Reliability Engineering and System Safety 236 (2023) 109314

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

S-BORM: Reliability-based optimization of general systems using buffered optimization and reliability method

Ji-Eun Byun [a,*], Welington de Oliveira [b], Johannes O. Royset [c]

[a] James Watt School of Engineering, University of Glasgow, Glasgow, United Kingdom
[b] Mines Paris, Université PSL, Centre de Mathématiques Appliquées (CMA), Sophia Antipolis, France
[c] Operations Research Department, Naval Postgraduate School, CA, United States

ARTICLE INFO

Keywords:
Reliability-based optimization
Data-driven optimization
General systems
Buffered failure probability
Difference-of-convex programming
Proximal bundle method
Superquantile

ABSTRACT

Reliability-based optimization (RBO) is crucial for identifying optimal risk-informed decisions for designing and operating engineering systems. However, its computation remains challenging as it requires a concurrent task of optimization and reliability analysis. Moreover, computation becomes even more complicated when considering performance of a general system, whose failure event is represented as a link-set of cut-sets. This is because even when component events have smooth and convex limit-state functions, the system limit-state function has neither property, except in trivial cases. To address the challenge, this study develops an efficient algorithm to solve RBO problems of general system events. We employ the buffered optimization and reliability method (BORM), which utilizes, instead of the conventional failure probability definition, the buffered failure probability. The proposed algorithm solves a sequence of difference-of-convex RBO models iteratively by employing a proximal bundle method. For demonstration, we design various numerical examples with increasing complexity that include up to 10,062 cut-sets, which are solved by the proposed algorithm within a reasonable computational time with high accuracy. We also demonstrate the algorithm's robustness by performing extensive parametric studies.

1. Introduction

To secure disaster resilience of a community, it is crucial to make optimal decisions when designing and operating engineering systems (e.g., structural systems, infrastructures, and mechanical systems) and these decisions should account appropriately for hazard risks. This can be done by performing reliability-based optimization (RBO), where a design cost is minimized while satisfying reliability constraints [1,2]. A common, reasonable way to define such reliability constraints is to constrain a failure probability under a target level. However, such a combined task of probabilistic analysis and optimization makes RBO problems theoretically and computationally challenging. Moreover, RBO problems become even more challenging when an event of interest is a system event, whose performance is determined by joint performance of multiple component events. Nonetheless, to enable accurate decision-making, it is critical to consider interdependent component events simultaneously [3,4].

To formulate and solve an RBO problem, one needs to not only represent a system failure probability as a function of design variables, but also compute (sub)gradients of such function. However, computing (sub)gradients is often challenging (or even impossible) since the conventional definition of failure probability involves discontinuous 0–1 expressions. To address this issue, various methods have been proposed to approximate the function of a failure probability. One of the most common approaches is to find the most probable point (MPP) in a failure domain and utilize it to approximate a failure probability. For example, an MPP can be used to apply the first- or second-order reliability method (FORM or SORM) which fit the function of a failure probability using polynomial functions [5]. Another successful approaches are performance measure approach (PMA) [6] and sequential optimization and reliability assessment (SORA) [7], which use MPP(s) to define reliability constraints with quantile values instead of failure probabilities. Despite their long-standing successful history, using MPPs still poses several challenges. It requires one to project original distributions into a standard normal space, which often becomes challenging for high-dimensional or non-standard distributions. Also, the implementation can become complicated if the failure domain is disconnected and there are multiple MPPs.

Various RBO methods are developed to replace failure probability by alternative metrics such as reliability index, which is referred to as reliability index approach (RIA) [8–10], or failure rate [11]. While they

* Corresponding author.
E-mail address: Ji-Eun.Byun@glasgow.ac.uk (J.-E. Byun).

https://doi.org/10.1016/j.ress.2023.109314
Received 6 October 2022; Received in revised form 11 February 2023; Accepted 15 April 2023
Available online 21 April 2023

can effectively capture essential properties of failure probability, there is still a fundamental limitation that optimization is not performed directly on failure probability and may lead to suboptimal solutions. Other successful approach is to perform directional sampling [12] or importance sampling (IS) [13] so that gradients of failure probability can be obtained as a by-product of reliability analysis. However, deciding optimal sampling directions or IS density becomes difficult as dimensionality of a distribution increases.

To address complexity of a system function, surrogate models can be employed to approximate the surface of a function, whose example models include kriging [14–16], co-kriging [17], parametric functions [2], polytope [18], or hyperplanes [19]. While they have been successfully applied to an extensive range of system types, surrogate modeling often becomes inefficient with high-dimensional probability distributions. In addition, some methods require a surrogate model for the limit-state function of each component event, which makes it arduous to handle a large number of component events. Metaheuristic optimization algorithms can be an effective solution to overcome intractability of RBO problems, which include genetic algorithms [20,21], particle swarm optimization algorithms [22], or a variation of Markov Chain Monte Carlo simulation [23]. However, as a trade-off of their general applicability, these algorithms have weaker mathematical justifications and often result in higher computational cost than gradient-based algorithms.

To overcome aforementioned limitations, a promising approach is to directly utilize realizations of random variables, i.e., samples or data points. This strategy exempts us from deriving problem-specific formulas. It enables data-driven optimization, which is particularly favored when underlying parametric distributions of given data are unknown. Moreover, if one can directly use Monte Carlo Simulation (MCS) samples, high-dimensional distributions or a large number of cut sets are no longer an issue. However, the definition of failure probability makes it challenging to handle realizations during optimization. This is because a sample is assigned either 1 or 0 depending on whether it lies in the failure domain or not. Such binary assignment lacks gradient information. This makes it difficult to employ efficient gradient-based optimization algorithms [24]. Although [25] proposed formulations to calculate such gradients, some restrictive conditions are necessary (e.g., random variables follow the multivariate normal distribution or transformation thereof). The challenge can be addressed by utilizing, instead of the conventional failure probability, the *buffered failure probability* [26–29]. This way of defining reliability permits (sub)gradient information even in a data-driven setting [28]. Motivated by this fact, [27] proposed the *buffered optimization and reliability method* (BORM), which enables efficient data-driven optimization of reliability. It is noted that the use of the buffered failure probability still closely aligns with the risk management aimed by the conventional failure probability as the two failure probabilities have strongly positive correlations [27].

General systems are often represented as a series system of multiple parallel systems or equivalently, a link-set of cut-sets. This representation is useful since it can cover any system type. However, it also highlights the challenges associated with optimization of general systems: A system event tends to have a nonsmooth and nonconvex limit-state function with respect to design variables even when component events have smooth and convex limit-state functions [26]. This effect is unavoidable because the limit-state function of a system event is represented as a *max–min* function of the limit-state functions of component events.

Motivated by the challenge mentioned above, this study proposes a novel algorithm for reliability-based optimization that is particularly specialized to handle general systems. To this end, we leverage BORM that makes handling a max–min function more manageable because it avoids the 0–1 discontinuity caused by failure probability formulations. In contrast to [27], however, which considers a "system" consisting of a single component, the proposed algorithm can handle general systems. Since the algorithm addresses *system* reliability optimization using BORM, it is named *S-BORM*. The S-BORM algorithm solves subproblems that are formulated by BORM and uses data or sample points to estimate the buffered failure probability. The subproblems fall into the class of difference-of-convex optimization problems [30] and are handled by the difference-of-convex bundle method of [31]. As we show in the following discussions, the difference-of-convex representation of subproblems can be derived by adaptively linearizing limit-state functions at a current solution candidate. We further enhance computational efficiency by employing an active-set strategy, which significantly reduces the number of samples (or data points) that need to be considered in each iteration. S-BORM has rigorous convergence properties: for limit-state functions that are linear in the decision variables, the algorithm can only converge to critical points. We design three numerical examples with increasing complexity to demonstrate that the algorithm works practically well for both linear *and* nonlinear limit-state functions. In addition, we perform extensive parametric test, which shows the robustness of the algorithm. The S-BORM algorithm is developed as a Matlab-based function applicable for customized problems, which is available at https://github.com/jieunbyun/sborm.

The paper is organized as follows. Section 2 illustrates background theories related to developing the S-BORM algorithm. The algorithm is proposed in Section 3. Performance of the algorithm is thoroughly investigated by three numerical examples in Section 4, various parametric studies in Section 5, and further extensions of examples in Section 6. Concluding remarks are presented in Section 7. More technical details can be found in Appendix.

2. Background

2.1. General system events and reliability-based optimization

A failure event of a general system is represented as a series system of parallel systems of component failure events, or equivalently, a link-set of cut-sets, i.e.,

$$E_{\text{sys}} = \bigcup_{k=1}^{K} \bigcap_{q \in \mathbb{Q}_k} E_q, \tag{1}$$

where E_{sys} and E_q ($q \in \mathbb{Q}_k$, $k = 1, \dots, K$) refer to the failure event of a system and component q, respectively. To determine whether a system or component event E is either failure or survival, a limit-state function $g(\boldsymbol{x}, \boldsymbol{V})$ can be used to account for performance of the corresponding system or component. The function depends on design variables $\boldsymbol{x} = (x_1, \dots, x_D)$ and random vector $\boldsymbol{V} = (V_1, \dots, V_M)$. A realization $\boldsymbol{v}$ of $\boldsymbol{V}$ is considered a failure if $g(\boldsymbol{x}, \boldsymbol{v}) > 0$ and a survival, otherwise.[1] Thereby, the definition of a system failure event in (1) can be represented in terms of limit-state functions as

$$g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{v}) = \max_{k=1,\dots,K} \min_{q \in \mathbb{Q}_k} g_q(\boldsymbol{x}, \boldsymbol{v}), \tag{2}$$

where $g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{v})$ and $g_q(\boldsymbol{x}, \boldsymbol{v})$ denote the limit-state function of a system and a component, respectively.

