int_{\boldsymbol{x}} p(x)\,\mathrm{d}x + \int_{\boldsymbol{x}} \boldsymbol{R}\,\mathrm{d}x, \tag{4}$$

where $\int_{\boldsymbol{x}} \boldsymbol{R}\,\mathrm{d}x = \boldsymbol{R}\mathrm{vol}(\boldsymbol{x})$. The Taylor model approach tend to be more accurate for integration [10] as a closed form formula can be derived.

3.2 Probabilistic Constraints and Interval-based quadrature

A classical application of interval analysis is finding and characterizing solutions of a *Continuous Constraint Satisfaction Problem* (CCSP). Considering a set of n variables $x = (x_1, \ldots, x_n)$ whose values lie within an initial box domain $\boldsymbol{x}^{\text{init}} \in \mathbb{IR}^n$, a CCSP is the problem of finding all assignments of variables values within $\boldsymbol{x}^{\text{init}}$ satisfying a set of constraints, usually represented as nonlinear equalities and inequalities. Here, we solely focus on CCSP whose solution set $\mathcal{H}$ is defined with respect to q inequality constraints, i.e. $\mathcal{H} = \{x : g(x) \leq 0, x \in \boldsymbol{x}^{\text{init}}\}$ with $g : \mathbb{R}^n \to \mathbb{R}^q$.

A solution to a CCSP is usually built as a *paving* of $\mathcal{H}$, i.e. a set of possibly edge adjacent boxes $\mathcal{H}_\square$ covering $\mathcal{H}$, obtained through a *Branch & Prune* (B&P) algorithm. This method iteratively subdivides (branching step) the initial domain $\boldsymbol{x}^{\text{init}}$ (considered as the initial paving of $\mathcal{H}$) and eliminates/reduces (pruning step) sub-boxes containing assignments not satisfying the constraints. This latter step is typically done through constraint propagation and consistency techniques [2,3, 18]. A box $\boldsymbol{x}$ from the paving can be classified into different status. Considering the interval $[\underline{g}, \overline{g}] = \boldsymbol{g}(\boldsymbol{x})$, $\boldsymbol{x}$ is: *inner* (within $\mathcal{H}$) if $\overline{g} \leq 0$, *outer* (outside $\mathcal{H}$) if $\exists i = \{1, \ldots, q\}$ s.t. $\underline{g_i} > 0$ ($\boldsymbol{x}$ is usually eliminated from the paving) and *boundary* (possibly intersecting the boundary of $\mathcal{H}$) otherwise. The aim of B&P is to cover $\mathcal{H}$ accurately enough with a tight paving of $\mathcal{H}$, in particular with a sharp global enclosure of the boundary of $\mathcal{H}$ with boundary boxes.

In the *Probabilistic Continuous Constraint Programming* (PCCP) paradigm [5,6], the variables of a CCSP are considered uncertain. Hence, the CCSP is associated to a probability space, and in particular a probability measure of any events based on a joint *probability distribution function* (PDF) of the random variables x. The purpose of PCCP is then to evaluate with guarantee the probability of the event $x \in \mathcal{H}$ (such as in (2)). To do so, the method follows a B&P scheme with the objective of obtaining an enclosure of the probability as sharp as required instead of obtaining a sharp paving of $\mathcal{H}$. In order to compute an enclosure of the probability of $x \in \mathcal{H}$ given a paving $\mathcal{H}_\square$, we will use the following interval quadrature function [5,6,10]

$$\boldsymbol{I}^{\boldsymbol{g}}_{g,\Phi}(\boldsymbol{x}) := \begin{cases} \boldsymbol{T}_\Phi(\boldsymbol{x}) & \text{if } \boldsymbol{x} \textit{ inner} \\ \square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x}))\mathrm{vol}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \textit{ boundary} \\ 0 & \text{otherwise } (\textit{outer}) \end{cases}, \tag{5}$$

Algorithm 1: Branch & Prune for quadrature

Input: Constraints g; PDF Φ; initial box paving $\mathcal{H}_{\square}^{init} \subseteq \boldsymbol{x}^{\text{init}}$; Precision of the quadrature ϵ
Output: Enclosure of $P(\mathcal{H})$

$\mathcal{H}_{\square} \leftarrow \mathcal{H}_{\square}^{init}$;
while $\text{wid}(\boldsymbol{P}(\mathcal{H}_{\square})) > \epsilon$ **do**
  $\boldsymbol{x} \leftarrow \mathit{Extract}(\mathcal{H}_{\square})$;
  $\mathcal{S} \leftarrow \mathit{Split}(\boldsymbol{x})$;
  foreach $\boldsymbol{x}' \in \mathcal{S}$ **do**
    if $\boldsymbol{x}$ *is not* outer **then** $\mathcal{H}_{\square} \leftarrow \mathcal{H}_{\square} \cup \{\boldsymbol{x}'\}$;
  end
end
return $\boldsymbol{P}(\mathcal{H}_{\square})$

where $\boldsymbol{T}_{\Phi}(\boldsymbol{x})$ is defined as in (4), i.e. considering Taylor models of Φ on box $\boldsymbol{x}$. Given a paving $\mathcal{H}_{\square}$ of $\mathcal{H}$, the probability of the event $x \in \mathcal{H}$ satisfies [10]:

$$P(x \in \mathcal{H}) \in \boldsymbol{P}(\mathcal{H}_{\square}) = \sum_{\boldsymbol{x} \in \mathcal{H}_{\square}} \boldsymbol{I}_{g,\Phi}^{\boldsymbol{g}}(\boldsymbol{x}). \tag{6}$$

In order to compute $\boldsymbol{P}(\mathcal{H}_{\square})$, we use Algorithm 1, the B&P algorithm for quadrature as defined in [10] and whose convergence has been assessed. We note that here we have considered events not related to a decision as in (2), although these results are naturally extended for obtaining an enclosure of the probability of $x \in \mathcal{H}(y)$, given any decision y.

4 Reliability analysis over a decision box

Considering a decision y, the corresponding safety region in the space of random variables realisations is defined by $\mathcal{H}(y)$, which is used to define the probability of safety (2). Inside the B&B algorithm that is described in the following section, it will be necessary to be able to build an enclosure of the reliability of any decision within a decision box $\boldsymbol{y}$. Hence, if we can show that all decisions in $\boldsymbol{y}$ are not reliable, the corresponding part of the decision space can be eliminated from the search.

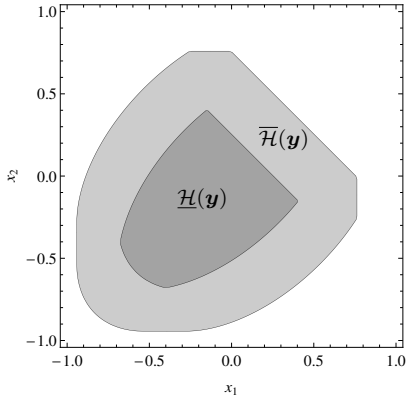
4.1 Interval quadrature with a decision box

An enclosure of the reliability of any decision within a box $\boldsymbol{y}$ can be defined as follows. Note that all the results presented here are valid since the PDF function Φ to integrate is positive.

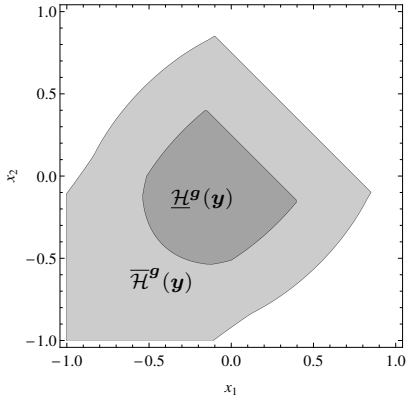
Definition 1 (Integral enclosure of the reliability of decision boxes) Consider a RBO problem with PDF Φ and a decision box $\boldsymbol{y}$. The following two sets

$$\underline{\mathcal{H}}(\boldsymbol{y}) := \left\{ x \in \boldsymbol{x}^{\text{init}} : \forall y \in \boldsymbol{y},\ g(x,y) \leq 0 \right\}, \tag{7}$$

$$\overline{\mathcal{H}}(\boldsymbol{y}) := \left\{ x \in \boldsymbol{x}^{\text{init}} : \exists y \in \boldsymbol{y},\ g(x,y) \leq 0 \right\}, \tag{8}$$



(a) Sets $\underline{\mathcal{H}}(\boldsymbol{y})$ and $\overline{\mathcal{H}}(\boldsymbol{y})$ enclosing the safety region with respect to any decision within $\boldsymbol{y}$.



(b) Under and over approximation of $\underline{\mathcal{H}}(\boldsymbol{y})$ and $\overline{\mathcal{H}}(\boldsymbol{y})$ by *inner* and *boundary* random values.

Fig. 1 2-decision and 2-random variables safety region defined by $g_1(x,y) = (x_1+y_1)^2 + (x_2+y_2)^2 - (x_1+y_1)(x_2+y_2) - 0.5 \leq 0$ and $g_2(x,y) = (x_1+y_1) + (x_2+y_2) - 0.5 \leq 0$. Considering the decision box $\boldsymbol{y} = ([-1/8, 1/8], [-1/8, 1/8])$.

verifying $\underline{\mathcal{H}}(\boldsymbol{y}) \subseteq \mathcal{H}(y) \subset \overline{\mathcal{H}}(\boldsymbol{y})$ $(\forall y \in \boldsymbol{y})$, satisfy

$$\int_{\underline{\mathcal{H}}(\boldsymbol{y})} \Phi(x)\,\mathrm{d}x \leq P(x \in \mathcal{H}(y)) \leq \int_{\overline{\mathcal{H}}(\boldsymbol{y})} \Phi(x)\,\mathrm{d}x,\ \forall y \in \boldsymbol{y}. \tag{9}$$

In other words, the probability of any decision within $\boldsymbol{y}$ for being safe is greater than the probability of safety of all possible decisions, and lower than the probability of safety of at least one decision. An example of such two sets are depicted on Figure 1(a).

With this definition, we can introduce interval quadrature functions over the random variables that produce verified enclosure similar to (5). For a given random box $\boldsymbol{x}$, those interval functions must satisfy

$$\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x}) \supseteq \left[\int_{\underline{\mathcal{H}}(\boldsymbol{y})\cap\boldsymbol{x}} \Phi(x)\,\mathrm{d}x, \int_{\overline{\mathcal{H}}(\boldsymbol{y})\cap\boldsymbol{x}} \Phi(x)\,\mathrm{d}x\right]. \tag{10}$$

We can further note that when $\boldsymbol{x} \subseteq \underline{\mathcal{H}}(\boldsymbol{y})$ then $\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x}) \supseteq \int_{\boldsymbol{x}} \Phi(x)\,\mathrm{d}x$ and when $\boldsymbol{x} \subseteq \overline{\mathcal{H}}(\boldsymbol{y}) \backslash \underline{\mathcal{H}}(\boldsymbol{y})$, then $\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x}) \supseteq [0, \int_{\boldsymbol{x}} \Phi(x)\,\mathrm{d}x]$. Also, when $\boldsymbol{y}$ is a degenerated box, i.e. a single decision y, then $\underline{\mathcal{H}}(\boldsymbol{y}) = \overline{\mathcal{H}}(\boldsymbol{y}) = \mathcal{H}(y)$.

