Applied Soft Computing Journal 90 (2020) 106154

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

KASRA: A Kriging-based Adaptive Space Reduction Algorithm for global optimization of computationally expensive black-box constrained problems

Hossein Akbari, Afshin Kazerooni *

Department of Mechanical Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran

ARTICLE INFO

Article history:
Received 30 May 2019
Received in revised form 29 January 2020
Accepted 30 January 2020
Available online 10 February 2020

Keywords:
Surrogate-based optimization
Efficient global optimization
Expensive black-box problems
Kriging model
Space reduction approaches

ABSTRACT

Efficient Global Optimization (EGO) methodology over the entire design space can be considerably time-consuming as much as the expensive simulation computer codes on High-multimodal and computationally Expensive Black-box (HEB) constrained problems. This paper introduces a strategy specifically, the Kriging-based Adaptive Space Reduction Algorithm, named KASRA, to enhance the performance of EGO for HEB constrained optimization problems. A new measure is proposed according to the activity of the decision variables to adaptively reduce the size of design intervals centered at the current best solution. The shrunken intervals gradually are expanded to decrease the risk of missing the desirable region. The design sub-spaces are explored based on the weighed constrained expected improvement criterion. The weighting coefficients of exploration and exploitation dynamically are regulated according to the volume ratio of the current hyper-box-shaped region and the original one. The sequential quadratic programming and exponential tunneling algorithms as two local and global optimizers are employed on Kriging-based functions to achieve a more accurate solution at the end of the procedure if necessary. The genetic algorithm with different tuning strategies is used to defeat the extreme time challenge of constructing Kriging-based surrogates. The proposed algorithm is applicable even if there is no feasible point in the initial samples. The efficiency of KASRA is demonstrated on twenty-two mathematical and ten classical engineering benchmark problems. Experimental results and comparative studies confirm that the proposed approach has a promising performance to deal with HEB constrained optimization problems and generally performs better than the competitor methods on most of the benchmark problems.

1. Introduction

High-performance computing facilities (hardware) with computer-based codes (software) such as Finite Element Methods (FEM) and Computational Fluid Dynamics (CFD) provide the modern resources that progressively allow engineers to analyze efficiently the simulation-based complex engineering problems that are difficult to study directly. One of the important outcomes of developing these simulation/analysis resources is to increase efforts to optimize the problems with High-multimodal and computationally Expensive Black-box objective and constraint functions (HEB problems) [1]. Despite all these tremendous progress, simulation-based analysis on each individual design can become prohibitive depending on the fidelity level of the simulation code; hence, the optimization cost and CPU time saving will be a challenging task, particularly according to the global optimization loops where searching thoroughly the entire design space needs the massive calls and simulations. All traditional global optimization methods including deterministic mathematical techniques and stochastic nature inspired meta-heuristic algorithms, to require a rather large number of calculations [2]. Therefore, in cases where total evaluation budget or time is a limiting factor – usually in engineering applications with high fidelity computer simulations [3] – obtaining the global optimum at the acceptable and efficient computational burden will be obligatory.

In recent decades the approach leads to the approximation methods, known as Surrogate-Based Global Optimization (SBGO), to produce low-cost but still acceptable mathematical models as a replacement for original functions of the simulation-based optimization problem. The basic idea of approximation is sampling based on the methodologies of Design and Analysis of the Computer Experiments (DACE) [4], to provide a surrogate model referred to as response surface or meta-model that mimics the behavior of each function of the HEB problem. A variety of surrogate models or predictors including polynomials, Radial Basis

* Corresponding author.
E-mail addresses: h.akbari@sru.ac.ir (H. Akbari), kazerooni@sru.ac.ir (A. Kazerooni).

https://doi.org/10.1016/j.asoc.2020.106154
1568-4946/

Functions (RBF), Artificial Neural Networks (ANN), Support Vector Regression (SVR), and Kriging has been developed in the literature and guidelines to select and construct a proper surrogate are discussed in some detailed reviews [1,5–8]. The Kriging-Based Surrogate Model (KBSM) as a Gaussian process has acquired reputation because of its flexibility to imitate complicated functions while having a random error estimate between the true and the surrogate functions to quantitatively measure the uncertainty in prediction.

The real-world HEB problems in modern engineering design which have many constraints are often challenging to fit the accurate global surrogates for the entire actual search space at one-shot experimental designs—they generally require a relatively large number of the initial sample points. Therefore, the Efficient Global Optimization (EGO) algorithm [9,10], also called adaptive sampling [7,11], as an alternative is used to economically build accurate meta-models in the promising regions of design landscape with as few points as possible. The EGO algorithm starts with DACE-based initial sampling such as space-filling Latin Hypercube Design (LHD) technique [4,12] and approximates the underlying functions, using a surrogate model; the model must facilitate estimation of the model uncertainty, e.g. Kriging. EGO continues with sequentially updating single or multiple infill-point(s) using the variance-based criteria such as estimated root Mean Square Error (MSE), Probability of Improvement (PI), Expected Improvement (EI), Lower Confidence Bound (LCB), Expected Prediction Error (EPE), or a combination/modification of several criteria [7,8,11,13–16]. Surrogate or ensemble of surrogates was then refined in an iterative process to gradually search the global optimum.

Although no formal optimization process calling expensive simulation is used to search global optimum, the procedure on infill criteria to update a new infill point is a global sub-optimization problem. Meanwhile, iteratively hyperparameter tuning of the current KBSMs of the objective and constraint functions by maximizing the natural logarithm of likelihood estimation is itself another global sub-optimization problem—hyperparameter tuning can be substantially time-consuming when training data is increasing because it needs the inversion of a larger covariance matrix for each surrogate several times [8]. These issues can lead to a slow convergence rate, especially when the dimensions are high and range of each design variable conservatively considered large on HEB problems with many peaks and valleys. In addition, the performance of EGO-based methods is quite sensitive to the choice of the initial space-filling design [17,18]—an initial rough estimate can fail the convergence.

The design space reduction/elimination (DSR) is an approach that has gained more attention to accelerate the convergence of the SBGO. The concept is step by step controlled focusing on better approximated promising regions and managing the optimization process over these regions to increase the chance to find the global optimum quickly. Based on this concept, this paper attempts to mild discussed limitations of the traditional EGO through proposing a strategy named Kriging-based Adaptive Space Reduction Algorithm (KASRA) involving five main elements: (1) shrinking the size of each design interval centered at the coordinate of the current best point with a proposed self-adapting capability based on the activity of the decision variables through KBSM; (2) gradually expanding the size of each reduced interval to guarantee global search on total design domain; (3) searching each design sub-space based on weighed constrained expected improvement (WCEI) criterion while the weighting factor of global exploration and local exploitation search of promising sub-space dynamically is set according to the volume ratio of current design domain and original one; (4) changing the hyperparameters tuning degree by limiting the number of function evaluations of genetic algorithm to defeat the time-consuming challenge of fitting KBSMs; (5) using the sequential quadratic programming (SQP) [19], and exponential tunneling algorithms [20,21], as the local and global optimizers on Kriging-based functions to get closer the right solution at the final step if needed. The space-filling LHD is used to generate the initial starting points by this explanation that KASRA does not depend on the feasibility of the initial points.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 briefly overviews the dependent theoretical aspects and background techniques. Section 4 explains the main components of KASRA procedure in detail. Section 5 describes the computational experiments while the performance of KASRA is discussed and graphically illustrated with two examples. Section 6 compares the numerical results obtained by the proposed algorithm with the alternative methods and discusses its unique features and limitations on a wide set of benchmark constrained global optimization problems (up to 20-dimensional and 38-constraints). Section 7 concludes the paper with a summary.