The most common formulation of reliability-based optimization is to minimize design cost $c(\boldsymbol{x})$ while satisfying a reliability constraint (i.e., system failure probability be less than a target value p_f^t):

$$\min_{\boldsymbol{x} \in \mathbb{X}} \quad c(\boldsymbol{x}) \tag{3a}$$

$$\text{subject to} \quad p(\boldsymbol{x}) \le p_f^t, \tag{3b}$$

where $\mathbb{X}$ denotes a constraint set, and the conventional failure probability is defined as

$$p(\boldsymbol{x}) = P[g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{V}) > 0]. \tag{4}$$

[1] If necessary, one can reverse the definition of failure and survival (i.e., a failure event if $g(\boldsymbol{x}, \boldsymbol{v}) < 0$) by reversing the sign of limit-state functions.

Alternatively, the reliability constraint (3b) can be represented in terms of a quantile: $p(\boldsymbol{x}) \le p_f^t$ if and only if $q_{1-p_f^t}(\boldsymbol{x}) \le 0$, where $q_\alpha(\boldsymbol{x})$ is the α-quantile of $g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{V})$. Thus, (3) is equivalent to the problem

$$\min_{\boldsymbol{x}\in\mathbb{X}} \quad c(\boldsymbol{x}) \tag{5a}$$

$$\text{subject to} \quad q_{1-p_f^t}(\boldsymbol{x}) \le 0. \tag{5b}$$

It is noted that solving (3) and (5) requires an accessible expression for the reliability constraints (3b) and (5b). However, for general systems, deriving such an analytical formula is usually impossible. In such case, the failure probability in (4) can be estimated by realizations of $\boldsymbol{V}$ (i.e., samples or data points), $\boldsymbol{v}_1, \dots, \boldsymbol{v}_N$, leading to the formula

$$\hat{p}(\boldsymbol{x}) = \sum_{n=1}^{N} p_n \cdot \mathbb{I}[g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{v}_n) > 0], \tag{6}$$

where $\hat{p}(\boldsymbol{x})$ is the estimated failure probability given $\boldsymbol{x}$, and p_n is a probability or weight of realization $\boldsymbol{v}_n$.[2] The Heaviside function $\mathbb{I}[\cdot]$ takes value 1 if the given statement is true and 0, otherwise. When replacing in (3) the probability $p(\boldsymbol{x})$ by $\hat{p}(\boldsymbol{x})$, however, the Heaviside function in (6) greatly complicates computation because it lacks (sub)gradient information. The gradient is not defined for $\boldsymbol{x}$ when there is some n leading to $g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{v}_n) = 0$ and remains 0 at all other values.

2.2. Buffered optimization and reliability method

The paper [27] recently proposed a framework for reliability-based optimization, namely *buffered optimization and reliability method* (BORM). Being traced back to [26], BORM replaces the failure probability in (3b) by the buffered failure probability. While details can be found in the references, this section presents a brief illustration that is directly related to the following discussions.

The two failure probabilities are distinguished by how the threshold value of limit-state functions is defined to determine a failure event. According to the conventional probability, this threshold is fixed at 0. In contrast, the buffered probability defines such threshold as a quantile value whose associated superquantile is 0. In more detail, consider a random variable Y and its CDF $F_Y(y)$. Then, the α-quantile of Y, q_α is defined as

$$q_\alpha = F_Y^{-1}(\alpha)$$

provided that F_Y is strictly increasing; a similar definition holds in general. The α-superquantile, denoted by $\bar{q}_\alpha$, is defined as the average value of Y beyond q_α, i.e.,[3]

$$\bar{q}_\alpha = q_\alpha + \frac{1}{1-\alpha}\mathbb{E}[\max\{Y - q_\alpha, 0\}].$$

Finally, the buffered failure probability $\bar{p}_f$ of the event $Y > 0$ is defined as

$$\bar{p}_f = 1 - \bar{\alpha}_0,$$

where $\bar{\alpha}_0$ is the probability that gives a zero superquantile, i.e., $\bar{q}_{\bar{\alpha}_0} = 0$.

These definitions motivate the shift from the RBO problem (3) to the problem

$$\min_{\boldsymbol{x}\in\mathbb{X}} \quad c(\boldsymbol{x}) \tag{7a}$$

$$\text{subject to} \quad \bar{p}(\boldsymbol{x}) \le \bar{p}_f^t, \tag{7b}$$

where $\bar{p}(\boldsymbol{x})$ is the buffered failure probability of the event $g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{V}) > 0$ and $\bar{p}_f^t$ is a threshold. We note that $\bar{p}(\boldsymbol{x}) \ge p(\boldsymbol{x})$ so that the buffered failure probability always bounds the conventional failure probability conservatively [26]. The new formulation facilitates a data-driven setting with $\boldsymbol{V}$ replaced by the outcomes $\boldsymbol{v}_1, \dots, \boldsymbol{v}_N$. In this case, (7) can be reformulated as

$$\min_{\boldsymbol{x}\in\mathbb{X},\gamma\in\mathbb{R}} \quad c(\boldsymbol{x}) \tag{8a}$$

$$\text{subject to} \quad \gamma + \frac{1}{\bar{p}_f^t}\sum_{n=1}^{N} p_n \max\{0, g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{v}_n) - \gamma\} \le 0, \tag{8b}$$

where γ is an additional real-valued design variable, which at optimality specifies the $(1 - \bar{p}_f^t)$-quantile value of $g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{V})$. It is noted that since γ appears in a well-structured manner, it does not increase the computational complexity of the optimization problem. The complexity is also not affected by the maximum operation in the constraint (8b) as it can be reformulated to retain convexity and/or smoothness of the functions within the curly brackets (detailed illustrations can be found in [26]). Accordingly, optimization complexity is governed by $\mathbb{X}$, $c(\boldsymbol{x})$, $g_{\text{sys}}(\boldsymbol{x}, \boldsymbol{v}_n)$, and sample size N. For instance, if these functions are convex and $\mathbb{X}$ is a convex set, the problem becomes convex and is thus easily solvable using standard algorithms should N be of moderate size. In such a convex setting, if instead N is too large (say $N \ge 10^4$), then the problem can be efficiently solved by nonlinearly-constrained convex bundle methods such as the one proposed in [32]. Even when the functions are neither linear nor convex, the gradients of component limit-state functions $g_q(\boldsymbol{x}, \boldsymbol{v}_n)$ can be used for optimization algorithms, which greatly facilitates implementation as we see below.

The buffered failure probability is always greater than or equal to the conventional failure probability as it defines a threshold value of failure domain more conservatively [26]. Nonetheless, the two probabilities are closely related as both represent the probability of the worst events (i.e., failure events). This is confirmed by [27] which showed that the two probabilities have strongly positive correlations, i.e., the greater one of the probabilities, the greater the other one. Accordingly, when target failure probabilities are properly scaled, the two probabilities lead to similar optimization results in most cases [27]. Still, minor differences in optimal solutions may arise as the buffered failure probability places more emphasis on tail behavior.

3. Proposed reliability-based optimization of general system events: S-BORM algorithm

3.1. Key ideas

To develop an efficient optimization scheme that can handle general systems, we introduce four ideas for solving the data-driven RBO problem (8) in settings of general systems. First, the reliability constraint (8b) is penalized and moved to the objective function. Second, we linearize the limit-state functions with respect to decision variables, adaptively at the current candidate solution. Such a linearization is necessary because although random variables are taken into account by MCS samples, limit-state functions remain (in most cases) nonlinear with respect to decision variables, precluding thus an explicit difference-of-convex decomposition of the objective function. Third, we reformulate the now modified objective as a difference-of-convex function, which produces a subproblem solvable by the difference-of-convex bundle method of [31]. Fourth, we improve computational efficiency further by employing an active-set strategy. That is, at each iteration of optimization, the algorithm considers only a subset of samples that are within or close enough to failure domains at a current solution, i.e., the samples with the highest limit-state function values. This is because each sample becomes a constraint, which leads to a significant number of constraints and, thereby, slows down optimization. Another advantage is that gradients need to be computed only for those selected constraints, which greatly reduces computational cost as computing gradients is often more expensive than computing function values. Since the number of failure events are in general very small, this

[2] For example, if $\boldsymbol{v}_1, \dots, \boldsymbol{v}_N$ are generated by Monte Carlo simulation (MCS), $p_n = 1/N$ for all n.

[3] If Y is continuously distributed, then $\bar{q}_\alpha$ is equivalent to the conditional mean $\mathbb{E}[Y|Y \ge q_\alpha]$.

strategy greatly facilitates computation. More details about active-set strategies are available in [27]. We name the proposed approach the S-BORM algorithm as it is designed to handle system events.

Our approach follows the standard path consisting of replacing a difficult problem with a sequence of simpler subproblems and/or models. However, our pathway is distinguished from what is largely considered in the optimization literature, where the subproblems typically have easily obtainable solutions via quadratic or convex optimization. We utilize more complex subproblems, which appears necessary to capture the max–min formula (2) for the system limit-state function. In turn, this requires us to adopt more advanced subroutines for solving the subproblems. Specifically, we leverage Algorithm 1 in [31]. The next subsections discuss the S-BORM algorithm in detail.

3.2. Linearization of limit-state functions for difference-of-convex decomposition

We introduce mild conditions on RBO problems: $\mathbb{X}$ is a polyhedral set and $c(\boldsymbol{x})$ as well as $g_q(\boldsymbol{x}, \boldsymbol{v}_n)$ are smooth (in $\boldsymbol{x}$) with Lipschitz continuous gradients on $\mathbb{X}$. These conditions should hold for many practical problems. It is noted that convexity is not assumed for either cost function or limit-state functions.