Similar to what has been introduced in Section 3.2, we can define an interval function based on an interval extension $\boldsymbol{g}$ of the constraint functions g, that satisfy (10). To do so, the notions of *inner* and *boundary* for a random box $\boldsymbol{x}$ can be used, but here considering a decision box $\boldsymbol{y}$ and denoting $[\underline{g}, \overline{g}] = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})$. E.g. $\boldsymbol{x}$ is *inner* if $\overline{g} = \sup \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq 0$. Consequently, the following interval function based on $\boldsymbol{g}$ verifies (10)

$$\boldsymbol{I}^{\boldsymbol{y},\boldsymbol{g}}_{g,\Phi}(\boldsymbol{x}) := \begin{cases} \boldsymbol{T}_{\Phi}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \textit{ inner} \\ \square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x}))\mathrm{vol}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \textit{ boundary} \\ 0 & \text{otherwise } (\textit{outer}) \end{cases}, \tag{11}$$

which is equivalent to (5), in particular it makes use of Taylor models.

From the properties of interval extension, *inner* boxes form a subset of $\underline{\mathcal{H}}(\boldsymbol{y})$ while *boundary* boxes form a superset of $\overline{\mathcal{H}}(\boldsymbol{y})\backslash\underline{\mathcal{H}}(\boldsymbol{y})$, the union of the two building a superset of $\overline{\mathcal{H}}(\boldsymbol{y})$. This is depicted on Figure 1(b).

Definition 2 The set of *inner* and *boundary* random values, given a decision box $\boldsymbol{y}$ and interval extension $\boldsymbol{g}$ are respectively defined as

$$\underline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y}) := \left\{ x \in \boldsymbol{x}^{\text{init}} : \sup \boldsymbol{g}(x, \boldsymbol{y}) \leq 0 \right\}, \tag{12}$$

$$\mathcal{H}_0^{\boldsymbol{g}}(\boldsymbol{y}) := \left\{ x \in \boldsymbol{x}^{\text{init}} : \inf \boldsymbol{g}(x, \boldsymbol{y}) \leq 0, \exists i = \{1, \ldots, q\},\ \sup \boldsymbol{g}_i(x, \boldsymbol{y}) > 0 \right\}, \tag{13}$$

which satisfy $\underline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y}) \subseteq \underline{\mathcal{H}}(\boldsymbol{y})$ and $\overline{\mathcal{H}}(\boldsymbol{y}) \subseteq \overline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{x})$, with $\overline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y}) := \underline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y}) \cup \mathcal{H}_0^{\boldsymbol{g}}(\boldsymbol{y})$.

Therefore if we use the Algorithm 1 with the integral function (11), a paving of $\overline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y})$ and an enclosure of

$$\left[\int_{\underline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y})} \Phi(x)\, \mathrm{d}x, \int_{\overline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y})} \Phi(x)\, \mathrm{d}x \right] \supseteq \left[\int_{\underline{\mathcal{H}}(\boldsymbol{y})} \Phi(x)\, \mathrm{d}x, \int_{\overline{\mathcal{H}}(\boldsymbol{y})} \Phi(x)\, \mathrm{d}x \right] \tag{14}$$

are maintained at each iteration. Of course as a consequence and contrary to [10], the quadrature of Algorithm 1 cannot asymptotically converge to a degenerated interval hence deprecating the stopping criteria. Instead, it can at best asymptotically converge, under the hypotheses in [10], to $\left[\int_{\underline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y})} \Phi(x)\, \mathrm{d}x, \int_{\overline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y})} \Phi(x)\, \mathrm{d}x\right]$ which in general is overestimating the tighter enclosure provided by (9). Nevertheless, with $\boldsymbol{g}$ being convergent, if we consider the degenerated decision box $\boldsymbol{y} = [y, y]$, then $\underline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y}) = \overline{\mathcal{H}}^{\boldsymbol{g}}(\boldsymbol{y}) = \underline{\mathcal{H}}(\boldsymbol{y}) = \overline{\mathcal{H}}(\boldsymbol{y}) = \mathcal{H}(y)$.

In order to produce an enclosure similar to (14) with Algorithm 1, we then use the quadrature function (11) and stopping criterion based on maximum number K of iterations. From this algorithm, we can have a proof of its correctness and convergence properties that will be necessary in the following section.

Proposition 1 *The quadrature algorithm, Algorithm 1, with quadrature function* (11) *is correct, i.e. given a decision box* $\boldsymbol{y}$*, and assuming the initial paving contains* $\overline{\mathcal{H}}(\boldsymbol{y})$*, at each iteration*

$$P(x \in \mathcal{H}(y)) = \int_{\mathcal{H}(y)} \phi(x)\, \mathrm{d}x \in \sum_{\boldsymbol{x} \in \mathcal{H}_\square} \boldsymbol{I}_{g,\Phi}^{\boldsymbol{y},\boldsymbol{g}}(\boldsymbol{x}), \forall y \in \boldsymbol{y}. \tag{15}$$

Proof This is similar to the proof of [10, Theorem 1]. The initial paving is assumed correct, and only outer boxes are removed from the paving throughout the iterations, i.e. $\overline{\mathcal{H}}(y)$ is always covered by the paving. This paving is also almost disjoint, i.e. the volume of the intersection between two boxes of the paving is zero. Therefore, due to (10), for all $y \in \boldsymbol{y}$:

$$P(x \in \mathcal{H}(y)) \in \sum_{\boldsymbol{x} \in \mathcal{H}_\square} \left[\int_{\underline{\mathcal{H}}(\boldsymbol{y}) \cap \boldsymbol{x}} \Phi(x)\, \mathrm{d}x, \int_{\overline{\mathcal{H}}(\boldsymbol{y}) \cap \boldsymbol{x}} \Phi(x)\, \mathrm{d}x \right] \subseteq \sum_{\boldsymbol{x} \in \mathcal{H}_\square} \boldsymbol{I}_{g,\Phi}^{\boldsymbol{y},\boldsymbol{g}}(\boldsymbol{x}). \tag{16}$$

□

Let $\boldsymbol{P}_k(\boldsymbol{y})$ denotes the enclosure of probability maintained by the quadrature algorithm at iteration k. Consequently $\boldsymbol{P}_k(\boldsymbol{y})$ includes the probability of safety of any decisions within $\boldsymbol{y}$.

Eventually, from a sequence of convergent decision boxes, we have the following corollary which depends on definitions and results exposed in Appendix A.

Corollary 1 *Under the conditions of Theorem 3 (p. 22), Algorithm 1 is asymptotically convergent, i.e for any convergent sequence of decision boxes $(\boldsymbol{y}^k)_{k\in\mathbb{N}}$ within $\boldsymbol{y}^{\text{init}}$:*

$$\lim_{k\to\infty} \text{wid}(\boldsymbol{P}_{\mu(k)}(\boldsymbol{y}^k)) = 0, \tag{17}$$

where $\mu : \mathbb{N} \to \mathbb{N}$, $\lim_{k\to\infty} \mu(k) = \infty$.

Proof Here, we use a "Worst First Search" (with respect to the width of the quadrature function evaluation) selection strategy. The proof follows from Theorem 3, and the proof of [10, Corollary 2] with the use of [10, Lemma 2]. □

This Corollary will be useful for proving that the quadrature algorithm can be effectively used to prune non-reliable parts of the decision space.

4.2 Interval quadrature with linear models

We describe another quadrature function that differs from (11) and that can give a sharper enclosure for a *boundary* box. The current implementation considers at most $n = 2$ random variables.

The general idea is to extract from a *boundary* random box $\boldsymbol{x}$ a portion that belongs to $\underline{\mathcal{H}}(\boldsymbol{y})$, to extract a portion that contains $\overline{\mathcal{H}}(\boldsymbol{y})\backslash\underline{\mathcal{H}}(\boldsymbol{y})$, and to evaluate the integrals on both parts. Of course, due to the non-linearity of the constraint functions g, an accurate general approach based on the expression of g cannot be computationally reasonable to use. Hence, the idea is to produce inside a *boundary* random box linear models which can be used to enclose certainly the part $\overline{\mathcal{H}}(\boldsymbol{y})\backslash\underline{\mathcal{H}}(\boldsymbol{y})$ within $\boldsymbol{x}$, given a decision box $\boldsymbol{y}$ (see Definition 1). Evaluating integrals over domains defined by linear inequalities is numerically easier.

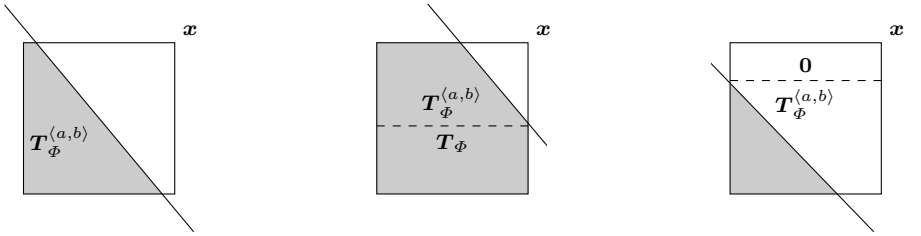
For a constraint function g_i and random/decision box $\boldsymbol{x}$ and $\boldsymbol{y}$, denote by $\langle \underline{a}_i, \underline{b}_i \rangle$ and $\langle \overline{a}_i, \overline{b}_i \rangle$, with $\underline{a}_i, \overline{a}_i \in \mathbb{R}^n$ and $\underline{b}_i, \overline{b}_i \in \mathbb{R}$, the linear models of g_i verifying

$$\underline{b}_i + \sum_{j=1}^{n} \underline{a}_{i,j} x_j \leq g_i(x, y) \leq \overline{b}_i + \sum_{j=1}^{n} \overline{a}_{i,j} x_j, \ \forall x \in \boldsymbol{x}, \ \forall y \in \boldsymbol{y}. \tag{18}$$

We will further define the set.

$$\mathcal{H}^{\langle a,b \rangle} := \left\{ x \in \boldsymbol{x}^{\text{init}} : b + \sum_{j=1}^{n} a_j x_j \leq 0 \right\}. \tag{19}$$

with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$.



(a) No decomposition.

(b) Decomposition with an inner box.

(c) Decomposition with an outer box.

Fig. 2 Decomposing 2-dimensional boxes with respect to a linear inequality. The integral quadrature function to apply for each sub-box is shown.