2. Related work

The methods on DSR can be classified into two categories: (1) reducing the dimensionality; (2) reducing the size of variables interval. Some research considered reducing the number of design variables [22], while other researches, also this work, allocated to contract the original range of the variables alternately.

Wang et al. [23], introduced the Adaptive Response Surface Method (ARSM) for functions of an overall convex shape. The ARSM maps the objective function value to a continuously reduced design space. Wang and Simpson [24], applied the fuzzy clustering on many inexpensive points and systematically reduced the design space as an intuitive method. Wang and Shan [25], proposed a DSR method according to a rough set for multi-objective optimization and robust design optimization problems. This method can support simultaneous computations and provide design engineers to focus on smaller design space to optimize more. Melo et al. [26] developed the Domain Optimization Algorithm (DOA) to reduce the search space. DOA can effectively remove the non-promising regions where, with high probability, there is no global optimum [26]. Regis and Shoemaker [17], introduced a controlled Gutmann-RBF (CG-RBF) method to globally optimize the expensive functions. The search in each iteration of the CG-RBF is limited to a sub-region of the design domain placed around a global minimum of the current RBF model. Younis et al. [27], introduced a new global optimization algorithm based on design experiments, region elimination and response surface modeling, namely, the Approximated Unimodal Region Elimination (AUMRE) method. The main parts of AUMRE include: (1) partitioning the design space into several unimodal regions using the design of the experiment data; (2) identifying and ranking these regions that most possibly involve the global minimum; (3) determining the minimum of sub-areas constructed based on the response surface model; (4) recognizing the global optimum of the problem through comparing the optima of all sub-areas. An extension of AUMRE specifically, Space Exploration and Unimodal Region Elimination (SEUMRE), is presented by Younis and Dong [28]. In the SEUMRE algorithm, Kriging models are fitted over sub-regions using LHD with the added design of experiment data. Gu et al. [29], introduced the HAM algorithm to be a global optimization method with Hybrid and Adaptive meta-models. The search is conducted by the HAM algorithm within a nearly small main region to improve search efficiency. The strategy for identifying the major region has several iterations over the entire design space. When the promising region gets small enough, the

global optimum is observed. Long et al. [30], introduced a new intelligent space exploration strategy named Significant Design Space (SDS) identification to improve the performance of ARSM. The SDS is a relatively small hypercube sub-region where local or global optima probably locate [30]. The SDS automatically is moved, contracted, or enlarged according to the location of the best sample, the size of the current design space, and the fitting quality of the current RSM. Liu et al. [31], introduced a sequential sampling design using the Monte Carlo method and Space Reduction strategy (MCSR). Regis [18] developed a Trust Region Implementation in Kriging-based optimization with Expected improvement (TRIKE). TRIKE selects infill points by maximizing the EI criterion for a trust region constructed around the current best solution. The trust region is regulated depending on the ratio of the actual improvement to the EI. Regis [18], also developed a variant of EGO called CYCLONE (CYClic Local search in OptimizatioN using Expected improvement) for bound constrained optimization problems that applies a cyclic pattern to determine the sizes of the search regions around the current best solution. Dong et al. [32], introduced a new Multi-Start approach for surrogate-based global optimization with a specific Space Reduction strategy (MSSR). The MSSR alternately explores the original design space or global space (GS), the reduced medium space (MS) that contains a promising region, and the local space (LS), with different ranges. The sizes of MS and LS change adaptively based on the current best-obtained value in optimizing the KBSM [32]. The unknown area of the design space is explored using the estimated mean square error of the KBSM. Dong et al. [33], proposed Hybrid Surrogate-based Optimization using Space Reduction (HSOSR) using Kriging and RBF. The proposed optimization framework uses the estimated MSE of Kriging to escape from a local valley. Dong et al. [34], recently extended the MSSR strategy for constrained problems and suggested SCGOSR (Surrogate-based Constrained Global Optimization using Space Reduction) including a multi-start constrained optimization on Kriging models of the problem functions to select promising solutions. In the SCGOSR algorithm, two different penalty functions are used to construct two surrogates of Subspaces 1 and 2 to speed up the search. The estimated MSE of Kriging is selected as a merit function to explore the unknown design space and warrant the balance between the exploitation and exploration.

3. Background theories

Although background theories of SBGO using the EGO process exist in previous publications [9,10,13,35], this section addresses some relevant concepts and relationships needed to clarify where and how KASRA originates.

3.1. EGO algorithm

The EGO algorithm is a sequential planning strategy that uses a Gaussian process-based surrogate model, e.g. Kriging (see Section 3.2), and a variance-based merit function or criterion (see Sections 3.3–3.5), to generate a candidate solution to the global optimum over current observed data. The fundamental elements of the EGO algorithms are shown in Algorithm 1.

3.2. Kriging surrogate model

The idea of Kriging was first proposed in the field of geostatistics by Krige [36] and developed by Matheron [37]. Its popularity in the computer-based engineering design is due to its use by Sacks et al. [38]. Two somewhat different approaches can be employed to derive key formulas of kriging for constructing a surrogate model of the deterministic computer code: (1) standard derivation based on a modification to linear regression that assumes the regression terms are a simple constant and error terms are a stochastic process related to experimental data [9,38]; (2) intuitive derivation begins with a slightly abstract concept [13–15,35]. This overview of Kriging is based on the second approach.

Suppose a smooth and continuous function, $y(\mathbf{x})$, of d variables which falls within black-box has been evaluated at n points $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}^{\mathrm{T}}$, with known outputs, $\mathbf{y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(n)}\}^{\mathrm{T}}$. These experiments are going to interpolate for building a prediction function, $\hat{y}(\mathbf{x})$, as a surrogate. If before sampling at any point $\mathbf{x}$, the known output is assumed to be the outcomes of a stochastic process, then there is uncertainty about the associated function values. To treat this uncertainty, the deterministic function value at any point $\mathbf{x}$ is regarded as a realization of a random variable $Y(\mathbf{x})$ that is normally distributed with mean μ and variance σ^2. Taking these in mind, the modeling of predictor, $\hat{y}(\mathbf{x})$, begins with this obvious saying that the function values $y(\mathbf{x}^{(i)})$ and $y(\mathbf{x}^{(j)})$, are close if $\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|$ is small, where $\|.\|$ is the Euclidean distance. In the stochastic process approach, correlation is used to statistically demonstrate this dependence for the points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ and corresponding random variable vectors $Y(\mathbf{x}^{(i)})$ and $Y(\mathbf{x}^{(j)})$, as defined in Eq. (1); the smaller the distance, the higher the correlation.

$$\mathrm{Corr}\left(Y\left(\mathbf{x}^{(i)}\right), Y\left(\mathbf{x}^{(j)}\right)\right) = \exp\left(-\sum_{l=1}^{d} \theta_l \left|x_l^{(i)} - x_l^{(j)}\right|^{p_l}\right). \tag{1}$$