By recalling (2), the optimization problem (8) takes the form

$$\min_{\boldsymbol{x}\in\mathbb{X},\gamma\in\mathbb{R}} \quad c(\boldsymbol{x}) \tag{9a}$$

$$\text{subject to} \quad \gamma + \frac{1}{\bar{p}_f^t}\sum_{n=1}^{N} p_n \max\{0, \max_{k=1,\dots,K} \min_{q\in\mathbb{Q}_k} g_q(\boldsymbol{x}, \boldsymbol{v}_n) - \gamma\} \le 0. \tag{9b}$$

We penalize the reliability constraint: for $\theta \in (0, \infty)$, the optimization problem becomes

$$\begin{aligned}&\min_{\boldsymbol{x}\in\mathbb{X},\gamma\in\mathbb{R}} F(\boldsymbol{x}, \gamma; \theta), \quad \text{with} \quad F(\boldsymbol{x}, \gamma; \theta) := c(\boldsymbol{x}) \\ &+ \theta \max\left\{0, \gamma + \frac{1}{\bar{p}_f^t}\sum_{n\in\hat{\mathbb{N}}} p_n \max\left\{0, \max_{k=1,\dots,K} \min_{q\in\mathbb{Q}_k} g_q(\boldsymbol{x}, \boldsymbol{v}_n) - \gamma\right\}\right\}.\end{aligned} \tag{10}$$

Above, an active-set strategy is assumed, i.e., the problem considers only active samples $\boldsymbol{v}_n$ with $n \in \hat{\mathbb{N}} \subset \{1, \dots, N\}$. The problem is further approximated by linearizing each $g_q(\boldsymbol{x}, \boldsymbol{v}_n)$ at a candidate solution $\hat{\boldsymbol{x}}^\nu$:

$$\min_{\boldsymbol{x}\in\mathbb{X},\gamma\in\mathbb{R}} \quad F^\nu(\boldsymbol{x}, \gamma; \theta, \hat{\boldsymbol{x}}^\nu), \tag{11}$$

where

$$\begin{aligned}F^\nu(\boldsymbol{x}, \gamma; \theta, \hat{\boldsymbol{x}}^\nu) &:= c(\boldsymbol{x}) + \theta \max\left\{0, \gamma + \frac{1}{\bar{p}_f^t}\sum_{n\in\hat{\mathbb{N}}} p_n \right. \\ &\left. \times \max\left\{0, \max_{k=1,\dots,K} \min_{q\in\mathbb{Q}_k} g_q(\hat{\boldsymbol{x}}^\nu, \boldsymbol{v}_n) + \langle \nabla g_q(\hat{\boldsymbol{x}}^\nu, \boldsymbol{v}_n), \boldsymbol{x} - \hat{\boldsymbol{x}}^\nu\rangle - \gamma\right\}\right\}.\end{aligned} \tag{12}$$

The function F^ν can be written as a difference-of-convex function, i.e., a convex function minus another convex function. We derive the specific formula in an extended arXiv report [33]. Thus, (11) is a subproblem that can be addressed by Algorithm 1 in [31]. The S-BORM algorithm solves such subproblems, with slight adjustments, repeatedly as described next.

3.3. S-BORM algorithm

Based on the derivation in Section 3.2 and [33], we now present the S-BORM algorithm.

S-BORM Algorithm:

Data Given an initial point $\boldsymbol{x}^0 \in \mathbb{X}$, samples $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_N\}$, and a parameter γ^0, choose algorithm parameters $\theta > 0$, $\theta^{\max} > \theta$, $\lambda > 0$, $\omega \ge 1$, $\kappa \in (0, 1)$, and `tol`.

Step 0 Set $\nu = 0$ and $\hat{\boldsymbol{x}}^\nu = \boldsymbol{x}^0$, $\hat{\gamma}^\nu = \gamma^0$, $\theta^\nu = \theta$, $\lambda^\nu = \lambda$.

Step 1 Evaluate $g_{\text{sys}}(\hat{\boldsymbol{x}}^\nu, \boldsymbol{v}_1), \dots, g_{\text{sys}}(\hat{\boldsymbol{x}}^\nu, \boldsymbol{v}_N)$ and obtain an index set of active samples, $\hat{\mathbb{N}}^\nu \subset \{1, \dots, N\}$, with the $\lceil \omega N \bar{p}_f^t \rceil$ greatest values of $g_{\text{sys}}(\hat{\boldsymbol{x}}^\nu, \boldsymbol{v}_n)$, $n = 1, \dots, N$.

Step 2 Compute $\nabla g_q(\hat{\boldsymbol{x}}^\nu, \boldsymbol{v}_n)$ and $g_q(\hat{\boldsymbol{x}}^\nu, \boldsymbol{v}_n)$ for all $q \in \mathbb{Q}_k$, $k = 1, \dots, K$ and $n \in \hat{\mathbb{N}}^\nu$.

Step 3 Let F^ν as in (12) and solve the subproblem

$$\min_{\boldsymbol{x}\in\mathbb{X},\gamma\in\mathbb{R}} F^\nu(\boldsymbol{x}, \gamma; \theta^\nu, \hat{\boldsymbol{x}}^\nu) + \frac{\lambda^\nu}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}^\nu\|_2^2 + \frac{\lambda^\nu}{2}(\gamma - \hat{\gamma}^\nu)^2 \tag{13}$$

and obtain a point $(\boldsymbol{x}^{\nu+1}, \gamma^{\nu+1})$ using [31, Alg. 1] with initial point $(\hat{\boldsymbol{x}}^\nu, \hat{\gamma}^\nu)$. If $\left\|\boldsymbol{x}^{\nu+1} - \hat{\boldsymbol{x}}^\nu\right\|_2^2 + (\gamma^{\nu+1} - \hat{\gamma}^\nu)^2 \le \texttt{tol}$, stop and return $(\hat{\boldsymbol{x}}^\nu, \hat{\gamma}^\nu)$ as a potential solution. Else, go to **Step 4**.

Step 4 With F from (10), define the predicted decrease as

$$\zeta^\nu := F(\hat{\boldsymbol{x}}^\nu, \hat{\gamma}^\nu; \theta^\nu) - F^\nu(\boldsymbol{x}^{\nu+1}, \gamma^{\nu+1}; \theta^\nu, \hat{\boldsymbol{x}}^\nu) - \frac{\lambda^\nu}{2}\|\boldsymbol{x}^{\nu+1} - \hat{\boldsymbol{x}}^\nu\|_2^2 - \frac{\lambda^\nu}{2}(\gamma^{\nu+1} - \hat{\gamma}^\nu)^2.$$

If $F(\boldsymbol{x}^{\nu+1}, \gamma^{\nu+1}; \theta^\nu) \le F(\hat{\boldsymbol{x}}^\nu, \hat{\gamma}^\nu; \theta^\nu) - \kappa\zeta^\nu$, declare a **Serious Step:**

$$(\hat{\boldsymbol{x}}^{\nu+1}, \hat{\gamma}^{\nu+1}) := (\boldsymbol{x}^{\nu+1}, \gamma^{\nu+1}) \quad \text{and} \quad \lambda^{\nu+1} := \lambda^\nu.$$

Else, declare a **Null Step:**

$$(\hat{\boldsymbol{x}}^{\nu+1}, \hat{\gamma}^{\nu+1}) := (\hat{\boldsymbol{x}}^\nu, \hat{\gamma}^\nu) \quad \text{and} \quad \lambda^{\nu+1} := 2\lambda^\nu.$$

Step 5 Set $\theta^{\nu+1} := \min\{1.5\theta^\nu, \theta^{\max}\}$. Replace ν by $\nu + 1$ and go to **Step 1** if Serious Step or **Step 3** if Null Step.

The procedure of the algorithm is as follows. **Step 1** evaluates limit-state functions with a current candidate solution $\hat{\boldsymbol{x}}^\nu$ and sorts out active samples $\hat{\mathbb{N}}$. Since there are $N\bar{p}_f^t$ samples of system failure when a failure probability constraint is satisfied, the size of $\hat{\mathbb{N}}$ is set as $\lceil \omega N \bar{p}_f^t \rceil$ with a parameter $\omega \ge 1$. Then, in **Step 2**, the limit-state functions in the penalized problem (10) are linearized to define a subproblem. **Step 3** defines the next iterate as a solution of the subproblem.[4] If the new iterate is close enough to a current candidate solution, the algorithm is terminated. Otherwise, in **Step 4**, a descent test is carried out to ensure that the algorithm makes progress in each serious step. If the new iterates lead to satisfactory decrease in the objective function, this update is considered a Serious Step: the new iterate becomes the current candidate solution. Otherwise, it is a Null Step: the candidate solution does not change, and the parameter λ^ν is increased to force the next iterate to be closer to the previous point $\hat{x}^\nu$. At the end of each iteration, in **Step 5**, θ^ν that penalizes violating reliability constraints is increased. Thereby, solutions can be gradually nudged towards sufficient reliability.

The proposed algorithm has three major parameters: the two penalty terms λ and θ and the ratio of active samples, ω. The default values of these parameters are proposed as $\lambda = 0.01$, $\theta = 1$, $\theta^{\max} = 10^5$ and $\omega = 2$, which are also used in the following examples. Other minor parameters are κ and `tol`; κ decides the criterion whether to accept a new candidate solution, and `tol` is the stopping threshold of Euclidean distance between the candidate solution and the new iterate. Default values of both parameters are proposed as 0.01. It is noted that these parameters have little influence on optimization results as demonstrated in Section 5, where we show robustness of the proposed algorithm by performing parametric study under various settings.

4. Numerical examples

4.1. Experiment settings

We design three numerical examples to demonstrate how the S-BORM algorithm provides an efficient and accessible means to solve problems that have been considered challenging. The first two examples are structural systems. A common way to define failure of a structural system is to identify multiple failure modes. Then, a system

[4] It suffices for the solution to be a critical point of the subproblem.

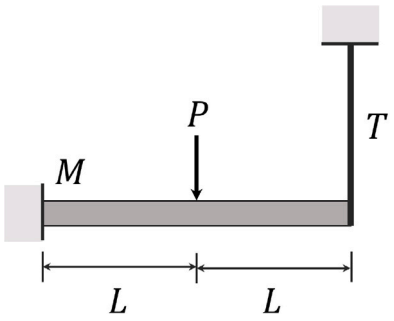

Fig. 1. Example cantilever beam-bar system.
Source: Figure recreated from [36].

failure is defined as occurrence of any of the failure modes. However, because of computational difficulty, failure modes are often handled separately by each being assigned a target failure probability [15,27]. In contrast, the S-BORM algorithm handles system failure without simplification.