Then from the property (18), we can deduce the following relations:

$$\boldsymbol{x} \cap \bigcap_{i=1}^{q} \mathcal{H}^{\langle \overline{a}_i, \overline{b}_i \rangle} = \bigcap_{i=1}^{q} \left\{ x \in \boldsymbol{x} : \overline{b}_i + \sum_{j=1}^{n} \overline{a}_{i,j} x_j \leq 0 \right\} \subseteq \underline{\mathcal{H}}(\boldsymbol{y}) \cap \boldsymbol{x}, \tag{20}$$

$$\boldsymbol{x} \cap \bigcap_{i=1}^{q} \mathcal{H}^{\langle \underline{a}_i, \underline{b}_i \rangle} = \bigcap_{i=1}^{q} \left\{ x \in \boldsymbol{x} : \underline{b}_i + \sum_{i=1}^{n} \underline{a}_{i,j} x_j \leq 0 \right\} \supseteq \overline{\mathcal{H}}(\boldsymbol{y}) \cap \boldsymbol{x}. \tag{21}$$

Given $\boldsymbol{x}$ being *boundary*, with respect to a decision box $\boldsymbol{y}$, suppose that

$$\forall j = 1, \ldots, q,\ i \neq j,\ \sup \boldsymbol{g}_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \wedge 0 \in \boldsymbol{g}_i(\boldsymbol{x}, \boldsymbol{y}). \tag{22}$$

In other words, the left hand size of equations (20) and (21) only involves the linear models of g_i. This means specifically for $n = 2$ that if

$$-(b + a_2 x_2)/a_1 \in \boldsymbol{x}_1,\ \forall x_2 \in \boldsymbol{x}_2, \tag{23}$$

with $\langle a, b \rangle$ being either $\langle \underline{a}_i, \underline{b}_i \rangle$ or $\langle \overline{a}_i, \overline{b}_i \rangle$ (and up to a permutation of x_1 and x_2 and their coefficients), then an enclosure of the integral over the domain delimited by the linear model is obtained by[2]

$$\boldsymbol{T}_\Phi^{\langle a,b \rangle}(\boldsymbol{x}_1, \boldsymbol{x}_2) := \int_{\underline{x}_2}^{\overline{x}_2} \int_{\underline{x}_1}^{-(b+a_2 x_2)/a_1} p(x_1, x_2) + \boldsymbol{R} \, \mathrm{d}x_1 \, \mathrm{d}x_2, \tag{24}$$

where $\langle p, \boldsymbol{R} \rangle$ is a Taylor model of Φ. In particular, a close-form formula of (24) can be easily obtained. This case is illustrated on Figure 2(a).

When (23) is not satisfied, it is necessary, in order to compute an enclosure of the integral in $\boldsymbol{x}$ within the region delimited by the linear model, to compute a decomposition of the box $\boldsymbol{x}$ such that one sub-box satisfies (23). This is depicted in Figure 2(b) and 2(c). The other cases that may happen are either the box $\boldsymbol{x}$ is entirely out of the region induced the linear model, or entirely inner. For those cases, the integral used is respectively 0 or $\boldsymbol{T}_\Phi$.

Obtaining this decomposition requires to find the intersections, if existing, of the boundary of the linear inequality with the boundary of the box and analysing

[2] Assuming $a_i > 0$. If $a_i < 0$, the range of x_1 is from $-(b + a_2 x_2)/a_1$ to $\overline{x}_1$. If $a_i = 0$, then x_1 and x_2 and induced coefficients of the linear model can be swapped.

the signs of the coefficients a. Here, we suppose that these intersections can be computed exactly but of course in practice with the use of interval floating point arithmetic, a (small) enclosure of those points is obtained. We discuss in the end of the section how roundings have to be considered in order to maintain correctness of the integral.

For a two-dimensional random space, a random box $\boldsymbol{x}$ can always be decomposed into three almost disjoint (possibly empty) boxes verifying:

$$\boldsymbol{x}^{bound} \cup \boldsymbol{x}^{in} \cup \boldsymbol{x}^{out} = \boldsymbol{x}, \tag{25}$$

$$\forall x \in \boldsymbol{x}^{in},\ b + a^T x \leq 0, \tag{26}$$

$$\forall x \in \boldsymbol{x}^{out},\ b + a^T x > 0, \tag{27}$$

$$\exists x, x' \in \boldsymbol{x}^{bound},\ b + a^T x \leq 0 \wedge b + a^T x' > 0. \tag{28}$$

This decomposition, as illustrated on Figure 2, is used to define the following interval quadrature function over random boxes verifying (22):

$$\boldsymbol{I}_{\Phi}^{\langle a,b\rangle}(\boldsymbol{x}) := \boldsymbol{T}_{\Phi}^{\langle a,b\rangle}(\boldsymbol{x}^{bound}) + \boldsymbol{T}_{\Phi}(\boldsymbol{x}^{in}) + \mathbf{0}(\boldsymbol{x}^{out}), \tag{29}$$

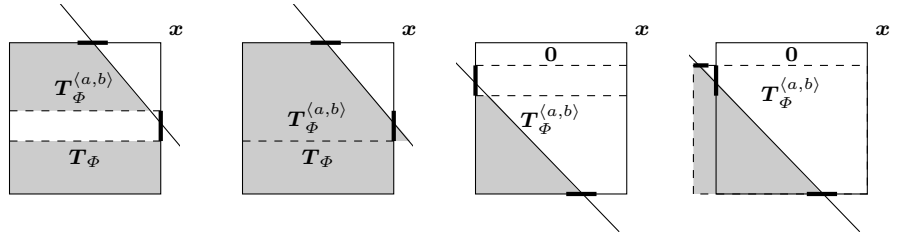
with $\mathbf{0}(.) := [0, 0]$.

We can then define the following interval quadrature function that can be used inside the quadrature algorithm described in the previous section:

$$\boldsymbol{L}_{g,\Phi}^{\boldsymbol{y},\boldsymbol{g}}(\boldsymbol{x}) := \begin{cases} \boldsymbol{T}_{\Phi}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \textit{ inner} \\ \left[\inf \boldsymbol{I}_{\Phi}^{\langle \overline{a}_i, \overline{b}_i\rangle}(\boldsymbol{x}), \sup \boldsymbol{I}_{\Phi}^{\langle \underline{a}_i, \underline{b}_i\rangle}(\boldsymbol{x})\right] & \text{if } \boldsymbol{x} \textit{ boundary} \wedge (22) \\ \square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x})\mathrm{vol}(\boldsymbol{x})) & \text{if } \boldsymbol{x} \textit{ boundary} \wedge \textit{not } (22) \\ 0 & \text{otherwise } (\textit{outer}) \end{cases}. \tag{30}$$

This function satisfies the correctness of the quadrature algorithm due to properties (20) and (21) of the linear models. It satisfies the convergence if its result is intersected with the default function (11). Note that condition (22) implies considering the linear models of a single inequality constraint. An improvement of this quadrature function can be expected by considering all the inequality constraints whose boundary are potentially crossed within $\boldsymbol{x}$, hence using one linear model per constraint leading to a polytope region. This however would require to decompose the box $\boldsymbol{x}$ with respect to the vertices of this polytope in order to obtain a closed form formula of the integral. Another important remark is that this analysis is limited to $n = 2$. We do not intend to go further since if $n > 2$, the general decomposition $\boldsymbol{x}$ in order to obtain closed form formula for the integral with respect to a linear function is more complex and would require a study on its own. Hence, we prefer to leave these two subjects to a later work on the use of linear models for general verified quadrature.

We describe now a way of obtaining the linear models satisfying (18). In [26], zero-order Taylor extensions are used to obtain a verified linear relaxation of a nonlinear optimization problem in order to compute lower and upper bounds. We use here a first-order Taylor model $\langle p_i, \boldsymbol{R}_i\rangle$ of the constraint function g_i over the random/decision domain $\boldsymbol{x}$ and $\boldsymbol{y}$ that provides two parallel linear models $\langle \underline{a}_i, \underline{b}_i\rangle$



(a) Underestimate of the case from Figure 2(b).

(b) Overestimate of the case from Figure 2(b).

(c) Underestimate of the case from Figure 2(c).

(d) Overestimate of the case from Figure 2(c).

Fig. 3 Under and overestimating the integral region for the quadrature based on linear models. The thick black lines represent the interval of uncertainty on the intersection of the boundary of the box and the boundary of the linear model region. The gray areas depict the region that is integrated.

and $\langle \overline{a}_i, \overline{b}_i \rangle$. The Taylor model is constructed as follows:

$$p_i(x) = g_i(\tilde{x}, \tilde{y}) + \nabla_x g_i(\tilde{x}, \tilde{y})^T (x - \tilde{x}), \tag{31}$$

$$\boldsymbol{R}_i = \nabla_y g_i(\tilde{x}, \tilde{y})^T (\boldsymbol{y} - \tilde{y}) \tag{32}$$

$$+ 1/2(\boldsymbol{x} - \tilde{x}, \boldsymbol{y} - \tilde{y})^T \boldsymbol{\nabla^2 g_i}(\boldsymbol{x}, \boldsymbol{y})(\boldsymbol{x} - \tilde{x}, \boldsymbol{y} - \tilde{y}), \tag{33}$$

where $\tilde{x}$ and $\tilde{y}$ are taken as the midpoint of respectively $\boldsymbol{x}$ and $\boldsymbol{y}$, and $\boldsymbol{\nabla^2 g_i}$ is an interval extension of the second-order derivatives of g_i. Consequently, the linear models are constructed as:

$$\underline{a}_i = \overline{a}_i = \nabla_x g_i(\tilde{x}, \tilde{y}), \tag{34}$$

$$\underline{b}_i = \underline{R}_i + g_i(\tilde{x}, \tilde{y}) - \nabla_x g_i(\tilde{x}, \tilde{y})^T \tilde{x}, \tag{35}$$

$$\overline{b}_i = \overline{R}_i + g_i(\tilde{x}, \tilde{y}) - \nabla_x g_i(\tilde{x}, \tilde{y})^T \tilde{x}, \tag{36}$$

which, by definition of Taylor models, satisfy (18).

Rounding. In practice, all computations are done via floating point interval arithmetic which uses outward rounding for each computation. For example, the computation of $\nabla_x g_i(\tilde{x}, \tilde{y})$ for obtaining the coefficients of $\underline{a}_i, \overline{a}_i$ is done via an interval extension, providing a small interval with floating point bounds containing the true value of $\nabla_x g_i(\tilde{x}, \tilde{y})$.

Such rounding is not critical for the correctness of a closed form formula (24): it only increases the overestimations of the integral. The issue is when computing the decomposition as the intersections of the linear model and the box boundaries are uncertain. The idea to handle these uncertainties is then to underestimate the integration region when evaluating the integral with respect to $\langle \overline{a}_i, \overline{b}_i \rangle$, and overestimate it for $\langle \underline{a}_i, \underline{b}_i \rangle$. This is illustrated on Figure 3. For example, Figure 3(a) and 3(b) depicts the decomposition of an under and overestimation of integration region based on the case on Figure 2(b). This principle can easily be generalized to any level of uncertainty, but is valid only because the function to integrate is positive.