As seen in Eq. (1), the correlation uses a special weighted distance formula instead of simply the Euclidean distance. The Euclidean distance weights all the variables equally, while the correlation considers each variable independently and the hyperparameters $\theta_l > 0$ and $1 \le p_l \le 2$ in Eq. (1), provide more degrees of freedom for the predictor to imitate the behavior of multimodal functions. The parameter θ_l or 'width' parameter is interpreted as a measure to show the depth of influence or "activity" of the variable x_l of a sample point in the function (more active as the θ_l's increase), the fact that is used in this work to derive a metric for adjusting the size of each variable interval. The exponent p_l is considered as the 'smoothness' of the function in the lth coordinate direction (smoother as the p_l's increase) [9,10,35]. The correlation matrix is constructed of all the observed data by Eq. (2):

$$\mathbf{R} = \begin{bmatrix} \mathrm{Corr}\left[Y\left(\mathbf{x}^{(1)}\right), Y\left(\mathbf{x}^{(1)}\right)\right] & \cdots & \mathrm{Corr}\left[Y\left(\mathbf{x}^{(1)}\right), Y\left(\mathbf{x}^{(n)}\right)\right] \\ \vdots & \ddots & \vdots \\ \mathrm{Corr}\left[Y\left(\mathbf{x}^{(n)}\right), Y\left(\mathbf{x}^{(1)}\right)\right] & \cdots & \mathrm{Corr}\left[Y\left(\mathbf{x}^{(n)}\right), Y\left(\mathbf{x}^{(n)}\right)\right] \end{bmatrix} \tag{2}$$

The function's behavior can be captured by estimating the parameters μ and σ as well as the hyperparameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_{\mathrm{d}}\}^{\mathrm{T}}$ and $\mathbf{p} = \{p_1, p_2, \ldots, p_{\mathrm{d}}\}^{\mathrm{T}}$ using maximizing the likelihood of the observed data. In practice, the natural logarithm of the likelihood function is maximized to give Eq. (3):

$$\ln(L) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\left(\sigma^2\right) - \frac{1}{2}\ln|\mathbf{R}| - \frac{(\boldsymbol{y} - 1\mu)^{\mathrm{T}}\mathbf{R}^{-1}(\boldsymbol{y} - 1\mu)}{2\sigma^2} \tag{3}$$

where $\mathbf{1}$ is an (n × 1)-vector of ones. Maximum likelihood estimates for parameters μ, σ^2 can be obtained by taking the derivatives of Eq. (3) and equating them to zero, yielding:

$$\hat{\mu} = \frac{1^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}}{1^{\mathrm{T}}\mathbf{R}^{-1}1} \tag{4}$$

$$\hat{\sigma}^2 = \frac{(\boldsymbol{y} - 1\mu)^{\mathrm{T}}\mathbf{R}^{-1}(\boldsymbol{y} - 1\mu)}{n} \tag{5}$$

Algorithm 1 The pseudo codes of the Standard constrained EGO (single-objective)

Input: (1) $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{1+k}(\mathbf{x})]^{\mathrm{T}}$: a vector involves 1 objective and k constraints deterministic black-box functions whose function values may be the output of a complex computer simulation; (2) $D = [\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$, $d \geq 1$: original search space which is defined in a closed hyper-box-shaped; (3) $m = 0$: a counter of expensive function evaluations; (4) $t = 0$: a counter of updating infill points.

Output: (1) $(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))$: best solution as a global optimum; (2) m; (3) t.

Begin
- Choose an initial set of points as matrix $\mathbf{X}_D = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}]^{\mathrm{T}} \subset D$, by a space-filling sampling approach.
- Call vector $\mathbf{y}(\mathbf{x})$ to evaluate responses of $\mathbf{X}_D$ as matrix $\mathbf{Y}_D = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(n)}]^{\mathrm{T}}$, and increment counter $m = n$ (it is assumed that the values of the objective and constraint functions calculated in one simulation).
- Construct the initial Gaussian process-based surrogate models of objective and constraints $\hat{y}_j(\mathbf{x})$, e.g. Kriging, using all available data points $\{(\mathbf{x}^{(i)}, \mathbf{y}(\mathbf{x}^{(i)})) : i = 1, \ldots, n\}$.
- **while** the stopping criterion is not met **do**
  - Select the best solution $\mathbf{x}^*$.
  - Locate the new point $\mathbf{x}^{(n+t)} \subset D$, with the help of the variance-based constrained infill criteria, reset $t = t + 1$.
  - Call $\mathbf{y}(\mathbf{x})$ to obtain the true responses of the new sampled point as $\mathbf{y}^{(n+t)}$, reset $m = m + 1$.
  - Update observed data: $\mathbf{X}_D = \mathbf{X}_D \cup \mathbf{x}^{(n+t)}$, $\mathbf{Y}_D = \mathbf{Y}_D \cup \mathbf{y}^{(n+t)}$.
  - Refit the surrogate model.
- **end**

End

Substituting Eqs. (4) and (5) into Eq. (3) gives the concentrated (ln)-likelihood function in Eq. (6):

$$\ln(L) \approx -\frac{n}{2}\ln\left(\hat{\sigma}^2\right) - \frac{1}{2}\ln|\mathbf{R}| \tag{6}$$

The value of this function depends only on $\mathbf{R}$, and hence, on the unknown hyperparameters $\boldsymbol{\theta}$ and $\mathbf{p}$. In practice, optimal values for $\boldsymbol{\theta}$ and $\mathbf{p}$ are tuned by optimizing Eq. (6) using a direct numerical global optimization that usually is a time-consuming process. In this work, we use a variable tuning degree strategy of genetic algorithm, which will be discussed later.

Now, the Kriging predictor, $\hat{y}(\mathbf{x})$, can be made by adding a new pseudo-point ($\mathbf{x}^*$, y^*) to the data as the (n + 1)th observation to construct the 'augmented' likelihood function and maximizing this augmented likelihood function with respect to y^*, to give the standard formula for the Kriging predictor in Eq. (7):

$$\hat{y}(\mathbf{x}) = \hat{\mu} + \mathbf{r}^T\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{1}\hat{\mu}\right) \tag{7}$$

where $\mathbf{r}$ denotes the vector of correlation between the observed data and the prediction. Eq. (8) permits to model uncertainty at $y(\mathbf{x})$ by calculating an estimated mean squared error (MSE) [38]:

$$\hat{s}^2(\mathbf{x}) = \sigma^2\left[1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r}\right] + \sigma^2\left[\frac{\left(1 - \mathbf{1}^T\mathbf{R}^{-1}\mathbf{r}\right)^2}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{r}}\right] \tag{8}$$

The first term represents the uncertainty due to the correlation of the point in question, $\mathbf{x}$, with the observed data. The second term reflects the uncertainty of the estimated parameter μ. MSE tends to σ^2 if point $\mathbf{x}$ goes far away from the observed data (where $\mathbf{r} \to \mathbf{0}$) also MSE would be close to zero if point $\mathbf{x}$ sufficiently approaches each observed data.

As mentioned in the introduction section, when performing an adaptive sampling strategy, the next candidate point(s) in promising areas is selected based on some infill sampling criteria. Both prediction function in Eq. (7) and MSE in Eq. (8) provide the necessary facilities for a class of infill sampling criteria to balance the local exploitation and global exploration for positioning update infill point(s). This class is considered in the following.