The third example investigates the issue of identifying optimal allocations of testing time across components from the perspective of system reliability. This requires us to combine reliability optimization with reliability growth models (RGMs). Although the issue has been investigated from various aspects (e.g., identification of most critical components [34] or adaptive testing strategies [35]), this example is the first attempt to perform optimization within the context of a general system.

Algorithm parameters are set as the default values proposed in Section 3.3. For all problems, initial solutions are set as the midpoint of upper and lower bounds of design variables. The target buffered failure probability is $\bar{p}_f^t = 1 \cdot 10^{-3}$, and the target coefficient of variance (c.o.v.) is $\delta^t = 0.05$. This leads to the number of samples, $(1 - \bar{p}_f^t)/(\bar{p}_f^t \cdot (\delta^t)^2) = 399{,}600$ [27]. Then, with the default parameter $\omega = 2$, the number of active samples becomes $\lceil 2 \cdot 399{,}600 \cdot 10^{-3} \rceil = 800$. For computation, a personal desktop is used with processor 11th Gen Intel® Core™ i7 and RAM 16.0 GB.

4.2. Design of cantilever beam-bar system

This example investigates an optimal design of a cantilever beam-bar system illustrated in Fig. 1. The system consists of an ideally plastic cantilever beam of moment capacity M and length $2L = 2 \cdot 5$, which is propped by an ideally rigid-brittle bar of strength T [36]. The structure is subjected to load P that is applied on the middle of the bar. Load P is a random variable following the normal distribution with mean $\mu_P = 150$ and standard deviation $\sigma_P = 30$. Moment M and strength T are also normal random variables with mean μ_M and μ_T and standard deviation $\sigma_M = 300$ and $\sigma_T = 20$, respectively. In this example, we optimize two design variables $x_1 = \mu_M \in [500, 1500]$ and $x_2 = \mu_T \in [50, 150]$. The cost function is $c(x) = 2x_1 + x_2$.

As seen from [36], we identify six limit-state functions such that

$$\begin{aligned} g_1(\boldsymbol{x}, \boldsymbol{v}) &= -(x_2 + v_2 - 5v_3/16), \\ g_2(\boldsymbol{x}, \boldsymbol{v}) &= -(x_1 + v_1 - Lv_3), \\ g_3(\boldsymbol{x}, \boldsymbol{v}) &= -(x_1 + v_1 - 3Lv_3/8), \\ g_4(\boldsymbol{x}, \boldsymbol{v}) &= -(x_1 + v_1 - Lv_3/3), \\ g_5(\boldsymbol{x}, \boldsymbol{v}) &= -(x_1 + v_1 + 2L \cdot (x_2 + v_2) - Lv_3), \end{aligned} \tag{14}$$

where v_1 and v_2 are realizations of the normal distribution with zero mean and standard deviation σ_M and σ_T, respectively; and v_3 is a realization of P. It is noted that v_1 and v_2 denote deviations of M and T from their means x_1 and x_2, respectively. The system event consists of three cut-sets such that $E_{\text{system}} = E_1E_2 \cup E_3E_4 \cup E_3E_5$.

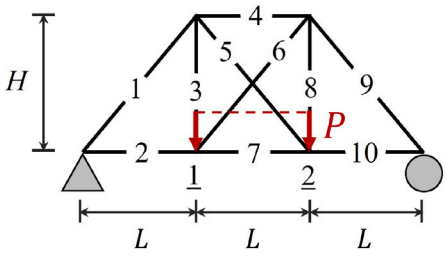

Fig. 2. Example truss structure system.

Using the S-BORM algorithm, the computed solution is $(x_1, x_2) = (1297, 150.0)$, resulting in estimated buffered failure probability $\hat{\bar{p}}_f = 9.985 \cdot 10^{-4}$ and cost 2743. For comparison, a grid search is performed by discretizing each variable into 100 intervals, which leads to 10.1 and 1.01 spacing for x_1 and x_2, respectively. The search identifies the best point as (1298, 149.0) leading to cost 2745, $\hat{\bar{p}}_f = 9.935 \cdot 10^{-4}$, and $\hat{p}_f = 2.653 \cdot 10^{-4}$. This agrees with the solution computed by the proposed algorithm. On the other hand, using the grid search results, we compare the computed solution with the one obtained with the conventional failure probability. By setting the target failure probability as $p_f^t = 2.653 \cdot 10^{-4}$, we obtain solution (1298, 135.9) with cost 2732, $\hat{\bar{p}}_f = 12.21 \cdot 10^{-4}$, and $\hat{p}_f = 2.678 \cdot 10^{-4}$. As the two solutions are similar especially in regard to the costs, the result confirms the similarity in optimal solutions between the two failure probabilities. The minor difference between the two solutions is expected as discussed in Section 2.2.

The optimization requires a marginal computational efforts taking 0.1 s. It runs 7 outer loops and 7 rounds of gradient evaluations, which implies that only 7 evaluations of limit-state functions are made for each sample and 7 gradient evaluations are made for each active sample. A summary of the results can be found in Table 1.

4.3. Design of indeterminate truss bridge system

Consider the truss bridge structure in Fig. 2. The structure consists of 10 members with strength R_i, $i = 1, \ldots, 10$, which are independent normal random variables with mean $\mu_R = 276$ and standard deviation $\sigma_R = 13.8$. The parameters of member lengths are set as $H = 1.6$ and $L = 2$. The truss structure is subjected to load P exerted on nodes 1 and 2, which follows the normal distribution with mean $\mu_P = 190$ and standard deviation $\sigma_P = 19$. Then, the design variables are cross-section areas of the members. To reflect the practical aspect of construction, the members are grouped into four sets $\{1, 2, 9, 10\}$, $\{3, 8\}$, $\{4, 7\}$, and $\{5, 6\}$. Within a group, members have the same cross-section area, which is represented by a design variable $x_d \in [1, 2]$, $d = 1, \ldots, 4$. The cost function is the total volume of the members, i.e., $c(\boldsymbol{x}) = \sum_{q=1}^{10} l_q \cdot x_{d(q)}$, where l_q is the length of member q, and $d(q)$ denotes the index of the group that member q belongs to, e.g., $d(1) = 1$ and $d(3) = 2$.

The system failure is defined as an occurrence of structural instability, which can be identified by performing a sequence of linear analyses. For example, a linear analysis is performed with member 1 removed; then, since this leads to a singular matrix of member forces, member 1 constitutes a failure mode. Another example is that, while a failure of member 3 does not lead to a singular matrix, a following failure of member 7 does. This concludes a failure mode consisting of member 3 and member 7. Such failure of member q at failure mode k can be represented by a limit-state function

$$g_{kq}(\boldsymbol{x}, \boldsymbol{v}) = v_0 \cdot \delta_{kq} - x_{d(q)} \cdot v_q, \tag{15}$$

where v_0 is a realization of P; δ_{kq} is the force experienced by member q in failure mode k when a unit force is applied to nodes 1 and 2; and v_q, $q = 1, \ldots, 10$, is a realization of R_q. It is noted that δ_{kq} for member q varies depending on failure mode k (e.g., member 7 experiences

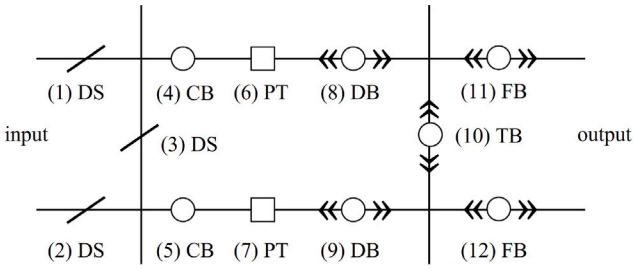

Fig. 3. Example electrical system.
Source: Figure recreated from [37].

different forces between when all members are in place and when member 3 is removed).

For this structure, 108 failure modes are identified. Then, each failure mode can be represented as a cut-set, whereby the system event is defined as a link-set of those cut-sets, i.e.,

$$E_{\text{sys}} = \bigcup_{k=1}^{108} \bigcap_{q \in \mathbb{Q}_k} E_{kq},$$

where $\mathbb{Q}_k$ denotes the index set of members that constitute a failure mode k.

As summarized in Table 1, a solution is obtained as (1.586, 1.000, 1.459, 1.000) with cost 28.63, $\hat{\tilde{p}}_f = 9.735 \cdot 10^{-4}$, and $\hat{p}_f = 3.654 \cdot 10^{-4}$. The optimization takes 13.6 s with 3 rounds of outer loops and 3 rounds of gradient evaluations.

4.4. Testing time allocation on electrical components

Consider the two transmission-line electrical substation system in Fig. 3 [37]. The system consists of 12 components, which either fail or survive. Then, system failure is defined as a disconnection between the input and output nodes. Each component type is under a test phase, and we aim to find an optimal allocation of testing time over the component types. There are 6 component types, i.e., disconnect switch (DS), circuit breaker (CB), power transformer (PT), drawout breaker (DB), tie breaker (TB), and feeder breaker (FB), and their testing time is denoted by design variables $x_1, \dots, x_6$, respectively. We assume that their reliability growth follows the non-homogeneous Poisson process (NHPP) RGM [38], by which the fault rate of component q of type $d(q)$, $q = 1, \dots, 12$ is defined as

$$\lambda_q(x_{d(q)}) = \frac{\alpha\beta}{\exp(\beta x_{d(q)})}, \tag{16}$$

where the parameters are set as $\alpha = 9$ and $\beta = 2$. The cost function is the total testing time, i.e., $c(x) = \sum_{d=1}^{4} x_d$. On the other hand, for each component q, the limit-state function is defined as

$$\begin{aligned} g_q(x_{d(q)}, v_q) &= \Delta T - \frac{-\ln v_q}{\lambda_q(x_{d(q)})} \\ &= \Delta T + \ln v_q \cdot \frac{\exp(\beta x_{d(q)})}{\alpha\beta}, \end{aligned} \tag{17}$$

where $\Delta T = 365$ is the target operation time, and ξ_q, $q = 1, \dots, 12$, is a realization of the uniform distribution $U[0, 1]$. The system consists of 25 minimum cut sets {(1, 2), (4, 5), (4, 7), (4, 9), (5, 6), (6, 7), (6, 9), (5, 8), (7, 8), (8, 9), (11, 12), (1, 3, 5), (1, 3, 7), (1, 3, 9), (2, 3, 4), (2, 3, 6), (2, 3, 8), (4, 10, 12), (6, 10, 12), (8, 10, 12), (5, 10, 11), (7, 10, 11), (9, 10, 11), (1, 3, 10, 12), (2, 3, 10, 11)} [37].