5 Interval Branch & Bound for RBO

Interval-based Branch & Bound (B&B) methods have been widely studied in the literature for solving nonlinear continuous optimization problems rigorously, see e.g [13,16,23,28]. In order to be used efficiently, it however requires explicit knowledge of the objective function and constraints: closed-form formulas and derivatives. As has been discussed above, there is in general no explicit formulation in terms of the decision variables of the reliability constraint of (1). Hence, solving RBO problems with B&B is challenging. In particular, the resulting interval B&B would be strongly subjected to the cluster effect, a well known phenomenon in which an expensive decomposition of the search space occurs around global optimal solutions due to the difficulty of proving non-optimality or infeasibility, considered here as non-reliability.

The generic interval B&B algorithm for solving general nonlinear optimization, here formulated for RBO, is given in Algorithm 2. The set $\mathcal{S}$ can be viewed as the set of leaves of a search tree whose search nodes contain a decision box $\boldsymbol{y}$ and a corresponding local lower bound on the objective function $\underline{f}$. The global lower bound f_L and global upper bound are both initialized at line 2.1. Then the root is built with the decision box $\boldsymbol{y}$ and the lower bound f_L. The branches of the tree are built through the splitting step. Nodes from $\mathcal{S}$ whose decision box are proved to be non-optimal or non-reliable are discarded through the pruning step. The set $\mathcal{S}_{out}$ stores the nodes that have been closed and that will be returned at the end of the algorithm (it is first empty).

The main B&B loop occurs at line 2.4. The algorithm alternates splitting and pruning steps until the set $\mathcal{S}$ is empty (in which case there are no more search nodes to proceed) or until the following criteria is met:

$$(f_U - f_L) \leq \delta \max(|f_U|, 1). \tag{37}$$

It stops the algorithm once a sufficient relative precision δ on the objective function is reached. This is a classical stopping criterion to avoid an extensive and unnecessary search. At line 2.5, a node $(\boldsymbol{y}, \underline{f})$ is extracted from $\mathcal{S}$. For this, we consider extracting first the node with the lowest lower bound $\underline{f}$ (best first search). Then, the node is split into several sub-nodes at line 2.6 (effectively expanding the search tree) which are stored in the temporary list $\mathcal{S}'$. We consider here, as for the quadrature algorithm presented in the previous section, a bisection of $\boldsymbol{y}$ with respect to the component $\boldsymbol{y}_i$ of largest width. Afterwards, for each sub-node $(\boldsymbol{y}', \underline{f}')$ in $\mathcal{S}'$, pruning techniques are applied in order to discard non-optimal and non-reliable decisions from $\boldsymbol{y}'$. This step is described in details hereafter in Section 5.2, from the result of Section 4. Then, if the box $\boldsymbol{y}'$ has not been entirely pruned, the bounds f_U and $\underline{f}'$ are updated consequently at line 2.10. This step is described in details in Section 5.1 and 5.2. The node $(\boldsymbol{y}', \underline{f}')$ is then checked if it can be closed at line 2.11, i.e. if it has to be stored in $\mathcal{S}_{out}$ and not be considered again by the algorithm. A classical criterion that we use here is based on a precision ϵ_y on the decisions, i.e. a node with decision box $\boldsymbol{y}$ is closed if $\text{wid}(\boldsymbol{y}) \leq \epsilon_y$. Another criterion we use is based on (37): if it is satisfied, then all nodes in $\mathcal{S}$ are considered closed and stored in $\mathcal{S}_{out}$ (see line 2.20). Otherwise, the node is inserted into $\mathcal{S}$. At the end of each iteration, the global lower bound f_L is updated to the minimal local lower bound of all nodes in $\mathcal{S} \cup \mathcal{S}_{out}$.

Algorithm 2: Interval Branch & Bound for RBO

Input: RBO problem $\mathcal{P} = (f, g, \Phi, r)$ with objective function f; constraint functions g; PDF Φ; reliability level r; initial random and decision box $\boldsymbol{x}^{\text{init}}$ and $\boldsymbol{y}^{\text{init}}$; objective precision δ

Output: nodes $\mathcal{S}_{out}$ containing reliably optimal solutions; upper bound value f_U

2.1 $f_L, f_U \leftarrow \mathit{InitializeBounds}(\boldsymbol{y}^{\text{init}}, \boldsymbol{x}^{\text{init}}, \mathcal{P})$;
2.2 $\mathcal{S} \leftarrow \{(\boldsymbol{y}^{\text{init}}, f_L)\}$;
2.3 $\mathcal{S}_{out} \leftarrow \emptyset$;
2.4 **while** $\mathcal{S} \neq \emptyset$ ***or not*** (37) **do**
2.5 $\quad (\boldsymbol{y}, \underline{f}) \leftarrow \mathit{Extract}(\mathcal{S})$;
2.6 $\quad \mathcal{S}' \leftarrow \mathit{Split}(\boldsymbol{y}, \underline{f}, \mathcal{P})$;
2.7 $\quad$ **foreach** $(\boldsymbol{y}', \underline{f}') \in \mathcal{S}'$ **do**
2.8 $\quad\quad \boldsymbol{y}' \leftarrow \mathit{Prune}(\boldsymbol{y}', \boldsymbol{x}^{\text{init}}, \underline{f}', f_U, \mathcal{P})$;
2.9 $\quad\quad$ **if** $\boldsymbol{y}' \neq \emptyset$ **then**
2.10 $\quad\quad\quad f_U, \underline{f}' \leftarrow \mathit{UpdateBounds}(\boldsymbol{y}', \boldsymbol{x}^{\text{init}}, \mathcal{P})$;
2.11 $\quad\quad\quad$ **if** $\mathit{IsClosed}(\boldsymbol{y}', \underline{f}', f_U)$ **then**
2.12 $\quad\quad\quad\quad \mathcal{S}_{out} \leftarrow \mathcal{S}_{out} \cup \{(\boldsymbol{y}', \underline{f}')\}$;
2.13 $\quad\quad\quad$ **else**
2.14 $\quad\quad\quad\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{(\boldsymbol{y}', \underline{f}')\}$;
2.15 $\quad\quad\quad$ **end**
2.16 $\quad\quad$ **end**
2.17 $\quad$ **end**
2.18 $\quad f_L \leftarrow \min_{(\boldsymbol{y}, \underline{f}) \in \mathcal{S} \cup \mathcal{S}_{out}} \underline{f}$;
2.19 **end**
2.20 **if** (37) **then** $\mathcal{S}_{out} \leftarrow \mathcal{S}_{out} \cup \mathcal{S}$;
2.21 **return** $\mathcal{S}_{out}, f_U$

5.1 Convergence results of B&B for RBO

In this section, we discuss the correctness and convergence of Algorithm 2 for solving RBO problems. For this purpose, we describe a basic B&B algorithm upon which more practical implementations are built and presented in Section 5.2. We use the quadrature algorithm from Algorithm 1 as detailed in Section 4 for performing reliability analysis over decision boxes. We recall that it uses either quadrature function (11) or (30) and a stopping criterion based on maximum number K of iterations.

We will note $\mathcal{S}_k$ the set $\mathcal{S}$ from Algorithm 2 at the kth iteration. In this section, the procedures *Extract*, *Prune* and *UpdateBounds* are implemented as follows.

Extract : Extract first the node whose lower bound $\underline{f}$ is the lowest.

Prune : Discard the node if $f_U < \underline{f}$. Apply the quadrature Algorithm 1 implemented as previously described with respect to the current decision box $\boldsymbol{y}$. Discard the node if $\sup \boldsymbol{P}(\boldsymbol{y}) < r$.

UpdateBounds : Given $\boldsymbol{f}$ an interval extension of f, $\underline{f} = \inf \boldsymbol{f}(\boldsymbol{y})$. Noting $\tilde{y} = \text{mid}(\boldsymbol{y})$, then $f_U = f(\tilde{y})$ if $f(\tilde{y}) < f_U$ and $\tilde{y}$ is proved reliable by the quadrature algorithm.

For the method *Split*, we will need a fair splitting strategy (as in [10]). Consider a decision box $\boldsymbol{y}^k$ from a node from $\mathcal{S}_k$ at the kth iteration of the B&B algorithm. Denote by $\text{height}(\boldsymbol{y}^k)$ the number of times the induced node has been split during the algorithm (its height in the search tree). Then we say that the splitting is *fair*

if:

$$\lim_{k\to\infty} \text{height}(\boldsymbol{y}^k) = \infty \Longrightarrow \lim_{k\to\infty} \text{wid}(\boldsymbol{y}^k) = 0 \tag{38}$$

For example, the strategy of bisecting the component $\boldsymbol{y}_i$ of largest width is fair.

The first theoretical result is the correctness of Algorithm 2.

Theorem 1 (Correctness) *The B&B Algorithm 2 is correct, i.e. each globally optimal reliable solution of the RBO problem belongs to a decision box of a node in $\mathcal{S}_k$ at each iteration k.*

Proof Since the pruning step does not discard nodes whose decision box contains globally optimal solutions and reliable solutions (due to the correctness of the quadrature), then there is always nodes in $\mathcal{S}_k$ for all k containing each globally optimal reliable decision. □

We ensure this way that globally optimal reliable decisions are never discarded from the search, validating the global scope of Algorithm 2.

Now we prove the asymptotic convergence of the B&B algorithm for RBO problems: when considering $\delta = 0$ and $\epsilon_y = 0$, we show that any sequence of nodes that is infinitely expanded by selection and splitting converges to a reliable optimal solution. To do so, we first prove the corresponding sequence of decision boxes cannot contain non-reliable decisions as they are eventually pruned by *Pruning*.

Proposition 2 *Let $(\boldsymbol{y}^k)_{k\in\mathbb{N}}$ be a sequence of decision boxes such that $\lim_{k\to\infty} \text{height}(\boldsymbol{y}^k) = \infty$ verifying (38), and $\boldsymbol{y}^k \subseteq \boldsymbol{y}^{\text{init}}$. Define by $K = \ell_k = \text{height}(\boldsymbol{y}^k)$ the number of iterations of the quadrature algorithm. Under the conditions of Corollary 1, if there exists a decision $\hat{y} \in \boldsymbol{y}^k$, $\forall k \in \mathbb{N}$ with $P(x \in \mathcal{H}(\hat{y})) < r$, then there exists a $\overline{k}$ such that the enclosure $\boldsymbol{P}(\boldsymbol{y}^{\overline{k}})$ produced by the quadrature algorithm verifies $\sup \boldsymbol{P}(\boldsymbol{y}^{\overline{k}}) < r$.*

Proof For a given $\boldsymbol{y}^k$, the result of the quadrature algorithm is $\boldsymbol{P}_{\ell_k}(\boldsymbol{y}^k)$ where $\ell_k = \text{height}(\boldsymbol{y}^k)$ is globally increasing with k by assumption. Moreover, $(\boldsymbol{y}^k)_{k\in\mathbb{N}}$ is convergent due to (38) and $\hat{y} \in \boldsymbol{y}^k, \forall k \in \mathbb{N}$. We can then apply Corollary 1, which states that $\lim_{k\to\infty} \text{wid}(\boldsymbol{P}_{\ell_k}(\boldsymbol{y}^k)) = 0$.