3.3. Probability of improvement criterion

When the function's value uncertainty at $\mathbf{x}$ is described as a realization of normally distributed random variable $Y(\mathbf{x})$ with the estimated mean and variance are suggested in Eqs. (7) and (8), then there is some probability that the function's value at $\mathbf{x}$ will be better than current best function value $y_{\min}$. Formally, the improvement at the point $\mathbf{x}$ upon $y_{\min}$ is $I = \max(y_{\min} - Y(\mathbf{x}), 0)$, and the likelihood of achieving this improvement is calculated by integration the area enclosed by the normal density function where below $y_{\min}$ as Eq. (9); (see [9,10,13,35] for more graphically details):

$$P[I(\mathbf{x})] = \frac{1}{\hat{s}(\mathbf{x})\sqrt{2\pi}}\int_{-\infty}^{0} \mathrm{e}^{-\left[I-\hat{y}(\mathbf{x})\right]^2/\left(2\hat{s}^2(\mathbf{x})\right)}\mathrm{d}I \tag{9}$$

The $P[I(\mathbf{x})]$ can be calculated using the error function as Eq. (10):

$$P[I(\mathbf{x})] = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{y_{\min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})\sqrt{2}}\right)\right] \tag{10}$$

3.4. Expected improvement criterion and adding weighting factors

The $P[I(\mathbf{x})]$, does not show the magnitude of improvement that is expected; that is why the concept of the expected improvement, $\mathrm{E}[I(\mathbf{x})] \equiv \mathrm{E}(\max(y_{\min} - Y(\mathbf{x}), 0))$, employed to get the expected value [9]. If the possible improvements are weighted by the associated density value, the expected improvement is provided as Eq. (11):

$$\mathrm{E}[I(\mathbf{x})] = \begin{cases} \left(y_{\min} - \hat{y}(\mathbf{x})\right)\Phi\left(\frac{y_{\min}-\hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x})\,\phi\left(\frac{y_{\min}-\hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) & \text{if } \hat{s}(\mathbf{x}) > 0 \\ 0 & \text{if } \hat{s}(\mathbf{x}) = 0 \end{cases} \tag{11}$$

where $\Phi(.)$ and $\phi(.)$ are the cumulative distribution function and probability the density function respectively. Eq. (11) is evaluated using the error function in Eq. (12):

$$\mathrm{E}[I(\mathbf{x})] = \left(y_{\min} - \hat{y}(\mathbf{x})\right)\left[\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{y_{\min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})\sqrt{2}}\right)\right] + \hat{s}(\mathbf{x})\frac{1}{\sqrt{2\pi}}\exp\left[\frac{-\left(y_{\min} - \hat{y}(\mathbf{x})\right)^2}{2\hat{s}^2(\mathbf{x})}\right] \tag{12}$$

The $\mathrm{E}[I(\mathbf{x})]$ will tend to be large at $\mathbf{x}$ with $\hat{y}(\mathbf{x})$ smaller than $y_{\min}$ and/or at $\mathbf{x}$ with much $\hat{s}(\mathbf{x})$ associated with the prediction. Thus, The $\mathrm{E}[I(\mathbf{x})]$, achieves a balance between the local and global scope

of the search into a single criterion. To make a controllable search, Sobester et al. [39], proposed a 'weighted expected improvement' criterion, which offers the possibility of fully covering the continuum between local exploitation and global exploration as in Eq. (13):

$$\mathrm{WE}\left[I\left(\mathbf{x}\right)\right]=\begin{cases} w\left(y_{\min}-\hat{y}\left(\mathbf{x}\right)\right)\Phi\left(\frac{y_{\min}-\hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) & \\ \quad +(1-w)\hat{s}\left(\mathbf{x}\right)\phi\left(\frac{y_{\min}-\hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) & \text{if}\quad \hat{s}\left(\mathbf{x}\right)>0 \\ 0 & \text{if}\quad \hat{s}\left(\mathbf{x}\right)=0 \end{cases} \tag{13}$$

where the weighting factor $w \in [0, 1]$. Clearly, $w = 1$ will tend the search to the current best basin of attraction (local), while $w = 0$ will spread the scope of the search on the entire range (global). Naturally, $w = 0.5$ will yield half of E[$I(\mathbf{x})$]. The selection of weighting is extremely difficult for a particular application. Thus in this work, a dynamically setting is employed to cover a reasonable range that will be discussed later.

3.5. Constrained expected improvement criterion

The optimization problems in engineering design are almost always along with a number of constraints. If there is an inequality constraint function as $g(\mathbf{x}) > g_{\text{limit}}$, where g_{limit} is constraint limit value, then a new infill point will be feasible when its constraint function value is greater than g_{limit}. Now consider a constraint function, also modeled by a Gaussian process, based on sample data in exactly the same way as for objective function. Rather than calculating $P[I(\mathbf{x})]$ in Eq. (9), this model can be used to calculate the probability of the prediction being greater than the constraint limit, i.e. the probability that the constraint is met, $P[F(\mathbf{x})]$ [13]. So, the 'probability of feasibility' for each constraint, g, is calculated in Eq. (14) following the same logic as for an improvement, only now instead of the current best design value, the constraint limit value is used.

$$P\left[F\left(\mathbf{x}\right)\right]=P\left[g\left(\mathbf{x}\right)>g_{limit}\right]=\frac{1}{\hat{s}\left(\mathbf{x}\right)\sqrt{2\pi}}\int_{0}^{\infty}\mathrm{e}^{-\left[(G(\mathbf{x})-g_{\text{limit}})-\hat{g}(\mathbf{x})\right]^{2}/\left(2\hat{s}^{2}(\mathbf{x})\right)}\mathrm{d}G \tag{14}$$

where $G(\mathbf{x}) - g_{\text{limit}}$ is the measure of feasibility, $G(\mathbf{x})$ is a random variable, and $\widehat{s}^2(\mathbf{x})$ is the variance of the Kriging model of the constraint [13].

Provided that the constraints and objective are all independent models, Schonlau [40], suggested a criterion referred to constrained expected improvement [10], to handle constraints in SBGO by simply multiplying the expected improvement of the objective function and the probability of feasibility as Eq. (15):

$$\mathrm{CE}\left[I\left(\mathbf{x}\right)\right]=\mathrm{E}\left[I\left(\mathbf{x}\right)\cap F\left(\mathbf{x}\right)\right]=\mathrm{E}\left[I\left(\mathbf{x}\right)\right]P\left[F\left(\mathbf{x}\right)\right] \tag{15}$$

Note that when there is more than one constraint, the sum of $P[F(\mathbf{x})]$ of each constrain must be calculated. In other words, if all inequality constraints are transformed to standard form ("$\leq$ type" inequalities) with $g_{\text{limit}} = 0$ as this work, then $P[all\ g \leq 0]$ mast be satisfied. Eq. (15) is usually transcribed into the $\{-\log_{10}\mathrm{CE}[I(\mathbf{x})]\}$ and minimized to update a new infill point, see Section 5.4.2 of Ref. [13] for more details.

At the end of this section, it is noticeable that the focus of this article is on solving inequality constraints optimization problems as Eq. (16):

$$\begin{aligned} &\min y\left(\mathbf{x}\right)\\ &\text{s.t.}\quad \mathbf{x}\in\mathbb{R}^{d},\ \mathbf{a}\leq\mathbf{x}\leq\mathbf{b}\\ &\quad g_{i}\left(\mathbf{x}\right)\leq 0, i=1,2,\ldots,m \end{aligned} \tag{16}$$

where $[\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$ and y, g_1, …, g_m are deterministic, computationally expensive black-box objective and constraint functions respectively.

4. The proposed algorithm—KASRA

The convergence of the EGO algorithm using the EI criterion has been established by Vazquez and Bect [41], for P-almost all continuous functions, where P is the (prior) probability distribution of the Gaussian process. However, in practice, using the EGO algorithm would hardly be justified when the number of sample points is increasing, particularly for HEB problems handling many nonlinear constraints. KASRA is trying to enhance the rate of convergence using of space reduction-like approach to broaden the applicability of the EGO method. Fig. 1 illustrates the outline of the proposed optimization framework as a flowchart.

A more detailed description of the procedure listed in Algorithm 2 and presented in the following major steps. During the implementation of the main Algorithm 2, five sub-algorithms 2(a) to 2(e) are invoked; the details of each sub-algorithm separately are summarized in Sections 4.1–4.5.

Step 1. (lines 1–3): The first phase begins with initial DACE sampling using space-filling LHD and their corresponding true objective and constraints function evaluations over the original design domain D.