As summarized in Table 1, a solution is computed as (7.017, 7.047, 7.095, 7.024, 1.000, 7.016) with cost 36.20, $\hat{\tilde{p}}_f = 9.635 \cdot 10^{-4}$, and $\hat{p}_f = 4.179 \cdot 10^{-4}$. Since all component types are assigned an identical RGM, difference in testing time arises solely from varying topological importance. The optimization takes 2.83 s with 10 rounds of evaluations of limit-state functions and 10 rounds of gradient evaluations of active samples.

5. Parametric test of the proposed algorithm

5.1. Algorithm parameters

While default values of the algorithm parameters are proposed in Section 4.1, we test the robustness of the algorithm by running optimization with their values changed for the three examples in Sections 4.2, 4.3, and 4.4. First, experiments are performed by changing λ to 0.005, 0.02, 0.04, 0.08, and 1. The results are summarized in Table 2. As illustrated in the table, the parameter does not cause notable differences. In all of the three examples, the computation time remains stable taking less than 1 min. Also, for other results including the number of outer loops and gradient evaluations, cost, $\hat{\tilde{p}}_f$, and $\hat{p}_f$, highest and lowest values all remain very close. Similarly, θ is changed to 0.25, 0.5, 2, 4, and 8. Table 3 summarizes the highest and lowest values of various results. Again, the results do not show notable variances. Finally, ω is tested with values 1.2, 1.5, 2, 3, and 5, as illustrated in Table 4. This parameter also does not lead to meaningful differences.

Although the parameters are found to have insignificant influences, there are still marginal variances in computation time and quality of solutions (i.e., the cheapest solution that satisfies reliability constraints). Numerical experiments suggest that the proposed default values show the most stable performance. Nevertheless, one may alter their values to fit the characteristics of a given problem. For example, one may increase λ if solutions are too slow to converge, jumping between distant regions. The parameter θ can be increased if the algorithm finds it hard to satisfy reliability constraints. The parameter ω needs to be increased if active sets change wildly as an optimal solution is updated, or decreased if evaluation of limit-state functions is costly.

5.2. Target buffered failure probability

To test its stability to the magnitude of target buffered failure probability (i.e., $\bar{p}_f^t$), the algorithm is tested with target values $1 \cdot 10^{-2}$ and $1 \cdot 10^{-4}$. The optimization results are summarized in Table 5. In the table, comparisons are made in parentheses with the results in Table 1 with target probability $1 \cdot 10^{-3}$. As expected, in all examples, the cost of obtained solutions increases with a higher $\bar{p}_f^t$. It is noted that computing time and the number of iterations remain stable, while the estimated buffered failure probabilities remain close but lower than the target values. This demonstrates the robustness of the algorithm with respect to the level of $\bar{p}_f^t$.

5.3. Initial points

Since problems of interest are nonconvex, the quality of computed solutions is most dependent on initial points. To test this, for each example, 100 experiments are performed with different initial points sampled by Latin hypercube sampling. The best solutions of each example, i.e., the solution that satisfies reliability constraints and incur the lowest cost, are summarized in Table 6. By considering a solution the best if it yields a cost less than 3% higher than the lowest obtained value, it is observed that 100%, 46%, and 7% of the initial points lead to one of the best solutions in the first, second, and third examples, respectively. This suggests that the third example has in particular multiple local optima. It is noted that the results in Table 1, which are obtained with initial solutions as a midpoint of the lower and upper bounds, are all one of the best obtained solutions.

Nonetheless, it is unavoidable to fall in local solutions as nonconvexity is highly likely for most practical problems. Therefore, we strongly suggest to try multiple initializations to ensure the quality of an obtained solution. The proposed algorithm is advantageous to this end owing to its computational efficiency as demonstrated by various tests illustrated above.

Table 1
Optimization results of three numerical examples.

Sec.	Time (s)	No. of outer loops (function calls)[a]	No. of serious steps (gradient calls)[b]	Computed solution	Cost	$\hat{\hat{p}}_f$ $(\cdot 10^{-4})$	$\hat{p}_f$ $(\cdot 10^{-4})$
4.2	0.136	7 (2,797,200)	7 (5600)	(1297, 150.0)	2743	9.985	2.678
4.3	13.6	3 (1,198,800)	3 (2400)	(1.586, 1.000, 1.459, 1.000)	28.63	9.735	3.654
4.4	2.83	10 (3,996,000)	10 (8000)	(7.017, 7.047, 7.095, 7.024, 1.000, 7.016)	36.20	9.860	4.429

[a](No. of outer loops)·(No. of samples) = (Total evaluation number of limit-state functions).
[b](No. of serious steps)·(No. of active samples) = (Total evaluation number of limit-state function gradients).

Table 2
Optimization results obtained by different parameters $\lambda = 0.005, 0.01, 0.02, 0.04, 0.08,$ and 1.

Sec.		Time (s)	No. of outer loops	No. of serious steps	Cost	$\hat{\hat{p}}_f$ $(\cdot 10^{-4})$	$\hat{p}_f$ $(\cdot 10^{-4})$
4.2	Highest	0.73 ($\lambda = 1$)	26 ($\lambda = 1$)	26 ($\lambda = 1$)	2743 ($\lambda = 0.005$)	9.985 (all λ)	3.103 ($\lambda = 1$)
	Lowest	0.14 ($\lambda = 0.01$)	7 ($\lambda = 0.01$)	7 ($\lambda = 0.01$)	2718 ($\lambda = 0.04$)	9.985 (all λ)	2.678 ($\lambda = 0.005$)
4.3	Highest	55 ($\lambda = 1$)	4 ($\lambda = 0.05$)	4 ($\lambda = 0.05$)	29.35 ($\lambda = 0.04$)	9.760 ($\lambda = 1$)	3.754 ($\lambda = 1$)
	Lowest	19 ($\lambda = 0.01$)	2 ($\lambda = 1$)	2 ($\lambda = 1$)	28.63 ($\lambda = 0.01$)	8.834 ($\lambda = 0.08$)	3.353 ($\lambda = 0.005$)
4.4	Highest	4.9 ($\lambda = 1$)	10 (all λ)	10 (all λ)	36.78 ($\lambda = 1$)	9.960 ($\lambda = 0.08$)	4.354 ($\lambda = 0.08$)
	Lowest	2.1 ($\lambda = 0.08$)	10 (all λ)	10 (all λ)	36.14 ($\lambda = 0.08$)	9.635 ($\lambda = 0.04$)	4.304 ($\lambda = 0.04$)

Table 3
Optimization results obtained by different parameters $\theta = 0.25, 0.5, 1, 2, 4,$ and 8.

Sec.		Time (s)	No. of outer loops	No. of serious steps	Cost	$\hat{\hat{p}}_f$ $(\cdot 10^{-4})$	$\hat{p}_f$ $(\cdot 10^{-4})$
4.2	Highest	0.21 ($\theta = 0.05$)	12 ($\theta = 0.25$)	10 ($\theta = 0.5$)	2743 ($\theta = 0.25$)	9.985 (all θ)	3.128 ($\theta = 8$)
	Lowest	0.14 ($\theta = 1$)	3 ($\theta = 4$)	3 ($\theta = 4$)	2718 ($\theta = 4$)	9.985 (all θ)	2.678 ($\theta = 0.25$)
4.3	Highest	51 ($\theta = 0.5$)	4 ($\theta = 0.25$)	3 ($\theta = 8$)	28.82 ($\theta = 2$)	9.960 ($\theta = 2$)	3.879 ($\theta = 2$)
	Lowest	19 ($\theta = 1$)	2 ($\theta = 2$)	2 ($\theta = 2$)	28.61 ($\theta = 8$)	7.432 ($\theta = 4$)	2.928 ($\theta = 4$)
4.4	Highest	2.2 ($\theta = 2$)	10 (all θ)	10 (all θ)	39.21 ($\theta = 8$)	9.910 ($\theta = 2$)	4.455 ($\theta = 2$)
	Lowest	2.1 ($\theta = 4$)	10 (all θ)	10 (all θ)	36.15 ($\theta = 4$)	9.735 ($\theta = 0.25$)	4.204 ($\theta = 0.25$)

Table 4
Optimization results obtained by different parameters $\omega = 1.2, 1.5, 2, 3,$ and 5.

Sec.		Time (s)	No. of outer loops	No. of serious steps	Cost	$\hat{\hat{p}}_f$ $(\cdot 10^{-4})$	$\hat{p}_f$ $(\cdot 10^{-4})$
4.2	Highest	0.40 ($\omega = 5$)	10 ($\omega = 1.2$)	7 ($\omega = 2$)	2743 ($\omega = 1.2$)	9.985 (all ω)	3.05 ($\omega = 5$)
	Lowest	0.14 ($\omega = 1.2$)	8 ($\omega = 5$)	6 ($\omega = 1.2$)	2718 ($\omega = 3$)	9.985 (all ω)	2.678 ($\omega = 2$)
4.3	Highest	42 ($\omega = 1.2$)	6 ($\omega = 1.5$)	5 ($\omega = 1.5$)	29.29 ($\omega = 3$)	9.860 ($\omega = 1.2$)	3.779 ($\omega = 1.2$)
	Lowest	19 ($\omega = 2$)	2 ($\omega = 1.2$)	2 ($\omega = 1.2$)	28.61 ($\omega = 1.2$)	8.684 ($\omega = 1.5$)	3.353 ($\omega = 3$)
4.4	Highest	5.13 ($\omega = 5$)	10 (all ω)	10 (all ω)	38.39 ($\omega = 5$)	9.860 ($\omega = 2$)	4.330 ($\omega = 2$)
	Lowest	1.4 ($\omega = 1.2$)	10 (all ω)	10 (all ω)	36.06 ($\omega = 1.5$)	9.585 ($\omega = 1.5$)	4.054 ($\omega = 1.5$)

Table 5
Optimization results obtained by different target values of buffered failure probability (in parentheses, comparisons are made with results in Table 1.).