From Proposition 1, $P(x \in \mathcal{H}(\hat{y})) \in \boldsymbol{P}_{\ell_k}(\boldsymbol{y}^k)$, since $\hat{y} \in \boldsymbol{y}^k$ for all k. As a consequence, $\sup \boldsymbol{P}_{\ell_k}(\boldsymbol{y}^k) - P(x \in \mathcal{H}(\hat{y})) \leq \text{wid}(\boldsymbol{P}_{\ell_k}(\boldsymbol{y}^k))$. As it converges to zero, for each $\epsilon > 0$, there is a $\overline{k}$ such that $\text{wid}(\boldsymbol{P}_{\ell_{\overline{k}}}(\boldsymbol{y}^{\overline{k}})) < \epsilon$. Taking $\epsilon = r - P(x \in \mathcal{H}(\hat{y}))$ entails that $\sup \boldsymbol{P}_{\ell_{\overline{k}}}(\boldsymbol{y}^{\overline{k}}) < r$, completing the proof. □

This proposition ensures we are able to decide, with sufficient decomposition of decision domains and sufficient number of quadrature iterations, the non-reliable parts of the decision space. The next theorem validates the asymptotic convergence of Algorithm 2.

Theorem 2 (Asymptotic convergence) *Define by $K = \ell_k = \text{height}(\boldsymbol{y}^k)$ the maximal number of iterations of the quadrature algorithm. Given that the RBO problem to solve is bounded and contains an optimal reliable solution, that the splitting method is fair, and that $\boldsymbol{f}$ is convergent in $\boldsymbol{y}^{\text{init}}$, then under the conditions of Corollary 1, the B&B algorithm in Algorithm 2 verifies that for any infinite sequence of nodes $((\boldsymbol{y}^k, \underline{f}^k))_{k\in\mathbb{N}} \in \mathcal{S}_k$ with $\boldsymbol{y}^{k+1} \subseteq \boldsymbol{y}^k$ and $\lim_{k\to\infty} \text{height}(y^k) = \infty$ (corresponding to an infinitely expanded branch of the search tree), there exists a $\hat{y} \in \boldsymbol{y}^k$ for all $k \in \mathbb{N}$ such that $\hat{y}$ is a globally reliable optimal solution and $\lim_{k\to\infty} \text{wid}(\boldsymbol{y}^k) = 0$.*

Proof Recall that we have $\delta = 0$ and $\epsilon_y = 0$, and that the problem is bounded, contains at least one reliable solution and Algorithm 2 is correct (see Theorem 1). Assuming that the algorithm does not finitely terminates[3], the search tree is infinitely expanded through selection and splitting entailing the existence of an infinite sequence of nodes with decreasing decision boxes (in the sense of inclusion) with infinitely increasing height. Let $((\boldsymbol{y}^k, \underline{f}^k))_{k\in\mathbb{N}} \in \mathcal{S}_k$ be such a sequence. Since the splitting is fair, then from (38), we have that $\lim_{k\to\infty} \text{wid}(\boldsymbol{y}^k) = 0$. We also have a decision $\hat{y} \in \boldsymbol{y}^k$, for all $k \in \mathbb{N}$.

By applying Proposition 2, we can show that $\hat{y}$ is reliable (otherwise, the corresponding node is eventually discarded after a finite number of iterations). Suppose that $\hat{y}$ is not globally optimal. Then there is a globally reliable optimal decision y^* such that $f(y^*) < f(\hat{y})$. Since the B&B is correct, we denote by $(\boldsymbol{y}^{*k}, \underline{f}^{*k}) \in \mathcal{S}_k$ a sequence of nodes such that $y^* \in \boldsymbol{y}^{*k}$ for all $k \in \mathbb{N}$ with $\inf \boldsymbol{f}(\boldsymbol{y}^{*k}) \leq \underline{f}^{*k} \leq f(y^*)$. Because $\boldsymbol{f}$ is convergent inside $\boldsymbol{y}^{\text{init}}$, there is an iteration $\overline{k}$ at which $\underline{f}^{*k} \leq f(y^*) < \inf \boldsymbol{f}(\boldsymbol{y}^k) \leq \underline{f}^k \leq f(\hat{y})$ for all $k \geq \overline{k}$. Therefore, starting at iteration $\overline{k}$, the node $(\boldsymbol{y}^k, \underline{f}^k)$ is not extracted from $\mathcal{S}_k$ and split as the node $(\boldsymbol{y}^{*k}, \underline{f}^{*k})$ is preferred by the extraction strategy, contradicting $\lim_{k\to\infty} \text{height}(\boldsymbol{y}^k) = \infty$. Thus, $\hat{y}$ is a globally optimal reliable solution to the RBO problem. □

Note that if the feasible space is empty, i.e. there is no reliable solutions, then the B&B will eliminate all the search nodes due to Proposition 2.

5.2 Implementation

We investigate here two different alternatives for implementing Algorithm 2 which differ on how the rigorous reliability analysis is conducted during the search. In particular, one considers to share and propagate the paving obtained at the end of the quadrature algorithm described in Section 4 applied to a given search node to its child nodes. These implementations use the same extraction procedure and update of lower bounds as in Section 5.1. Pruning a decision box $\boldsymbol{y}$ and updating the upper bound f_U are done within the same procedure. This is justified by the fact that the update of the upper bound f_U requires to find a reliable decision within $\boldsymbol{y}$. Knowing that such a decision exists within $\boldsymbol{y}$ makes it unnecessary to prove the non-reliability of $\boldsymbol{y}$. This is used by one of the alternative algorithm to save computations. This *PruningAndUpdateUB* procedure will be used in place of the procedure *Prune* in Algorithm 2. It follows the framework presented in Algorithm 3.

It has two parameters K^{UB} and K^{box} that are used as an an upper bound on the numbers of iterations of the quadrature Algorithm 1 as described in Section 4, respectively for asserting the reliability of decision points used for updating the upper bound, and evaluate the reliability of decision box. It also takes a boolean *sharing* that determines the variant of the procedure. It takes as input a search node, the upper bound f_U and the considered RBO problem. Between lines 3.1

[3] Otherwise, all decision boxes in $\mathcal{S}_{out}$ are degenerated, reliable and one of them is a global optimum or the global upper bound and global lower bound are equal meaning a globally optimal solution has been found.

Algorithm 3: PruningAndUpdateUB

Parameters: Integer K^{UB} and K^{box} maximum number of quadrature iterations; reliability r; boolean *sharing*
Input: node $(\boldsymbol{y}, \underline{f})$; upper bound f_U set; RBO problem $\mathcal{P}$ with reliability level r
Output: narrowed box $\boldsymbol{y}$

3.1 **if** $f_U < \underline{f}$ **then return** $\emptyset$;
3.2 $\boldsymbol{y} \leftarrow \mathit{ConstraintPropagation}(\boldsymbol{y}, f(y) \leq f_U)$;
3.3 **if** $\boldsymbol{y} = \emptyset$ **then return** $\emptyset$;
3.4 $\tilde{y} \leftarrow \text{mid}(\boldsymbol{y})$;
3.5 *boundUpdated* $\leftarrow$ **false**;
3.6 **if** $f(\tilde{y}) \leq f_U$ ***and*** $\inf \mathit{Quadrature}(\tilde{y}, g, \Phi, \mathit{GetRandomPaving}(\text{parent}(\boldsymbol{y}))) \geq r$ **then**
3.7 $\quad f_U \leftarrow f(\tilde{y})$;
3.8 $\quad$ *boundUpdated* $\leftarrow$ **true**;
3.9 **end**
3.10 **if** *(sharing* ***or not*** *boundUpdated)*
3.11 ***and*** $\sup \mathit{Quadrature}(\boldsymbol{y}, g, \Phi, \mathit{GetRandomPaving}(\text{parent}(\boldsymbol{y}))) < r$ **then**
3.12 $\quad$ **return** $\emptyset$
3.13 **else**
3.14 $\quad$ **return** $\boldsymbol{y}$
3.15 **end**

and 3.3, the node is pruned with respect to its bound and the objective function. In particular, the procedure *ConstraintPropagation* applies a constraint propagation algorithm on the decision domain $\boldsymbol{y}$ with respect to the constraint $f(y) \leq f_U$. This is a common approach in interval B&B in order to discard from $\boldsymbol{y}$ decisions that imply objective values worse than f_U. We use here a propagation method based on the filtering algorithm by hull-consistency HC4 [2]. The procedure returns a narrowed, possibly empty, box.

The update of the upper bound occurs between lines 3.4 and 3.9. The midpoint $\tilde{y}$ of $\boldsymbol{y}$ is computed and it is checked whether it is reliable using the *Quadrature* procedure, referring to the quadrature algorithm. As initial paving, it uses the one returned by $\mathit{GetRandomPaving}(\text{parent}(\boldsymbol{y}))$. This procedure depends on *sharing* and is explained below. Here, $\text{parent}(\boldsymbol{y})$ designates the node for which $\boldsymbol{y}$ is the immediate child from the *Split* procedure in Algorithm 2. If $\tilde{y}$ has a better objective value than f_U and is proved to be reliable, the boolean *boundUpdated* is set to *true*.

The reliability analysis of the box $\boldsymbol{y}$ is done between lines 3.10 and 3.15. This is where the two variants of the algorithm appear, depending on the boolean *sharing*.

1- *sharing* = **false**: In that case, $\mathit{GetRandomPaving}(\text{parent}(\boldsymbol{y})) = \{\boldsymbol{x}^{\text{init}}\}$ and the *Quadrature* is solely used to test whether decision points or boxes are or are not reliable. This requires K^{UB} and K^{box} to be sufficiently high. When the midpoint of $\boldsymbol{y}$ is proved to be reliable, it is then not necessary to check the non-reliability of the decision box, saving computations.

2- *sharing* = **true**: In that case, pavings obtained at the end of the quadrature algorithm for each search node of the B&B are shared to its children[4] (for the initial node, the paving is $\{\boldsymbol{x}^{\text{init}}\}$). Hence, $\mathit{GetRandomPaving}(\text{parent}(\boldsymbol{y}))$ returns the paving $\mathcal{H}_{\square}$ obtained after the quadrature algorithm applied to the parent node of $\boldsymbol{y}$. Then, the aim of *Quadrature* is to improve the enclosure of the reliability, enabling the use of a smaller K^{box} number of iterations. The

[4] Taking $\boldsymbol{y}' \subseteq \boldsymbol{y}$, then each *inner* or *outer* random boxes with respect to $\boldsymbol{g}$ and $\boldsymbol{y}$ are also *inner* or *outer* for $\boldsymbol{y}'$. Only *boundary* boxes for $\boldsymbol{y}$ can have a different status for $\boldsymbol{y}'$.

aim of *Quadrature* for decision midpoints is the same as before (hence requiring enough K^{UB} iterations), but the shared paving can be used as an initialization.