Step 2. (lines 4–12): The current best feasible point or in its absence, the infeasible one in the current design database is selected as $\mathbf{x}^*$. The feasible $\mathbf{x}^*$ is the one that has the lowest objective function value, while the best infeasible $\mathbf{x}^*$ comprises the minimum sum of constraint violations.

Step 3. (line 13): The size of intervals adaptively is modified using Algorithm 2(a) to capture the promising design sub-space denoted by $D_c \subset D$, around the current $\mathbf{x}^*$. As soon as a new $\mathbf{x}^*$ is found the intervals are shrunken around it, as a sub-space denoted by D_{sh}, otherwise promising region gradually is expanded as D_{ex} to provide the possibility for searching other regions of D that most likely contain the global optimum; note that at the first time of invoking Algorithm 2(a), certainly D_c will be D_{sh}.

Step 4. (lines 14–18): For updating a new infill point named as $\mathbf{x}^{\text{infill}}$, the sample points in D_{sh} are needed to be clustered. Once D_c is expanded, all sample points within D_c are not selected, but only those which are surrounded by boundaries of D_{sh}, as well as new $\mathbf{x}^{\text{infill}}$, are clustered as $\mathbf{X}_s$; this is to avoid the time consumed to tune the hyperparameters of the surrogates.

Step 5. (lines 19–23): If current $\mathbf{x}^*$ is infeasible then Algorithm 2(b) is activated to search D_c for updating a new $\mathbf{x}^{\text{infill}}$; the Algorithm 2(c) instead of Algorithm 2(b) is called if $\mathbf{x}^*$ is feasible.

Step 6. (lines 24–25): All the sample points $\mathbf{X}_D$, and corresponding true responses $\mathbf{Y}_D$, are stored in the design database for later use; the responses include both objective and constraint functions.

Step 7. (lines 26–29): The process returns to step 1 if D_c is not equal D and the total function evaluation budget for phase 1 is still not completed, otherwise, phase 2 is performed. For benchmark testing problems where the global optimum is known, the relative error between y^* and y^{optimal} in Eq. (17) is typically used as the stopping and jumping criterion from phase 1 to phase 2. In this study, the permissible relative error is 1% and 0.001% according to Eq. (17). For real-world problems with unknown global optimum, the small relative error between

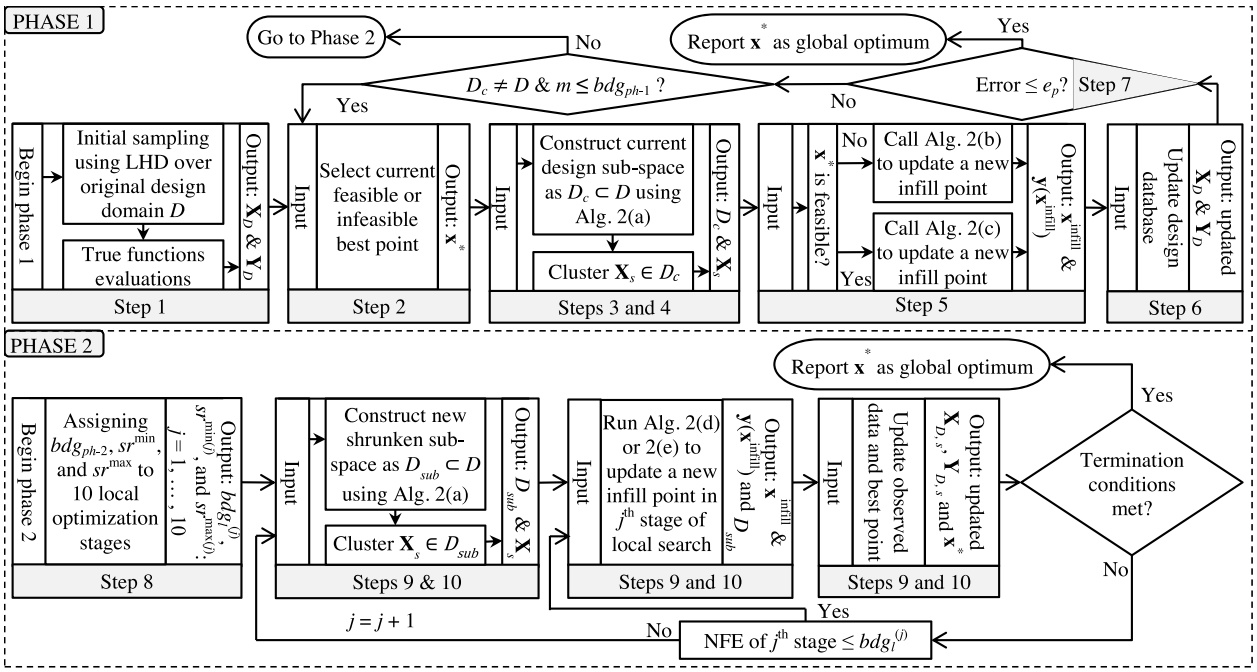


Fig. 1. Flowchart of KASRA optimization process.

two successive y^* can be considered.

$$\begin{cases} \left|\left(y^* - y^{\text{optimal}}\right)/y^{\text{optimal}}\right| \times 100 \leq 1\%, & \text{if} \quad y^{\text{optimal}} \neq 0 \\ y^* \times 100 \leq 0.001\%, & \text{if} \quad y^{\text{optimal}} = 0 \end{cases} \tag{17}$$

Step 8. (lines 30–31): If current $\mathbf{x}^*$ does not adequately get close to global optimum, the second phase will begin with assigning the total function evaluation budget of phase 2 to the local search including ten successive stages, each with the distinct size of sub-region D_{sub}. The minimum and maximum space reductions as a percent of D, named as $sr^{\min}$ and $sr^{\max}$, are chosen for each sub-region so that D_{sub} only captures a small vicinity of $\mathbf{x}^*$ to promote fitting quality of local meta-models. The reason for dividing phase 2 into several stages is that, firstly, the accumulation of infill points and, consequently, the waste of time in the optimization process does not occur and, secondly, each time more focus is made on the current $\mathbf{x}^*$.

Step 9. (line 32–44): The first stage of local optimization gets started by constructing the new sub-regions D_{sub} in the same way as D_{sh} and selecting the points that belong to it. Then, Algorithm 2(d) runs to update both the new infill point and the current sub-region. By updating the current x* and other parameters, KASRA process is continued. Upon completion of this stage budget and the failure of the terminating condition, the process will resume in the next stage.

Step 10. (line 45–56): In this step, nine local optimization stages each one involves two deterministic global and local optimization routines respectively and continuously are carried out on KBSMs of objective and constraint functions according to Algorithm 2(e) to update a new infill point. The process in each stage may end if the terminating condition is satisfied. Note that the new D_{sub} is constructed at the beginning of each local optimization stage.

4.1. Algorithm 2(a)

The Algorithm 2(a) is proposed to adaptively resize (shrink or expand) current D_c, given to the information acquired during the optimization trend. Here the basic idea to shrink design space is that not all dimensions of the domain need to contract uniformly. Instead, changes are made based upon reflects of the objective function behavior in each dimension. This means, to obtain a much local fitting quality of the function prediction, it seems better to focus on more significant variables. As mentioned in Section 3.2, the hyperparameters $\boldsymbol{\theta}$ indicates the order of importance or activity of variables. Therefore, the largest and smallest interval reduction of design variables can be assigned to the more and less active variables respectively, the $\theta_l^{\max}$ and $\theta_l^{\min}$ in the Eq. (1), and others are changed due to the magnitude of the corresponding θ_l. To avoid missing regions without scanning and being trapped at local optima, reduced space gradually is expanded so that the expanded region matches to the original space at the end of phase 1, if the total function evaluation budget for phase 1 is still not completed. Algorithm 2(a) consists of the following steps.