$\bar{p}_f^t = 0.1 \cdot 10^{-2}$							
Sec.	Time (s)	No. of outer loops	No. of serious steps	Computed solution	Cost	$\hat{\bar{p}}_f$	$\hat{p}_f$
4.2	0.131 (−0.00527)	8 (+1)	8 (+1)	(1092, 150.0)	2334 (−409.3)	$9.975 \cdot 10^{-3}$	$3.030 \cdot 10^{-3}$
4.3	40.4 (+26.8)	2 (−1)	2 (−1)	(1.501, 1.090, 1.338, 1.000)	27.73 (−0.6300)	$9.015 \cdot 10^{-3}$	$3.636 \cdot 10^{-3}$
4.4	3.21 (+0.387)	6 (−4)	6 (−4)	(6.435, 6.435, 6.435, 6.435, 2.218, 6.435)	34.39 (−1.809)	$9.949 \cdot 10^{-3}$	$4.268 \cdot 10^{-3}$
$\bar{p}_f^t = 0.1 \cdot 10^{-4}$							
4.2	0.561 (+0.425)	7 (+0)	7 (+0)	(1471, 150.0)	3091 (+348.2)	$9.976 \cdot 10^{-5}$	$3.100 \cdot 10^{-5}$
4.3	32.8 (+19.2)	3 (+0)	3 (+0)	(1.668, 1.000, 1.765, 1.000)	29.72 (+1.090)	$9.801 \cdot 10^{-5}$	$3.300 \cdot 10^{-5}$
4.4	5.80 (+2.98)	8 (−2)	8 (−2)	(7.615, 7.616, 7.617, 7.616, 1.000, 7.616)	39.08 (+2.881)	$9.801 \cdot 10^{-5}$	$4.350 \cdot 10^{-5}$

Table 6
Optimization results with varying initial solutions.

Sec.	Best optimal solution[a]	Cost	$\hat{\bar{p}}_f$ $(\cdot 10^{-4})$	$\hat{p}_f$ $(\cdot 10^{-4})$	Ratio of best solutions[b]
4.2	(1257, 150.0)	2709	9.960	3.203	100%
4.3	(1.581, 1.000, 1.443, 1.000)	28.52	9.835	3.679	46%
4.4	(7.215, 6.896, 7.391, 7.391, 1.000, 6.906)	36.80	6.857	3.053	7%

[a]The feasible solution leading to the lowest cost.
[b]Feasible solutions leading to a cost less than 3% higher than the lowest one.

6. Further extension of numerical examples

In this section, the S-BORM algorithm is further demonstrated by investigating variations of each of the numerical examples in Section 4. We investigate two aspects: (1) handling non-normal distributions and (2) handling large-scale systems, i.e., a large number of variables and cut sets. To this end, we modify the first two examples to have non-normal distributions (Sections 6.1 and 6.2) and enlarge the system size of the third example (Section 6.3).

To ensure the quality of the obtained solutions, in Section 6.1, an obtained solution is compared with the one obtained by a grid search. In Sections 6.2 and 6.3, since the problem size is too large to perform a grid search, we perform 30 runs of optimization with different initial points as recommended in Section 5.3. The initial points are randomly generated by Latin hypercube sampling.

6.1. Cantilever beam-bar system with truncated normal distributions

From the example in Section 4.2, we modify the distributions of M and T to truncated normal distributions by a range of $\pm 2\sigma$, while their means and standard deviations have the same values. All other definitions also remain the same. Then, the only required change for implementation is that now v_1 and v_2 in (14) are realizations of the normal distribution with zero mean and truncated by the range $[-2\sigma_M, 2\sigma_M]$ and $[-2\sigma_T, 2\sigma_T]$, respectively.

The S-BORM algorithm returns a solution computed as $(x_1, x_2) = (923.7, 119.9)$, which is cheaper than the solution in Section 4.2 because of the reduced variations in M and T. The solution leads to estimated buffered failure probability $\hat{\bar{p}}_f = 9.985 \cdot 10^{-4}$, cost 1967, and $\hat{p}_f = 3.629 \cdot 10^{-4}$. This also agrees with the result obtained by a grid search, which is obtained as (924.2, 119.7) with cost 1968, $\hat{\bar{p}}_f = 9.835 \cdot 10^{-4}$. On the other hand, by setting a target failure probability $p_f^t = 3.629 \cdot 10^{-4}$, we obtain from grid search an optimal solution (924.2, 115.6) with cost 1964, $\hat{\bar{p}}_f = 11.11 \cdot 10^{-4}$, and $\hat{p}_f = 3.604 \cdot 10^{-4}$. This shows that the two failure probabilities lead to similar optimal solutions. The details of the result is summarized in Table 7. The computational cost remains similar to the original setting, taking less than 1 s for computation.

6.2. Truss bridge system with Weibull and lognormal distributions

From the example in Section 4.3, we modify the distributions of R_q, $q = 1, \ldots, 10$, and P: R_q follows the lognormal distribution with mean $\mu_R = 276$ and standard deviation $\sigma_R = 13.8$ (i.e., the same mean and standard deviation as in the original example); and P follows the Weibull distribution with scale parameter 77.5 and shape parameter 1.5 (this leads P to have mean 70 and standard deviation 47.5). For implementation, this only changes that v_0 and v_q in (15) become realizations of the modified distributions of P and R_q, respectively.

As summarized in Table 7, the best solution among 30 obtained solutions is (1.863, 1.071, 1.657, 1.008) with cost 32.22, $\hat{\bar{p}}_f = 9.885 \cdot 10^{-4}$, and $\hat{p}_f = 3.629 \cdot 10^{-4}$, which is obtained from an initial point (1.998, 1.146, 1.902, 1.673). Is is noted that, compared to the original setting in Section 4.3, the increased variance of P results in a more expensive solution even though its mean is much lower.

6.3. Large-scale power network

This section investigates the power network in Fig. 4, which has been adopted and modified from [39]. The optimization task remains the same as in Section 4.4, i.e., allocation of testing time is optimized over 6 component types. The system has three input points and one output point, and system failure is defined as disconnection of the output point from all of the input points. The system consists of 70 components (i.e., there are 70 limit-state functions g_q, $q = 1, \ldots, 70$, defined as in Section 4.4), whose component types are as illustrated in Fig. 4. Their fault rates λ_q and limit-state functions g_q, $q = 1, \ldots, 70$ are defined as in (16) and (17), respectively.

To facilitate obtaining cut-sets, components are divided into 16 supercomponents as marked by square brackets in the figure. In terms of the supercomponents, 36 cut sets are obtained as {([10], [12]), ([15], [16]), ([5], [6], [16]), ([8], [9], [15]), ([10], [11], [15]), ([11], [12], [16]), ([4], [6], [13], [16]), ([5], [6], [8], [9]), ([5], [6], [10], [11]), ([7], [8], [14], [15]), ([8], [9], [11], [12]), ([1], [2], [7], [8], [15]), ([1], [3], [4], [6], [16]), ([4], [6], [8], [9], [13]), ([4], [6], [10], [11], [13]), ([5], [6], [7], [8], [14]), ([7], [8], [11], [12], [14]), ([1],

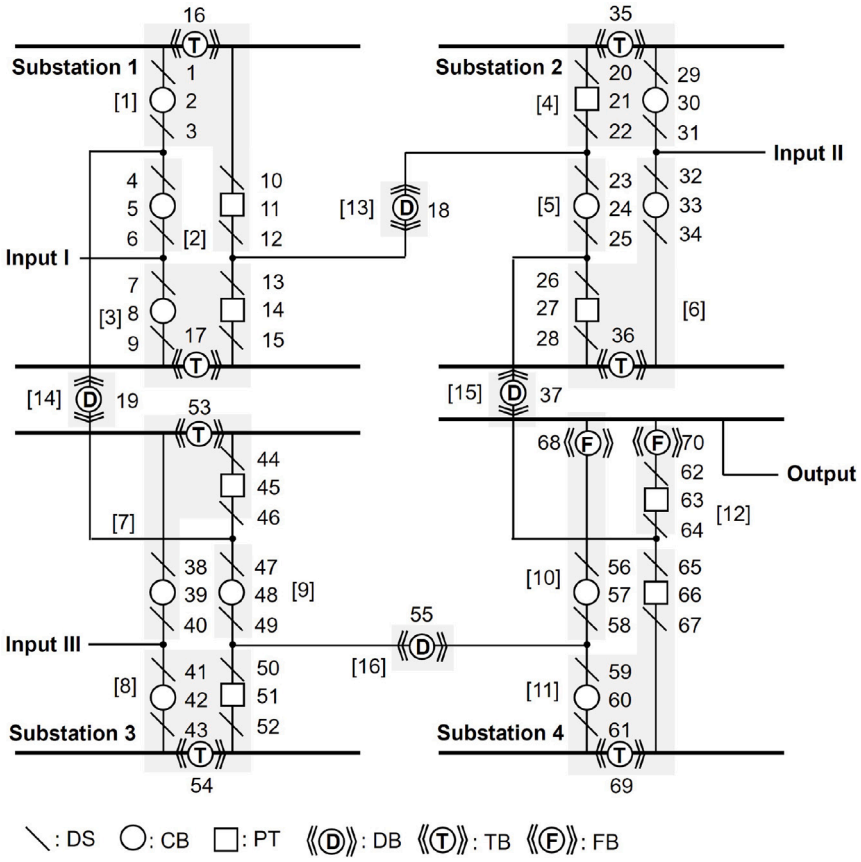

Fig. 4. Example power network.
Source: Adopted and modified from [39].