Intuitively, the non-shared version of the Algorithm 3 leads to a *two-level* B&B as the reliability analysis is independent to the optimization process. On the other hand, the shared version can be viewed as a *decoupled approach* as the reliability analysis is improved throughout the optimization process. Note also that the shared version of the pruning implies a theoretically convergent B&B contrary to the non-shared one[5]. Note finally that in any case, *Quadrature* algorithm stops itself once reliability is proved or disproved, avoiding unnecessary iterations.

Summarizing the different procedures of Algorithm 2, we implement:

Extract : Defined as in Section 5.1.
Prune : Implemented as Algorithm 3, in either the *shared* or *non-shared* version.
UpdateBounds : Defined as in Section 5.1, except that f_U is updated with respect to Algorithm 3.

All the other components are implemented as described in the beginning of this section.

6 Experiments

We implemented the B&B algorithm for RBO problems in C++ using Gaol [11] as interval arithmetic library and Realpaver [12] for constraint propagation routines. All the experiments in the following have been run on a computer under Ubuntu 14.10 64-bit, with processor Intel i7-4702MQ 2.20GHz and 8Gb of RAM. We have implemented the *non-shared* and *shared* version of the B&B algorithm as presented in Section 5.2, using the reliability analysis based on the default quadrature function (11) or the linear model-based one (30).

For the experimental results presented here, the parameters are fixed as follows. For the stopping criterion, $\delta = 10^{-2}$ and for the closure criterion $\epsilon_y = 10^{-4}$. As a safeguard for avoiding very slow convergence, we also stopped the algorithm when $|\mathcal{S}_{out}| \geq 1000$, or when reaching a timeout of $3600s$. We set $K^{UB} = 10000$, which is overall sufficient for asserting the reliability of a single decision on the problems we have considered. We have tested the algorithm with different settings for K^{box}, depending on which version of B&B is used: $K^{box} \in \{500, 1000, 2000, 5000, 10000, 20000\}$ for the *non-shared* version and $K^{box} \in \{20, 50, 100, 200, 500\}$ for the *shared* version.

We have considered the 5 RBO problems as described in Appendix B (here referred to RBO1-5), all of them containing 2 decision variables and 2 random variables. For each of them, we have taken the two reliability values 0.9 and 0.99 for r. For the sake of clarity, we do not show here all the detailed results. These are described inside the supplementary materials[6]. We show here results comparing the best combination of *non-shared* and *shared* version of B&B with or without linear model-based quadrature. The *best* is determined as the quickest method

[5] As the total number of iterations performed by the quadrature algorithm is proportional to the height of decision boxes due to the sharing.

[6] http://ben-martin.fr/files/publications/materials/RBO/detailedRBOExperiments.pdf

Table 1 Problem RBO1: performance results

Method	t	$\text{wid}(\boldsymbol{f}_r)$	$\lvert\mathcal{S}_T\rvert$	$\max\lvert\mathcal{S}_T\rvert$	$\lvert\mathcal{H}_\square\rvert$	$\max\lvert\mathcal{H}_\square\rvert$
$r = 0.9$						
B_n (500, True)	4.98	0.0089	17	17	0	501
B_s (20, True)	1.51	0.0089	18	18	4 327	4 327
B_n (2 000, False)	10.40	0.0089	27	30	0	2 001
B_s (200, False)	5.06	0.0089	31	44	70 459	104 323
$r = 0.99$						
B_n (500, True)	4.84	0.0063	9	17	0	501
B_s (20, True)	2.12	0.0063	13	23	3 106	5 381
B_n (2 000, False)	9.85	0.0063	17	34	0	2 001
B_s (100, False)	5.26	0.0063	30	74	34 379	95 025

terminating on the standard stopping criterion, or the most accurate method if all settings are stopped due to one of the safeguard criterion. Methods in the following are represented as $< \textit{version}(K^{box}, LM) >$, where LM is a boolean indicating the use of linear model-based quadrature, and *version* being either B_n or B_s for respectively the *non-shared* and *shared* version of the algorithm. We have reported: CPU time t (in seconds); final relative precision $\text{wid}(\boldsymbol{f}_r) = (f_U - f_L)/\max(|f_U|, 1)$; number of search nodes $|\mathcal{S}_T| = |\mathcal{S}| + |\mathcal{S}_{out}|$ at the end of the algorithm, maximum $\max|\mathcal{S}_T|$ of nodes stored during the algorithm; total number of shared random boxes $|\mathcal{H}_\square|$ at the end of algorithm and maximum number of total random boxes $\max|\mathcal{H}_\square|$ stored at anytime (these latter two measures only consider random boxes produced and shared when evaluating the reliability of decision boxes).

All the results are reported in Tables 1-5. Clearly, the use of linear model-based quadrature can improve significantly the convergence speed of the B&B. In particular, it allows having a smaller K^{box} and still ensuring convergence to the required precision. For example, the best setting K^{box} using linear model-based quadrature is the smallest among the considered alternatives, which always terminate to the prescribed precision. On the contrary, the default quadrature function requires using larger K^{box} in order to be able to detect and discard non-reliable decision boxes. As can be seen for the problems RBO4 and RBO5 on Tables 4 and 5, it is necessary to use linear model-based quadrature in order to reach the prescribed precision on the objective (except for B_n on RBO5 with $K^{box} = 20000$). We can still observe that the use of larger K^{box} tends to give better accuracy, as it is then possible to discard more non-reliable decision boxes during the search.

Additionally, the *sharing* variant generally leads to a setting that converges faster than all the *non-shared* settings. It requires however much more memory when taking into account the storing of the shared random boxes. As the problems we considered here have only two decision and two random variables, we can expect this extra memory consumption to grow, potentially significantly, when dealing with larger number of decisions and random variables.

7 Conclusion

An interval B&B algorithm for solving globally RBO problems with numerical guarantees is presented in this paper. It uses a reliability analysis technique based

Table 2 Problem RBO2: performance results

Method	t	$\text{wid}(\boldsymbol{f}_r)$	$\lvert\mathcal{S}_T\rvert$	$\max\lvert\mathcal{S}_T\rvert$	$\lvert\mathcal{H}_\square\rvert$	$\max\lvert\mathcal{H}_\square\rvert$
$r = 0.9$						
B_n (500, True)	5.38	0.0076	25	25	0	501
B_s (20, True)	1.59	0.0076	30	30	8 129	8 129
B_n (2 000, False)	6.95	0.0076	37	37	0	2 001
B_s (100, False)	3.39	0.0076	48	48	66 153	75 822
$r = 0.99$						
B_n (500, True)	6.23	0.0070	7	21	0	501
B_s (20, True)	2.45	0.0070	9	31	2 007	9 904
B_n (5 000, False)	12.08	0.0070	7	21	0	5 001
B_s (500, False)	5.94	0.0070	8	30	45 206	256 892

Table 3 Problem RBO3: performance results

Method	t	$\text{wid}(\boldsymbol{f}_r)$	$\lvert\mathcal{S}_T\rvert$	$\max\lvert\mathcal{S}_T\rvert$	$\lvert\mathcal{H}_\square\rvert$	$\max\lvert\mathcal{H}_\square\rvert$
$r = 0.9$						
B_n (500, True)	13.06	0.0099	43	43	0	494
B_s (20, True)	5.11	0.0099	54	55	16 727	17 042
B_n (5 000, False)	21.16	0.0099	50	51	0	4 953
B_s (500, False)	9.14	0.0099	64	64	465 261	465 261
$r = 0.99$						
B_n (2 000, True)	29.94	0.0100	36	36	0	1 996
B_s (100, True)	10.51	0.0100	36	37	44 363	46 147
B_n (10 000, False)	60.76	0.0100	91	92	0	9 977
B_s (500, False)	25.63	0.0099	133	133	873 046	875 067

Table 4 Problem RBO4: performance results

Method	t	$\text{wid}(\boldsymbol{f}_r)$	$\lvert\mathcal{S}_T\rvert$	$\max\lvert\mathcal{S}_T\rvert$	$\lvert\mathcal{H}_\square\rvert$	$\max\lvert\mathcal{H}_\square\rvert$
$r = 0.9$						
B_n (500, True)	9.66	0.0088	6	30	0	501
B_s (50, True)	8.02	0.0088	7	31	2 839	13 666
B_n (20 000, False)	3600.58	0.0365	242	293	0	20 001
B_s (500, False)	3600.03	0.0431	729	747	8 721 110	8 778 244
$r = 0.99$						
B_n (500, True)	9.18	0.0067	8	30	0	501
B_s (20, True)	6.61	0.0067	10	39	2 490	6 588
B_n (20 000, False)	3600.80	0.0276	228	259	0	20 001
B_s (500, False)	3600.06	0.0370	303	522	3 523 295	6 249 435

Table 5 Problem RBO5: performance results

Method	t	$\text{wid}(\boldsymbol{f}_r)$	$\lvert\mathcal{S}_T\rvert$	$\max\lvert\mathcal{S}_T\rvert$	$\lvert\mathcal{H}_\square\rvert$	$\max\lvert\mathcal{H}_\square\rvert$
$r = 0.9$						
B_n (500, True)	7.91	0.0092	12	15	0	501
B_s (20, True)	3.48	0.0092	16	16	5 317	5 436
B_n (20 000, False)	1422.51	0.0092	235	235	0	20 001
B_s (500, False)	2744.28	0.0202	1 837	1 840	23 810 402	23 839 337
$r = 0.99$						
B_n (500, True)	10.03	0.0072	20	22	0	501
B_s (20, True)	5.21	0.0096	23	29	5 803	8 029
B_n (20 000, False)	3600.49	0.0108	436	489	0	20 001
B_s (500, False)	3600.09	0.0217	724	762	9 216 821	9 867 993

on the Probabilistic Continuous Constraint Programming paradigm [6,5], which is numerically based on the quadrature algorithm presented in [10]. Several results for performing reliability analysis over decision boxes are described, in particular a technique which uses linear models of the safety region. A numerical study of two variants of this B&B algorithm is presented, showing that linear model-based quadrature improves significantly the performances of the algorithm. The two variants, namely sharing or not reliability analysis results over the iterations, show different but interesting approaches: *sharing* leading to faster convergence but at an increased memory cost.