Step 1. (lines 1–11): The Eq. (6) is maximized to tune the $\boldsymbol{\theta}$ values of the surrogate objective function. For saving time and since the promising region will be concentrated on current $\mathbf{x}^*$, a few numbers of points, n_{close}, in the neighborhood of $\mathbf{x}^*$ are employed for tuning. Since a global search method for tuning $\boldsymbol{\theta}$ values, such as a genetic algorithm in this paper, may not find the correct values for $\boldsymbol{\theta}$, especially if dimensions of the problem are large, the tuning process is repeated n_{it} times to increase the probability of finding the correct values of $\boldsymbol{\theta}$. Note that these comments will be done if $sr^{\min}$ and $sr^{\max}$ are not equal.

Step 2. (lines 12–22): By selecting $sr^{\min}$ and $sr^{\max}$, an array of space reduction values, denoted by $\mathbf{sr}$, is calculated to construct shrunken design space D_{sh}. Generally, although the smaller $sr^{\min}$ and $sr^{\max}$ will create more local and accurate meta-models of the functions, it will lead to more number of updated infill points for convergence. On the

Algorithm 2 The pseudo codes of KASRA (single-objective)

Input: (1) $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \dots, y_{1+k}(\mathbf{x})]^T$: a vector involves 1 objective and k constraints deterministic expensive-to-evaluate black-box functions whose function values may be the output of a complex computer simulation; (2) $D = [\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$, $d \geq 1$: original search space which is defined in a closed hyper-box-shaped; (3) $m = 0$: a counter of expensive function evaluations; (4) $t = 0$: a counter of updating infill points; (5) $bdg_{ph\text{-}1}$: total function evaluation budget for phase 1; (6) bdg_{lim}: a limited function evaluation budget to perform phase 1 faster; (7) $bdg_{ph\text{-}2}$: total function evaluation budget for phase 2; (8) $bdg_l^{(j)}$: a vector of 10 budgets for the successive stages of the phase 2; (9) $NFE_l = 0$: number of function evaluations of phase 2; (10) $fl = 0$: a flag for running phase 2; (11) $D_c = \emptyset$: the current design sub-space; (12) sw: a flag to switch from Alg. 2(b) to Alg. 2(c); (13) e_p: permissible error.
Output: (1) $(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))$: best solution as a global optimum; (2) m; (3) t.

1. **Begin**
2. Choose an initial set of points as matrix $\mathbf{X}_D = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}]^T \subset D$, by a space-filling DACE sampling approach.
3. Call $\mathbf{y}(\mathbf{x})$ to evaluate responses of $\mathbf{X}_D$ as matrix $\mathbf{Y}_D = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(n)}]^T$, and increment counter $m = n$. (it is assumed that the values of the objective and constraint functions calculated in one simulation).
4. **while** $D_c \neq D$ & $m \leq bdg_{ph\text{-}1}$ **do phase 1**
5. **if** $\{\mathbf{x}^{(i)} : i = 1, \dots, n + t\}$ are infeasible **then**
6. $\{v^{(i)}\} \leftarrow \{\sum \max(0, y_j^{(i)}) : j = 2, \dots, k + 1\}$.
7. $\mathbf{x}^* = \mathbf{x}^{(m)} \leftarrow v^{(m)} = \min\{v^{(i)}\}$.
8. $sw = 0$;
9. **elseif** there is at least one feasible sample point **then**
10. $\mathbf{x}^* = \mathbf{x}^{(m)} \leftarrow y_1^{(m)} = \min\{y_1^{(i)} : i = 1, \dots, n + t\}$.
11. $sw = 1$;
12. **end**
13. Run the Alg. 2(a) to resize (shrinking or expanding) design domain D centered at $\mathbf{x}^*$ as $D_c \subset D$; outputs of Alg. 2(a): D_c, D_{sh}, and D_{ex}, (note: at first time of running Alg. 2(a), region D_c will be equal to D_{sh})
14. **if** $D_c = D_{sh}$ **then**
15. Select $\mathbf{X}_s = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(s)}]^T \in D_{sh}$ with their responses $\mathbf{Y}_s = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(s)}]^T$;
16. **elseif** $D_c = D_{ex}$ **then**
17. Select $\mathbf{X}_s = \mathbf{X}_s \cup \mathbf{x}^{\text{infill}}$ with their responses $\mathbf{Y}_s = \mathbf{Y}_s \cup \mathbf{y}(\mathbf{x}^{\text{infill}})$;
18. **end**
19. **if** $sw = 0$ **then**
20. Implement Alg. 2(b) on D_c to update sw and new infill point $\mathbf{x}^{\text{infill}} \subset D_c$, with function responses $\mathbf{y}(\mathbf{x}^{\text{infill}})$; reset $t = t + 1$ and $m = m + 1$.
21. **elseif** $sw = 1$ **then**
22. Implement Alg. 2(c) on D_c to update new infill point $\mathbf{x}^{\text{infill}} \subset D_c$, with function responses $\mathbf{y}(\mathbf{x}^{\text{infill}})$; reset $t = t + 1$ and $m = m + 1$.
23. **end**
24. $\mathbf{x}^{(n+t)} = \mathbf{x}^{\text{infill}}$ and $\mathbf{y}^{(n+t)} = \mathbf{y}(\mathbf{x}^{\text{infill}})$.
25. Update observed data: $\mathbf{X}_D = \mathbf{X}_D \cup \mathbf{x}^{(n+t)}$; $\mathbf{Y}_D = \mathbf{Y}_D \cup \mathbf{y}^{(n+t)}$.
26. **if** error $\leq e_p$ **then**
27. $fl = 1$ and break 'while' loop.
28. **end**
29. **end**
30. **if** $fl = 0$ **do phase 2**
31. Assigning $bdg_l^{(j)}$, $sr^{\min}$, and $sr^{\max}$ to the distinct stages of the phase 2: $\{bdg_l^{(j)}, sr^{\min(j)}$, and $sr^{\max(j)}: j = 1, \dots, 10\}$. Note that, $\{\sum (bdg_l^{(j)}) = bdg_{ph\text{-}2}\}$ and recommended $sr^{\min(j)}$ and $sr^{\max(j)}$ are less than and equal to 5 and 10 in this work respectively.
32. Run comments the same as those in Alg. 2(a) from lines 3–24 with $n_{\text{close}} = 20$ and $n_{it} = 10$ with specified $sr^{\min}$ and $sr^{\max}$ of the first stage for constructing a new sub-region $D_{sub} \subset D$.
33. Select $\mathbf{X}_s = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(s)}]^T \in D_{sub}$ with their responses $\mathbf{Y}_s = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(s)}]^T$;
34. **for** $i \leftarrow 1$ to $bdg_l^{(1)}$.
35. Run Alg. 2(d) to update both new infill point $(\mathbf{x}^{\text{infill}}, \mathbf{y}(\mathbf{x}^{\text{infill}})) \subset D_{sub}$, and sub-region D_{sub}.
36. Update $\mathbf{X}_s$ and $\mathbf{Y}_s$: $\mathbf{X}_s = \mathbf{X}_s \cup \mathbf{x}^{\text{infill}}$; $\mathbf{Y}_s = \mathbf{Y}_s \cup \mathbf{y}^{\text{infill}}$.
37. Increment $t = t + 1$, $m = m + 1$, and $NFE_l = NFE_l + 1$.
38. $\mathbf{x}^{(n+t)} = \mathbf{x}^{\text{infill}}$ and $\mathbf{y}^{(n+t)} = \mathbf{y}(\mathbf{x}^{\text{infill}})$.
39. Update observed data: $\mathbf{X}_D = \mathbf{X}_D \cup \mathbf{x}^{(n+t)}$; $\mathbf{Y}_D = \mathbf{Y}_D \cup \mathbf{y}^{(n+t)}$.
40. Update current $\mathbf{x}^*$ using comments the same as those in lines 5–12.
41. **if** error $\leq e_p$ **then**
42. $fl = 1$ and break 'for' loop.
43. **end**
44. **end**
45. **for** $j \leftarrow 2$ to 10.
46. **if** $fl = 0$ **then**
47. Run comments the same as those in Alg. 2(a) from lines 3–24 with $n_{\text{close}} = 20$ and $n_{it} = 10$ with specified $sr^{\min}$ and $sr^{\max}$ of the j^{th} stage for constructing a new sub-region $D_{sub} \subset D$.
48. Select $\mathbf{X}_s = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(s)}]^T \in D_{sub}$ with their responses $\mathbf{Y}_s = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(s)}]^T$;
49. **for** $i \leftarrow 1$ to $bdg_l^{(j)}$.
50. Run Alg. 2(e) to update both new infill point $(\mathbf{x}^{\text{infill}}, \mathbf{y}(\mathbf{x}^{\text{infill}})) \subset D_{sub}$, and sub-region D_{sub}.
51. Run comments the same as those in lines 36–43.
52. **end**
53. **end**
54. **end**
55. **end**
56. **End**