Table 7
Optimization results of variations of three numerical examples.

Sec.	Time (s)	No. of outer loops (function calls)[a]	No. of serious steps (gradient calls)[b]	Computed solution	Cost	$\hat{p}_f$ $(\cdot 10^{-4})$	$\hat{p}_f$ $(\cdot 10^{-4})$
6.1	0.220	7 (2,797,200)	6 (4800)	(923.7, 119.9)	1967	9.985	3.629
6.2	8.83	2 (799,200)	2 (1600)	(1.863, 1.071, 1.657, 1.008)	32.22	9.885	3.629
6.3	4996	5 (1,998,000)	4 (3200)	(9.304, 9.321, 1.000, 9.812, 1.000, 9.337)	39.77	8.058	4.279

[a](No. of outer loops)·(No. of samples) = (Total evaluation number of limit-state functions).
[b](No. of serious steps)·(No. of active samples) = (Total evaluation number of limit-state function gradients).

[2], [5], [6], [7], [8]), ([1], [2], [7], [8], [11], [12]), ([1], [3], [4], [6], [8], [9]), ([1], [3], [4], [6], [10], [11]), ([2], [3], [4], [6], [7], [8]), ([2], [3], [4], [6], [14], [16]), ([2], [3], [7], [8], [13], [15]), ([4], [6], [7], [8], [13], [14]), ([1], [2], [4], [6], [7], [8], [13]), ([1], [3], [4], [6], [7], [8], [14]), ([2], [3], [4], [5], [7], [8], [15]), ([2], [3], [4], [6], [7], [8], [9]), ([2], [3], [4], [6], [7], [9], [16]), ([2], [3], [4], [6], [8], [9], [14]), ([2], [3], [4], [6], [10], [11], [14]), ([2], [3], [5], [6], [7], [8], [13]), ([2], [3], [7], [8], [11], [12], [13]), ([2], [3], [4], [5], [7], [8], [11], [12]), ([2], [3], [4], [6], [7], [9], [10], [11])}.[5] Then, the cut sets of the original components can be obtained by taking the union of those cut sets of the supercomponents. For example, the first cut set ([10], [12]) results in 16 cut sets such that (56, 62), (56, 63), (56, 64), (56, 70), (57, 62), (57, 63), (57, 64), (57, 70), (58, 62), (58, 63), (58, 64), (58, 70), (68, 62), (68, 63), (68, 64), and (56, 70).[6] Such process leads to around $1.91 \cdot 10^6$ cut sets. For computational purpose, we consider cut sets with a number of components equal to or less than 5, which leaves 10,062 cut sets (corresponding to the first 17 cut sets of supercomponents). This does not undermine the validity of optimization results since the more the components in a cut set, the lower probability the cut has and, therefore, less influences on a system failure probability.

As summarized in Table 7, the best solution among the 30 obtained solutions is (9.304, 9.321, 1.000, 9.812, 1.000, 9.337) with cost 39.77, $\hat{p}_f = 8.058 \cdot 10^{-4}$, and $p_f = 4.279 \cdot 10^{-4}$. This solution is obtained from an initial point (9.631, 9.876, 1.766, 4.627, 7.155, 3.932). The optimization takes 4996 s, which is much longer than for the original

[5] The cut sets have been obtained following similar steps explained in [39].

[6] Note that supercomponent [10] consists of components 56, 57, 58, and 68, and [12] of 62, 63, 64, and 70.

setting in Section 4.4 owing to the increased number of random variables and cut sets. On the other hand, there is no increase in the number of evaluations of limit-state functions and their gradients as there are 5 and 4 rounds, respectively.

7. Conclusions

This study proposes an efficient algorithm for reliability-based optimization (RBO), particularly for general system events that are represented as a link-set of cut-sets. To handle such general systems, we evaluate system reliability by realizations of random variables (i.e., samples or data points) instead of analytical calculation that requires problem-specific formulas. This can be done by employing the buffered optimization and reliability method (BORM) that replaces the conventional failure probability by the buffered failure probability. We call it the *S-BORM* algorithm.

The S-BORM algorithm efficiently solves RBO problems of general systems by leveraging four ideas. First, a reliability constraint is penalized and moved to the objective function. Second, limit-state functions are linearized adaptively at a current solution. Third, the modified objective function is reformulated as a difference-of-convex function so that its optimization can be solved by a difference-of-convex bundle method. Fourth, an active-set strategy is employed, which enables us to consider only a small subset of samples that are within or close enough to failure domains. We do not assume convexity either for cost function or for limit-state functions. This makes the S-BORM algorithm applicable for a wide class of systems. Although such minimal assumptions make it difficult (if not impossible) to theoretically guarantee global optimality and convergence, we provide justifications of the proposed approach (see the Appendix and the companion report [33] for more details) and empirical demonstrations by presenting newly designed numerical examples. Examples include complex and large-scale systems such as a power network with 70 random variables and 10,062 cut sets and combined with reliability growth models. We show that the S-BORM algorithm provides a handy means to solve these complex problems. Moreover, extensive parametric investigations show that the algorithm remains insensitive to algorithm parameter values and the magnitude of target failure probability.

Utilizing realizations of random variables greatly facilitates general applications as it eliminates the need for problem-specific derivations and enables non-parametric analysis. By leveraging such advantages, we develop a Matlab-based tool, which is available at https://github.com/jieunbyun/sborm. To run the algorithm, users only need to provide general information, i.e., cost functions and limit-state functions (including their gradients) and realizations of random variables. Meanwhile, this underlines a distinct potential of the BORM for further development of general software tools of reliability-based optimization. This is advantageous considering barriers of implementing RBO algorithms, which often require a high level of knowledge and engineering. Promising topics for being combined with such data-driven optimization include surrogate models, real-time data, and sequential decision-making.

CRediT authorship contribution statement

Ji-Eun Byun: Writing – original draft, Visualization, Software, Formal analysis. **Welington de Oliveira:** Writing – review & editing, Software, Methodology. **Johannes O. Royset:** Writing – review & editing, Validation, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

In the article, we have shared the link where data and codes can be downloaded.

Acknowledgments

The research of the first author is in part supported by the Humboldt Research Fellowship for Postdoctoral Researchers from Alexander von Humboldt Foundation, Germany. The second author acknowledges financial support from the Gaspard-Monge Program for Optimization and Operations Research (PGMO) project "SOLEM - Scalable Optimization for Learning and Energy Management". The research of the third author is supported in part by the Air Force Office of Scientific Research, Mathematical Optimization, United States (21RT0484).

Replication of results

The examples can be replicated by (Matlab-based) codes and data uploaded at https://github.com/jieunbyun/sborm.

Appendix. Algorithmic justifications

The function $F^\nu(\boldsymbol{x},\gamma;\theta^\nu,\hat{\boldsymbol{x}}^\nu)$, appearing in **Step 3** of the algorithm, can be written as the difference of two convex functions as required by [31, Alg. 1]; the specific formula is given in an extended arXiv report [33]. For limit-state functions that are linear in $\boldsymbol{x}$, S-BORM converges to critical points as defined in [33]. Further motivation for the algorithm follows below.

Generally, $F^\nu(\boldsymbol{x},\gamma;\theta^\nu,\hat{\boldsymbol{x}}^\nu)$ is an approximation of the actual function $F(\boldsymbol{x},\gamma;\theta^\nu)$ but it becomes exact at the current point $\hat{\boldsymbol{x}}^\nu$, i.e.,

$$F^\nu(\hat{\boldsymbol{x}}^\nu,\gamma;\theta^\nu,\hat{\boldsymbol{x}}^\nu) = F(\hat{\boldsymbol{x}}^\nu,\gamma;\theta^\nu), \quad \forall\,\gamma.$$

Therefore, since $(\hat{\boldsymbol{x}}^\nu,\hat{\gamma}^\nu)$ is feasible to the subproblem of **Step 3**, we have that

$$\begin{aligned} &F^\nu(\boldsymbol{x}^{\nu+1},\gamma^{\nu+1};\theta^\nu,\hat{\boldsymbol{x}}^\nu) + \frac{\lambda^\nu}{2}\|\boldsymbol{x}^{\nu+1}-\hat{\boldsymbol{x}}^\nu\|_2^2 + \frac{\lambda^\nu}{2}(\gamma^{\nu+1}-\hat{\gamma}^\nu)^2 \\ &\quad \le F^\nu(\hat{\boldsymbol{x}}^\nu,\hat{\gamma}^\nu;\theta^\nu,\hat{\boldsymbol{x}}^\nu) = F(\hat{\boldsymbol{x}}^\nu,\hat{\gamma}^\nu;\theta^\nu). \end{aligned}$$

This means that the predicted decrease ζ^ν is non-negative and S-BORM is a descent method in this sense.

If **Step 3** produces $(\boldsymbol{x}^{\nu+1},\gamma^{\nu+1}) = (\hat{\boldsymbol{x}}^\nu,\hat{\gamma}^\nu)$, then one can show that this point is critical for the penalized and linearized problem (11) (with $\theta=\theta^\nu$ and $\hat{\boldsymbol{x}}=\hat{\boldsymbol{x}}^\nu$). A similar conclusion appears to hold more generally too. Suppose that $\texttt{tol}=0$ and consider two cases.