The algorithm can be improved in several directions, in particular for dealing with higher dimensional problems. First of all, new, maybe more dedicated to the PDF, quadrature inclusion functions shall be investigated in order to reduce the overestimations. This would avoid many splitting in the random variable space and the consequent heavy computational cost. An immediate idea is to adapt the quadrature with linear models to more than two random variables. The use of these models for general quadrature shall also be investigated. A complementary idea would be to adapt heuristically the number of iterations of the quadrature algorithm. For example, an improvement factor, as used for detecting slow convergence of filtering techniques in numerical constraint programming, could be adapted. Another sharing variant of B&B could be implemented so as to give a better trade-off of convergence speed and memory consumption, especially if one wants to consider higher dimensional problems. For example, we could only share *boundary* random boxes and use a global cache (for all the search nodes) for storing the integral of all the *inner* random boxes. The use of approximate reliability analysis can be interesting to guide the search and delay the call to the verified, but expensive, reliability analysis. For example, decision boxes that are approximately determined as non-reliable could be dropped from the search and brought back only if the obtained optimal result is unsatisfactory. Eventually, it would be interesting to be able to derive some guaranteed information on the derivative of the reliability constraint in terms of the decision variables. This would help to detect easier non-reliable decision boxes, for example by providing fast but verified estimation of the reliability via some centered form extension.

Acknowledgements The authors are thankful to the Portuguese Foundation for Science and Technology for having granted this work through the project PROCURE (Probabilistic Constraints for Uncertainty Reasoning in Science and Engineering Applications), ref. PTDC/EEI-CTP/1403/2012. The authors are also thankful to the anonymous referees for their useful remarks improving the quality of the paper.

A Proof of convergence of the quadrature algorithm

We state here the theorem of convergence of the quadrature algorithm over a decision box domain. This theorem is used to prove Corollary 1. To do so, we will use the results from [10], and we need to introduce some necessary notations.

Let $\mathcal{H}^k_\square(\boldsymbol{y})$ be the set of boxes $\boldsymbol{x}$ maintained by Algorithm 1 at iteration k for the quadrature with respect to a decision box $\boldsymbol{y}$. Denote $\mathcal{H}'^k_\square(\boldsymbol{y})$ the set $\{\boldsymbol{x} \in \mathcal{H}^k_\square(\boldsymbol{y}) : \mathrm{wid}(\boldsymbol{I}^{\boldsymbol{y},\boldsymbol{g}}_{g,\Phi}(\boldsymbol{x})) > 0\}$. The set $\mathcal{H}^k_\square(\boldsymbol{y})$ is decomposed into boundary boxes $\mathcal{B}_k(\boldsymbol{y})$ and inner boxes $\mathcal{L}_k(\boldsymbol{y})$, and $\mathcal{B}'_k(\boldsymbol{y})$,$\mathcal{L}'_k(\boldsymbol{y})$ denotes respectively $\mathcal{B}_k(\boldsymbol{y}) \cap \mathcal{H}'^k_\square(\boldsymbol{y})$, $\mathcal{L}_k(\boldsymbol{y}) \cap \mathcal{H}'^k_\square(\boldsymbol{y})$.

Additionally, we denote by ϵ'_k the value

$$\epsilon'_k(\boldsymbol{y}) := \max_{\boldsymbol{x} \in \mathcal{H}'^k_{\square}(\boldsymbol{y})} \text{wid}(\boldsymbol{x}).$$

Recall that Φ is positive, continuously differentiable everywhere and bounded on $\boldsymbol{x}^{\text{init}}$. This entails that the natural interval extension $\boldsymbol{\Phi}$ is convergent, and that $\boldsymbol{\Phi}(\boldsymbol{x})$ is bounded.

Using notations from [10], the excess of a quadrature inclusion function $\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x})$ is defined by

$$\text{exc}(\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x})) := \frac{\text{wid}(\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x}))}{\text{vol}(\boldsymbol{x})}. \tag{39}$$

A quadrature inclusion function $\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}$ is weakly convergent inside $\boldsymbol{x}^{\text{init}}$ if for any $\boldsymbol{x} \subseteq \boldsymbol{x}^{\text{init}}$,there exists a $c > 0$ such that $\text{exc}(\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x})) \leq c$. It is convergent if for any sequence $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ with $\boldsymbol{x}^k \subseteq \boldsymbol{x}^{\text{init}}$, it satisfies

$$\lim_{k\to\infty} \text{wid}(\boldsymbol{x}^k) = 0 \implies \lim_{k\to\infty} \text{exc}(\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x})) = 0.$$

Denote

$$\overline{r}_\epsilon := \sup\{\text{exc}(\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}(\boldsymbol{x})) : \boldsymbol{x} \subseteq \boldsymbol{x}^{\text{init}}, \text{wid}(\boldsymbol{x}) \leq \epsilon\},$$

then if $\boldsymbol{I}^{\boldsymbol{y}}_{g,\Phi}$ is convergent, then $\lim_{\epsilon\to 0} \overline{r}_\epsilon = 0$

A $\boldsymbol{g}$-convergent quadrature inclusion function satisfies that the quadrature inclusion used for boundary boxes is at least weakly convergent, and the one for inner boxes is at least convergent (the case of (11) as shown in [10]).

Theorem 3 (Convergence) *Let $(\boldsymbol{y}^k)_{k\in\mathbb{N}}$ be an infinite convergent sequence of boxes included in $\boldsymbol{y}^{\text{init}}$ with the decision $\hat{y} \in \boldsymbol{y}^k$ for all $k \in \mathbb{N}$. Given that g is continuous, that $\boldsymbol{g}$ is convergent, and that $\mathcal{H}_0(y) := \{x \in \boldsymbol{x}^{\text{init}} : g(x,y) \leq 0, \exists i\ g_i(x,y) = 0\}$ satisfies $\text{vol}(\mathcal{H}_0(\hat{y})) = 0$, then Algorithm 1 with a $\boldsymbol{g}$-convergent quadrature function like* (11) *satisfies:*

$$\lim_{k\to\infty} \epsilon'_{\mu(k)}(\boldsymbol{y}^k) = 0 \implies \lim_{k\to\infty} \text{wid}(\boldsymbol{P}_{\mu(k)}(\boldsymbol{y}^k)) = 0, \tag{40}$$

with $\mu : \mathbb{N} \to \mathbb{N}$, $\lim_{k\to\infty} \mu(k) = \infty$.

Proof Recall that,

$$\boldsymbol{P}_{\mu(k)}(\boldsymbol{y}^k) := \sum_{\boldsymbol{x} \in \mathcal{H}^{\mu(k)}_{\square}(\boldsymbol{y}^k)} \boldsymbol{I}^{\boldsymbol{y}^k,\boldsymbol{g}}_{g,\Phi}(\boldsymbol{x}) \tag{41}$$

$$= \sum_{\boldsymbol{x} \in \mathcal{B}_{\mu(k)}(\boldsymbol{y}^k)} \square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x}))\text{vol}(\boldsymbol{x}) + \sum_{\boldsymbol{x} \in \mathcal{L}_{\mu(k)}(\boldsymbol{y}^k)} \boldsymbol{T}_\Phi(\boldsymbol{x}). \tag{42}$$

Therefore,

$$\text{wid}(\boldsymbol{P}_{\mu(k)}(\boldsymbol{y})) = \sum_{\boldsymbol{x} \in \mathcal{B}_{\mu(k)}(\boldsymbol{y}^k)} \text{wid}(\square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x}))\text{vol}(\boldsymbol{x})) + \sum_{\boldsymbol{x} \in \mathcal{L}_{k'}(\boldsymbol{y}^k)} \text{wid}(\boldsymbol{T}_\Phi(\boldsymbol{x})) \tag{43}$$

$$= \sum_{\boldsymbol{x} \in \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)} \text{wid}(\square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x})))\text{vol}(\boldsymbol{x}) + \sum_{\boldsymbol{x} \in \mathcal{L}'_{\mu(k)}(\boldsymbol{y}^k)} \text{wid}(\boldsymbol{T}_\Phi(\boldsymbol{x})). \tag{44}$$

The second summation satisfies

$$\sum_{\boldsymbol{x} \in \mathcal{L}'_{\mu(k)}(\boldsymbol{y}^k)} \text{wid}(\boldsymbol{T}_\Phi(\boldsymbol{x})) \leq \overline{r}_{\epsilon'_{\mu(k)}(\boldsymbol{y}^k)} \sum_{\boldsymbol{x} \in \mathcal{L}'_{\mu(k)}(\boldsymbol{y}^k)} \text{vol}(\boldsymbol{x}) \leq \overline{r}_{\epsilon'_{\mu(k)}(\boldsymbol{y}^k)} \text{vol}(\boldsymbol{x}^{\text{init}}), \tag{45}$$

noting that $\text{wid}(\boldsymbol{T}_\Phi(\boldsymbol{x})) = \text{exc}(\boldsymbol{T}_\Phi(\boldsymbol{x}))\text{vol}(\boldsymbol{x})$, and that the excess is lower than the maximum excess $\overline{r}_{\epsilon'_{\mu(k)}(\boldsymbol{y}^k)}$. Since $\epsilon_{\mu(k)}(\boldsymbol{y}^k)$ converges to zero as $\mu(k)$ (i.e. k) tends to infinity, then

Table 6 RBO benchmark problems characteristics

	q	$\boldsymbol{y}_1^{\text{init}}$	$\boldsymbol{y}_2^{\text{init}}$	$\boldsymbol{x}_1^{\text{init}}$	$\boldsymbol{x}_2^{\text{init}}$	$x_1 \sim$	$x_2 \sim$
RBO1	3	$[1, 10]$	$[1, 10]$	$[-1, 1]$	$[-1, 1]$	$\mathcal{N}(0, 0.2)$	$\mathcal{N}(0, 0.2)$
RBO2	3	$[-400, 300]$	$[-100, 100]$	$[-50, 50]$	$[-50, 50]$	$\mathcal{N}(0, 10)$	$\mathcal{N}(0, 10)$
RBO3	1	$[0, 15]$	$[0, 15]$	$[-10, 20]$	$[-6, 12]$	$\mathcal{N}(5, 1.5)$	$\mathcal{N}(3, 0.9)$
RBO4	2	$[1, 5.5]$	$[1, 5.5]$	$[-3, 3]$	$[-3, 3]$	$\mathcal{N}(0, 0.2)$	$\mathcal{N}(0, 0.2)$
RBO5	1	$[-2, 2]$	$[-2, 2]$	$[-3, 3]$	$[-3, 3]$	$\mathcal{N}(0, 0.05)$	$\mathcal{N}(0, 0.05)$

$\overline{r}_{\epsilon_{\mu(k)}}(\boldsymbol{y}^k)$ converges to zero due to the convergence of the quadrature inclusion $\boldsymbol{T}_{\Phi}$. The limit of the sum is hence zero.