other hand, the bigger $sr^{\min}$ and $sr^{\max}$ can compromise the fitting quality of the meta-models and may miss some promising solutions especially when the problem is multimodal. Hence, the recommended ranges for $sr^{\min}$ and $sr^{\max}$ are proposed [5%, 25%] and [25%, 50%] respectively. In this work, $sr^{\min}$ and $sr^{\max}$ are defined as 10% and 35% respectively.

Step 3. (lines 23–43): When the current infill point, $\mathbf{x}^{\text{infill}}$, is not a new $\mathbf{x}^*$, subsequently D_c begins to be expanded as D_{ex}. Here, the difference between the size of D_{sh} and D gradually is compensated under three different conditions. The first condition is when a new feasible or infeasible $\mathbf{x}^*$ is still not reached after initial sampling. In this situation, naturally, the entire infill sampling budget – ($bdg_{ph\text{-}1} - n$), where $bdg_{ph\text{-}1}$ is the total sampling budget for phase 1 and n is the number of initial points – is used to search for a new $\mathbf{x}^*$, while D_c becomes larger according to remaining budget. The second condition is similar to the first one with the difference that a new infeasible best point(s) has been found after initial sampling; here too, the remaining infill sampling budget – ($bdg_{ph\text{-}1} - n_{\text{best}}$), where n_{best} is sequence number of $\mathbf{x}^*$ – should be used for expanding D_c until a feasible $\mathbf{x}^*$ likely being trapped. In the third condition, at least a new feasible $\mathbf{x}^*$ is found after initial sampling and a limited infill budget, named as bdg_{lim}, is used for faster expanding D_c in phase 1.

4.2. Algorithm 2(b)

After constructing D_c and clustering points $\mathbf{X}_s$ with responses $\mathbf{Y}_s$ in the lines 13–18 of Algorithm 2, it is time to implement Algorithm 2(b) according to below steps if current $\mathbf{x}^*$ is infeasible.

Step 1. (lines 1–13): In a situation where there is still no feasible point, the priority of the infill sampling process is to control the searching direction towards the feasible zone. For this purpose, Eq. (6) using of $\mathbf{X}_s$ and $\mathbf{Y}_s$ is maximized via a genetic algorithm to estimate hyperparameters θ and $\mathbf{p}$ for each objective and constraint function. The generation, g, and population size, p, are variable in accordance to the number of clustering points $\mathbf{X}_s$ in this stage of the algorithm; the cause is to moderate the time will be spent on computation.

Step 2. (lines 14–15): The KBSMs of the problem functions are constructed using Eq. (7) and continue with estimating the MSE via Eq. (8).

Step 3. (lines 16–21): The sum of probability of feasibility for each constraint in Eq. (14) is maximized using a genetic algorithm with g and p equal to 50 and 100 respectively to locate a new infill point as $\mathbf{x}^{\text{infill}}$. The functions of the problem or computationally intensive simulations as array $\mathbf{y}(\mathbf{x})$, are called to evaluate the true responses at $\mathbf{x}^{\text{infill}}$. Then they are checked for the feasibility of $\mathbf{x}^{\text{infill}}$. A switching flag, named as sw, will change if $\mathbf{x}^{\text{infill}}$ does not violate constraints. In this situation, $sw = 1$ prevents Algorithm 2(b) from running again and instead activates Algorithm 2(c) in the following of procedure.

4.3. Algorithm 2(c)

Algorithm 2(c) instead of Algorithm 2(b) will be triggered whenever current $\mathbf{x}^*$ is feasible as the below steps; the main difference between two algorithms is the choice of infill criteria.

Step 1. (lines 1–15): This step is the same as steps 1 and 2 of Algorithm 2(b).

Step 2. (lines 16–18): The weighted constrained expected improvement from the combination of Eqs. (13) and (15) is employed as an infill criterion. The weighting factor, denoted by w, automatically is adjusted according to the volumetric ratio of D_c, to the D_{sh}, and D; the volumes of the hyper-box-shaped regions of D, D_{sh}, and D_c are considered as V_D, V_{sh}, and V_c. After allocating $w^{\max}$ and $w^{\min}$ to the smallest and largest volumes (V_{sh} and V_D), the current w can be calculated given to the current volume V_c according to the lines 17 and 18. The values of $w^{\max}$ and $w^{\min}$ are chosen to be half and zero, respectively. Thus, the searching process makes a balanced tradeoff between local exploitation and global exploration in the promising region when V_c is equal to V_{sh}. Subsequently, the contribution of the exploration of the unknown regions will be enhanced as the process progresses and D_c gets closer to D.

Step 3. (lines 19–21): After calculating current w, the weighted CE[$I(\mathbf{x})$] is maximized using a genetic algorithm with g and p equal to 50 and 100 respectively to locate a new $\mathbf{x}^{\text{infill}}$. Then array $\mathbf{y}(\mathbf{x})$ is called to evaluate the true responses at $\mathbf{x}^{\text{infill}}$.

4.4. Algorithm 2(d)

If phase 1 does not succeed to find a solution adequately close to the global optimum, the optimization procedure enters phase 2. The main goal of executing the phase 2 is the improvement of local accuracy of current $\mathbf{x}^*$; therefore a local sub-space, D_{sub}, centered at the current $\mathbf{x}^*$ needs to be made in the same way presented at the lines 3–24 of Algorithm 2(a). The steps of the Algorithm 2(d) are presented in the following.

Step 1. (lines 1–4): After determining D_{sub} and choosing sample points belonging to D_{sub}, the Eq. (6) is maximized using a genetic algorithm with g and p equal 50 and 60 respectively to estimate hyperparameters $\boldsymbol{\theta}$ and $\mathbf{p}$ for each objective and constraint surrogate function. Then the KBSMs of problem functions are fitted using Eq. (7) and continue with estimating the MSE via Eq. (8).

Step 2. (lines 5–6): The constrained expected improvement in Eqs. (12) and (15) is employed as an infill criterion. The E[$I(\mathbf{x})$] in Eq. (12) tends to balance the exploitation and exploration search. The CE[$I(\mathbf{x})$] in Eq. (15) is maximized using a genetic algorithm with g and p equal to 50 and 100 respectively to locate a new $\mathbf{x}^{\text{infill}}$. Then array $\mathbf{y}(\mathbf{x})$ is called to evaluate the true responses at $\mathbf{x}^{\text{infill}}$.