1. The algorithm produces only finitely many serious steps followed by an infinite sequence of null steps. Then, $(\hat{\boldsymbol{x}}^\nu,\hat{\gamma}^\nu)$ equals some fixed point $(\hat{\boldsymbol{x}},\hat{\gamma})$ for all ν after the last serious step. Furthermore, the linearization of limit-state functions are fixed for these iterations. The S-BORM algorithm therefore reduces to a penalized approach for solving

$$\min_{\boldsymbol{x}\in\mathbb{X},\gamma\in\mathbb{R}} \quad c(\boldsymbol{x}) \tag{A.1a}$$

$$\begin{aligned} \text{subject to} \quad & \gamma + \frac{1}{\bar{p}_f^t}\sum_{n=1}^{N} p_n \max\Big\{0, \max_{k\in1,\dots,K}\min_{q\in\mathbb{Q}_k} g_q(\hat{\boldsymbol{x}},\boldsymbol{v}_n) \\ & + \langle\nabla g_q(\hat{\boldsymbol{x}},\boldsymbol{v}_n), \boldsymbol{x}-\hat{\boldsymbol{x}}\rangle - \gamma\Big\} \le 0. \end{aligned} \tag{A.1b}$$

As the penalized parameter becomes $\theta^{\max}$ after finitely many steps, the iterates produced by **Step 3** converges to $(\hat{\boldsymbol{x}},\hat{\gamma})$ as λ^ν increases indefinitely and the value $F^\nu(\boldsymbol{x},\gamma;\theta^{\max},\hat{\boldsymbol{x}})$ given in (12) does not change (because both the penalization parameter and the current solution candidate are fixed for all ν large enough). This argument suggests that $(\hat{\boldsymbol{x}},\hat{\gamma})$ is a critical point for the linearized problem. The theory of exact penalty functions asserts that if a condition on the constraint of (A.1) exists, and $\theta^{\max}$ is greater than the largest optimal dual variable associated with this constraint, then solutions of the penalized problem (11) are solutions of (A.1); see, e.g., [40,41] and [24, Proposition 6.13].

2. The algorithm produces infinitely many serious steps. Again, recall that the penalty parameter becomes $\theta^{\max}$ after finitely many iterations. In this case, as already argued, the descent test ensures that the sequence of function values is non-increasing:

$$F(\hat{\boldsymbol{x}}^{\nu+1}, \hat{\gamma}^{\nu+1}; \theta^{\max}) \leq F(\hat{\boldsymbol{x}}^{\nu}, \hat{\gamma}^{\nu}; \theta^{\max}) - \kappa\zeta^{\nu}, \quad \text{with } \zeta^{\nu} \text{ given in Step 4.}$$

With the mild assumption that $F(\cdot, \cdot; \theta^{\max})$ has bounded level sets, a simple recursive argument (the telescope sum) on the above inequality shows that $\zeta^{\nu} \to 0$ and one can argue that any cluster point of the sequence of serious steps $\{(\hat{\boldsymbol{x}}^{\nu}, \hat{\gamma}^{\nu})\}$ is critical for the penalized problem (11). Once again, the penalization arguments concerning (11) and (A.1) apply.

Summary. The above arguments tell us that S-BORM always terminates after finitely many steps provided that $\texttt{tol} > 0$. When $\texttt{tol} = 0$ and the limit-state functions are linear in $\boldsymbol{x}$, then the algorithm is guaranteed to converge to a critical point. For more general limit-state functions, the discussion on the asymptotic behavior of the algorithm suggests that the approach computes a critical point for the linearized problem (A.1), with $\hat{\boldsymbol{x}}$ being the x-part of the last serious step, or an arbitrary cluster point (if any) of $\{\hat{\boldsymbol{x}}^{\nu}\}_{\nu=1}^{\infty}$. A full mathematical argument is beyond the scope of the present paper.

References

[1] Enevoldsen I, Srensen JD. Reliability-based optimization in structural engineering. Struct Saf 1994;15(3):169–96.
[2] Byun J-E, Song J. Efficient probabilistic multi-objective optimization of complex systems using matrix-based Bayesian network. Reliab Eng Syst Saf 2020;200:106899.
[3] Byun J-E, Song J. A general framework of Bayesian network for system reliability analysis using junction tree. Reliab Eng Syst Saf 2021;216:107952.
[4] Lim H-W, Song J. Efficient risk assessment of lifeline networks under spatially correlated ground motions using selective recursive decomposition algorithm. Earthq Eng Struct Dyn 2012;41(13):1861–82.
[5] Marti K. Optimal structural design under stochastic uncertainty by stochastic linear programming methods. Reliab Eng Syst Saf 2001;72(2):165–77.
[6] Youn BD, Choi KK, Park YH. Hybrid analysis method for reliability-based design optimization. J Mech Des 2003;125(2):221–32.
[7] Li F, Liu J, Wen G, Rong J. Extending SORA method for reliability-based design optimization using probability and convex set mixed models. Struct Multidiscip Optim 2019;59:1163–79.
[8] Nguyen TH, Song J, Paulino GH. Single-loop system reliability-based topology optimization considering statistical dependence between limit-states. Struct Multidiscip Optim 2011;44:593–611.
[9] Royset JO, Der Kiureghian A, Polak E. Reliability-based optimal structural design by the decoupling approach. Reliab Eng Syst Saf 2001;73(3):213–21.
[10] An X, Huang B, Shi D. A novel reliability index approach and applied it to cushioning packaging design. Adv Mech Eng 2022;14(11):16878132221135992.
[11] Bismut E, Pandey MD, Straub D. Reliability-based inspection and maintenance planning of a nuclear feeder piping system. Reliab Eng Syst Saf 2022;224:108521.
[12] Jerez DJ, Jensen HA, Valdebenito MA, Misraji MA, Mayorga F, Beer M. On the use of directional importance sampling for reliability-based design and optimum design sensitivity of linear stochastic structures. Probab Eng Mech 2022;70:103368.
[13] Yuan X, Lu Z. Efficient approach for reliability-based optimization based on weighted importance sampling approach. Reliab Eng Syst Saf 2014;132:107–14.
[14] Li X, Gong C, Gu L, Jing Z, Fang H, Gao R. A reliability-based optimization method using sequential surrogate model and Monte Carlo simulation. Struct Multidiscip Optim 2019;59:439—460.
[15] Kim J, Song J. Reliability-based design optimization using quantile surrogates by adaptive Gaussian process. ASCE J Eng Mech 2021;147(5):04021020.
[16] Yang S, Lee M, Lee I. A new sampling approach for system reliability-based design optimization under multiple simulation models. Reliab Eng Syst Saf 2023;231:109024.
[17] Perdikaris P, Venturi D, Royset JO, Karniadakis GE. Multi-fidelity modelling via recursive co-kriging and Gaussian–Markov random fields. Proc. R. Soc. A 2015;471:20150018.
[18] Canelas A, Carrasco M, López J. A new method for reliability analysis and reliability-based design optimization. Struct Multidiscip Optim 2019;59:1655–71.
[19] Liu K, Paulino GH, Gardoni P. Segmental multi-point linearization for parameter sensitivity approximation in reliability analysis. Struct Saf 2016;62:101–15.
[20] Ding Y, Hu Y, Li D. Redundancy optimization for multi-performance multi-state series-parallel systems considering reliability requirements. Reliab Eng Syst Saf 2021;215:107873.
[21] Okoro A, Khan F, Ahmed S. Dependency effect on the reliability-based design optimization of complex offshore structure. Reliab Eng Syst Saf 2023;231:109026.
[22] Li S, Chi X, Yu B. An improved particle swarm optimization algorithm for the reliability–redundancy allocation problem with global reliability. Reliab Eng Syst Saf 2022;225:108604.
[23] Wang J, Katafygiotis LS. Reliability-based optimal design of linear structures subjected to stochastic excitations. Struct Saf 2014;47:29–38.
[24] Royset JO, Wets RJ-B. An optimization primer. 1st ed.. Springer Cham; 2021.
[25] Royset JO, Polak E. Extensions of stochastic optimization results to problems with system failure probability functions. J Optim Theory Appl 2007;133:1–18.
[26] Rockafellar RT, Royset JO. On buffered failure probability in design and optimization of structures. Reliab Eng Syst Saf 2010;95(5):499–510.
[27] Byun J-E, Royset JO. Data-driven optimization of reliability using buffered failure probability. Struct Saf 2022;98:102232.
[28] Royset JO, Byun J-E. Gradients and subgradients of buffered failure probability. Oper Res Lett 2021;49(6):868–73.
[29] Chaudhuri A, Kramer B, Norton M, Royset JO, Willcox K. Certifiable risk-based engineering design optimization. AIAA J 2022;60(2):551–65.
[30] Tuy H. Convex analysis and global optimization. 2nd ed.. Springer; 2016.
[31] de Oliveira W. Proximal bundle methods for nonsmooth DC programming. J Global Optim 2019;75(2):523–63.
[32] van Ackooij W, de Oliveira W. Level bundle methods for constrained convex optimization with various oracles. Comput Optim Appl 2014;57(3):555–97.
[33] Byun J-E, de Oliveira W, Royset JO. S-BORM: Reliability-based optimization of general systems using buffered optimization and reliability method. 2022, arXiv:2209.02573.
[34] Pietrantuono R, Russo S, Trivedi KS. Software reliability and testing time allocation: An architecture-based approach. IEEE Trans Softw Eng 2010;36(3):323–37.
[35] Bertolino A, Miranda B, Pietrantuono R, Russo S. Adaptive test case allocation, selection and generation using coverage spectrum and operational profile. IEEE Trans Softw Eng 2019;47(5):881–98.
[36] Song J, Der Kureghian A. Bounds on system reliability by linear programming. ASCE J Eng Mech 2003;129(6):627–36.
[37] Der Kureghian A, Ditlevsen OD, Song J. Availability, reliability and downtime of systems with repairable components. Reliab Eng Syst Saf 2007;92(2):231–42.
[38] Byun J-E, Song J. Reliability growth analysis of k-out-of-N systems using matrix-based system reliability method. Reliab Eng Syst Saf 2017;165:410–21.
[39] Der Kiureghian A, Song J. Multi-scale reliability analysis and updating of complex systems by use of linear programming. Reliab Eng Syst Saf 2008;93(2):288–97.
[40] Han S, Mangasarian OL. Exact penalty functions in nonlinar programming. Math Program 1979;17(1):251–69.
[41] Le Thi HA, Pham Dinh T, Ngai HV. Exact penalty and error bounds in DC programming. J Global Optim 2012;52(3):509–35.