We are left with checking the limit of the first summation. Since the quadrature inclusion $\square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x}))\text{vol}(\boldsymbol{x})$ for any $\boldsymbol{x} \subseteq \boldsymbol{x}^{\text{init}}$ is weakly convergent, there exists a $c > 0$ such that $\text{exc}(\square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x}))\text{vol}(\boldsymbol{x})) \leq c$. This entails that,

$$\sum_{\boldsymbol{x} \in \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)} \text{wid}(\square([0,0] \cup \boldsymbol{\Phi}(\boldsymbol{x})))\text{vol}(\boldsymbol{x}) \leq c \sum_{\boldsymbol{x} \in \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)} \text{vol}(\boldsymbol{x}) = c\ \text{vol}(\cup \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)), \quad (46)$$

where $\cup \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)$ designates the union of boxes in $\mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)$. We study the limit of its volume.

First, we denote $\mathcal{B}' = \lim_{k \to \infty} \cup \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)$. It is defined as the set of points x such that there exist an infinite sequence of boxes $(\boldsymbol{x}^k)_{k \in \mathbb{N}}$ with $\boldsymbol{x}^k \in \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)$ and $x \in \boldsymbol{x}^k$ for all k. We need to show that the volume of $\mathcal{B}'$ is zero in order to complete the proof. To do so, we show by contradiction that $\mathcal{B}' \subseteq \mathcal{H}_0(\hat{y})$. Suppose not, then for a $x \in \mathcal{B}'$, $x \notin \mathcal{H}_0(\hat{y})$, we have by definition of $\mathcal{H}_0(\hat{y})$ either: (i) $\exists i, g_i(x, \hat{y}) > 0$; (ii) $g(x, \hat{y}) < 0$, (iii) $g(x, \hat{y}) > 0$.

Let $(\boldsymbol{x}^k)_{k \in \mathbb{N}}$ be an infinite sequence of boxes, with $x \in \boldsymbol{x}^k$ and $\boldsymbol{x}^k \in \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k)$, for all k. By definition of boundary boxes, this imposes $\inf \boldsymbol{g}(\boldsymbol{x}^k, \boldsymbol{y}^k) \leq 0$ and $\exists i, \sup \boldsymbol{g}_i(\boldsymbol{x}^k, \boldsymbol{y}^k) > 0$. Because $\epsilon'_{\mu(k)}(\boldsymbol{y}^k)$ converges to zero, so is the width of $\boldsymbol{x}^k$. The widths of the boxes $\boldsymbol{y}^k$ are also convergent to zero. Eventually, since $(x, \hat{y}) \in (\boldsymbol{x}^k, \boldsymbol{y}^k)$, $\hat{z} = g(x, \hat{y}) \in \boldsymbol{g}(\boldsymbol{x}^k, \boldsymbol{y}^k)$. Therefore, for any value $z \in \boldsymbol{g}(\boldsymbol{x}^k, \boldsymbol{y}^k)$:

$$|z - \hat{z}| \leq \text{wid}(\boldsymbol{g}(\boldsymbol{x}^k, \boldsymbol{y}^k))$$

The right hand side converges to zero due to the convergence of $\boldsymbol{g}$. As a consequence all values within $\boldsymbol{g}(\boldsymbol{x}^k, \boldsymbol{y}^k)$ converge to $\hat{z} = g(x, \hat{z})$. Hence, we can prove that if (i) $\exists i$, $g_i(x, \hat{y}) > 0$, then there exists a $\overline{k}$ such that $\inf \boldsymbol{g}_i(\boldsymbol{x}^{\overline{k}}, \boldsymbol{y}^{\overline{k}}) > 0$, contradicting $\boldsymbol{x}^k \in \mathcal{B}'_{\mu(k)}(\boldsymbol{y}^k), \forall k \in \mathbb{N}$. A similar contradiction holds for cases (ii) and (iii). Therefore, $x \in \mathcal{H}_0(\hat{y})$ which proves $\mathcal{B}' \subseteq \mathcal{H}_0(\hat{y})$ and by hypothesis that $\text{vol}(\mathcal{B}') \leq \text{vol}(\mathcal{H}_0(\hat{y})) = 0$. This completes the proof. □

B Benchmark problem descriptions

All the benchmark have $n = 2$ random variables and $m = 2$ decision variables. Random variables follow independent normal distributions. Number of constraints q modelling the safety region, initial domain of random and decision variables and random variables distributions are described in Table 6. Details on objective and constraint functions are described below. For simplicity, we denote $z_i = x_i + y_i$.

RBO1: This problem is taken and adapted from [1].

$$f(y) := y_2 \quad (47)$$

$$g_1(x, y) := -\frac{z_1^2 z_2}{20} + 1 \quad (48)$$

$$g_2(x, y) := z_1^2 + 8z_2 - 75 \quad (49)$$

$$g_3(x, y) := -\frac{(z_1 + z_2 - 5)^2}{30} - \frac{(z_1 - z_2 - 12)^2}{120} + 1 \quad (50)$$

RBO2: This problem is taken from [8].

$$f(y) := -y_2 \tag{51}$$
$$g_1(x,y) := -(x_1+y_1)^2 + 1000(x_2+y_2) \tag{52}$$
$$g_2(x,y) := (x_1+y_1) - (x_2+y_2) - 200 \tag{53}$$
$$g_3(x,y) := -(x_1+y_1) + 3(x_2+y_2) - 400 \tag{54}$$

RBO3: This problem is taken from [1].

$$f(y) := y_1^2 + y_2^2 \tag{55}$$
$$g_1(x,y) := -0.2y_1y_2(x_2)^2 + x_1 \tag{56}$$

RBO4: This problem is a nonlinear programming problem from the GLOBAL Library, taken from [19], transformed into a RBO.

$$f(y) := y_1 \tag{57}$$
$$g_1(x,y) := \frac{1}{4}z_1 + \frac{1}{2}z_2 - \frac{1}{16}z_1^2 - \frac{1}{16}z_2^2 - 1 \tag{58}$$
$$g_2(x,y) := \frac{1}{14}z_1^2 + \frac{1}{14}z_2^2 + 1 - \frac{3}{7}z_1 - \frac{3}{7}z_2 \tag{59}$$

RBO5: The definition of this problem with highly nonlinear safety region is inspired by a constraint problem from [20]. Here, $\epsilon = 0.75$.

$$f(y) := y_1 + y_2 \tag{60}$$
$$g_1(x,y) := z_1^8 - (1-\epsilon)z_1^6 + 4z_1^6z_2^2 - (3+15\epsilon)z_1^4z_2^2 + 6z_1^4z_2^4 \tag{61}$$
$$- (3-15\epsilon)z_1^2z_2^4 + 4z_1^2z_2^6 - (1+\epsilon)z_2^6 + z_2^8 \tag{62}$$

References

1. Y. Aoues and A. Chateauneuf. Benchmark study of numerical methods for reliability-based design optimization. *Structural and Multidisciplinary Optimization*, 41(2):277–294, 2010.
2. F. Benhamou, F. Goualard, L. Granvilliers, and J-F. Puget. Revising hull and box consistency. In *International Conference on Logic Programming*, pages 230–244. MIT press, 1999.
3. F. Benhamou, D. McAllister, and P. Van Hentenryck. CLP(Intervals) Revisited. In *International Symposium on Logic Programming*, pages 124–138, 1994.
4. M. Berz and K. Makino. New methods for high-dimensional verified quadrature. *Reliable Computing*, 5(1):13–22, 1999.
5. E. Carvalho. *Probabilistic Constraint Reasoning*. PhD thesis, Universidade Nova de Lisboa, 2012.
6. E. Carvalho, J. Cruz, and P. Barahona. Safe reliability assessment through probabilistic constraint reasoning. In Tomasz Nowakowski et al., editor, *Safety and Reliability: Methodology and Applications*, pages 2269 – 2277. CRC Press, 2015.
7. G. Cheng, L. Xu, and L. Jiang. A sequential approximate programming strategy for reliability-based structural optimization. *Computers & Structures*, 84(21):1353 – 1367, 2006.
8. K. Deb, S. Gupta, D. Daum, J. Branke, A.K. Mall, and D. Padmanabhan. Reliability-based optimization using evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 13(5):1054–1074, Oct 2009.
9. I. Enevoldsen and J.D. Sørensen. Reliability-based optimization in structural engineering. *Structural Safety*, 15(3):169 – 196, 1994.
10. A. Goldsztejn, J. Cruz, and E. Carvalho. Convergence analysis and adaptive strategy for the certified quadrature over a set defined by inequalities. *Journal of Computational and Applied Mathematics*, 260:543 – 560, 2014.

11. F. Goualard. *GAOL 3.1.1: Not Just Another Interval Arithmetic Library*. Laboratoire d'Informatique de Nantes-Atlantique, 4.0 edition, October 2006.
12. L. Granvilliers and F. Benhamou. Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques. *ACM Transactions Mathematical Software*, 32(1):138–156, 2006.
13. E. Hansen and G. W. Walster. *Global Optimization Using Interval Analysis - Revised And Expanded*. CRC Press, 2003.
14. L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, 2001.
15. R. B. Kearfott. Interval Computations: Introduction, Uses, and Resources. *Euromath*, Bulletin 2(1):95–112, 1996.
16. R. Baker Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, 1996.
17. N. Kuschel and R. Rackwitz. Two basic problems in reliability-based structural optimization. *Mathematical Methods of Operations Research*, 46(3):309–333, 1997.
18. O. Lhomme. Consistency Techniques for Numeric CSPs. In *International Joint Conference on Artificial Intelligence*, pages 232–238, 1993.
19. C. D. Maranas and C. A. Floudas. Global optimization in generalized geometric programming. *Computers & Chemical Engineering*, 21(4):351 – 369, 1997.
20. B. Martin, A. Goldsztejn, L. Granvilliers, and C. Jermann. Certified parallelotope continuation for one-manifolds. *SIAM Journal on Numerical Analysis*, 51(6):3373–3401, 2013.
21. R. Moore. *Interval Analysis*. Prentice-Hall, 1966.
22. A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1991.
23. A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 5 2004.
24. N. J. Oliemann. *Methods for robustness programming*. PhD thesis, Wageningen University, 2008.
25. S. Rahman and D. Wei. Design sensitivity and reliability-based structural optimization by univariate decomposition. *Structural and Multidisciplinary Optimization*, 35(3):245–261, 2008.
26. G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Inner regions and interval linearizations for global optimization. In *AAAI Conference on Artificial Intelligence*, 2011.
27. M. A. Valdebenito and G. I. Schuller. A survey on approaches for reliability-based optimization. *Structural and Multidisciplinary Optimization*, 42(5):645–663, 2010.
28. P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: A Modeling Language for Global Optimization*. MIT press, 1997.
29. B. D. Youn, K. K. Choi, R.-J. Yang, and L. Gu. Reliability-based design optimization for crashworthiness of vehicle side impact. *Structural and Multidisciplinary Optimization*, 26(3-4):272–283, 2004.
30. Y. M. Zhang, X. D. He, Q. L. Liu, and B. C. Wen. An approach of robust reliability design for mechanical components. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, 219(3):275–283, 2005.