Step 3. (lines 7–16): The D_{sub} is relatively small, and the variables of current $\mathbf{x}^{\text{infill}}$ may be located near the boundaries of D_{sub}. If this occurs, the interval of each active or close to the active variable will be expanded. The size of expanding is regarded as 50% of the current interval.

4.5. Algorithm 2(e)

In phase 2, stages 2–10, merit function to update infill points will be a meta-model of the objective function restricted to some meta-models of constraints. The same as Algorithm 2(d), $sr^{\min}$ and $sr^{\max}$ are chosen so that D_{sub} only captures a small vicinity of $\mathbf{x}^*$. The steps of the Algorithm 2(e) are listed in the following.

Step 1. (lines 1–3): This step is the same as lines 1–3 of Algorithm 2(d).

Step 2. (lines 4–10): The exponential tunneling and SQP as the deterministic global and local optimizers sequentially perform the KBSM-search starting from current $\mathbf{x}^*$ to mine a new $\mathbf{x}^{\text{infill}}$. Hence, as the optimization continues,

Algorithm 2(a) The pseudo codes to resize design space

Input: (1) $D = [\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$: original search space; (2) $\mathbf{X}_D = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n+t)}]^T \subset D$: a set of all sample points until now with their responses as matrix $\mathbf{Y}_D = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(n+t)}]^T$; (3) t: a counter of updating infill points; (4) $\mathbf{x}^*$: current best point; (5) n_{best}: row number of $\mathbf{x}^*$ in matrix $\mathbf{X}_D$; (6) $n_{\text{close}} = 5$: number of selected points that are closest to $\mathbf{x}^*$ for hyperparameter tuning; (7) $n_{it} = 10$: number of iterations for tuning process; (8) $sr^{\min} = 10$, $sr^{\max} = 35$: percentage of minimum and maximum space reduction; (9) $g = 100 \times d$, $p = 200$: maximum generation and population size for genetic algorithm; (10) $bdg_{ph\text{-}1}$: total function evaluation budget for phase 1; (11) bdg_{lim}: a limited function evaluation budget to perform phase 1 faster.

Output: (1) $D_c \subset D$: the current design sub-space; (2) D_{sh}: the shrunken design sub-space; (3) D_{ex}: the expanded design sub-space.

1. **Begin**
2. **if** $t = 0$ or $\mathbf{x}^{(n+t)} = \mathbf{x}^*$ **then** (reduce design space)
3. **if** $d \geq 2$ & $sr^{\min} \neq sr^{\max}$
4. Find n_{close} points that have the smallest distance from the $\mathbf{x}^*$.
5. **for** $i \leftarrow 1$ to n_{it}.
6. Maximize Eq.6 using genetic algorithm with g and p, to find hyperparameter array $\theta_l : l = [1, \ldots, d]$, in Eq.1.
7. Insert array θ_l in the i^{th} row of hyperparameter matrix $\boldsymbol{\Theta}$.
8. **end**
9. **for** $i \leftarrow 1$ to d.
10. Find the frequency of each element in the i^{th} column of $\boldsymbol{\Theta}$ and average the most repetitive one(s) to find the most accurate hyperparameter array $\boldsymbol{\theta}$.
11. **end**
12. Sort the elements of array $\boldsymbol{\theta}$ in ascending order as $[\boldsymbol{\beta}, \mathbf{I}]$, where $\mathbf{I}$ describe the arrangement of the elements of $\boldsymbol{\theta}$ into $\boldsymbol{\beta}$.
13. Round the elements of array $\boldsymbol{\beta}$ to four decimals.
14. $t_1 \leftarrow sr^{\max}$; $t_d \leftarrow sr^{\min}$.
15. $C \leftarrow (t_d - t_1) / (\beta_d - \beta_1)$.
16. **for** $i \leftarrow 2$ to $d - 1$.
17. $t_i \leftarrow (C \times (\beta_i - \beta_1)) + t_1$.
18. **end**
19. $\mathbf{sr}(\mathbf{I}) = \mathbf{t}$.
20. **else**
21. $\mathbf{sr} = sr^{\min} \times [1^{(1)}, 1^{(2)}, \ldots, 1^{(d)}]$.
22. **end**
23. $\mathbf{SR} \leftarrow (\mathbf{sr} / 100) \times (\mathbf{b} - \mathbf{a}) / 2$.
24. $D_{sh} := [\mathbf{x}^* - \mathbf{SR}, \mathbf{x}^* + \mathbf{SR}] \cap [\mathbf{a}, \mathbf{b}]$, $D_c = D_{sh} = [\mathbf{a}^{sh}, \mathbf{b}^{sh}]$, $D_{ex} = \emptyset$;
25. **else** (expand design space)
26. **if** ($\mathbf{x}^*$ is infeasible or feasible) & $n_{\text{best}} <= n$ **then**
27. $NSD_{\text{total}} \leftarrow (bdg_{ph\text{-}1} - n)$.
28. $\mathbf{Lb}^{ex} \leftarrow \mathbf{a}^{sh} - ((t / NSD_{\text{total}}) \times |\mathbf{a} - \mathbf{a}^{sh}|)$.
29. $\mathbf{Ub}^{ex} \leftarrow \mathbf{b}^{sh} + ((t / NSD_{\text{total}}) \times |\mathbf{b} - \mathbf{b}^{sh}|)$.
30. $D_{ex} := [\mathbf{Lb}^{ex}, \mathbf{Ub}^{ex}] \cap [\mathbf{a}, \mathbf{b}]$, $D_c = D_{ex}$.
31. **elseif** $\mathbf{x}^*$ is infeasible & $n_{\text{best}} > n$ **then**
32. $NSD_{\text{total}} \leftarrow (bdg_{ph\text{-}1} - n_{\text{best}})$.
33. $\mathbf{Lb}^{ex} \leftarrow \mathbf{a}^{sh} - ((((n + t) - n_{\text{best}}) / NSD_{\text{total}}) \times |\mathbf{a} - \mathbf{a}^{sh}|)$.
34. $\mathbf{Ub}^{ex} \leftarrow \mathbf{b}^{sh} + ((((n + t) - n_{\text{best}}) / NSD_{\text{total}}) \times |\mathbf{b} - \mathbf{b}^{sh}|)$.
35. $D_{ex} := [\mathbf{Lb}^{ex}, \mathbf{Ub}^{ex}] \cap [\mathbf{a}, \mathbf{b}]$, $D_c = D_{ex}$.
36. **elseif** $\mathbf{x}^*$ is feasible & $n_{\text{best}} > n$ **then**
37. $NSD_{\text{total}} = bdg_{lim}$.
38. $\mathbf{Lb}^{ex} \leftarrow \mathbf{a}^{sh} - ((((n + t) - n_{\text{best}}) / NSD_{\text{total}}) \times |\mathbf{a} - \mathbf{a}^{sh}|)$.
39. $\mathbf{Ub}^{ex} \leftarrow \mathbf{b}^{sh} + ((((n + t) - n_{\text{best}}) / NSD_{\text{total}}) \times |\mathbf{b} - \mathbf{b}^{sh}|)$.
40. $D_{ex} := [\mathbf{Lb}^{ex}, \mathbf{Ub}^{ex}] \cap [\mathbf{a}, \mathbf{b}]$, $D_c = D_{ex}$.
41. **end**
42. **end**
43. **End**

the accuracy of KBSMs will be regularly improved in D_{sub}, and accuracy of $\mathbf{x}^*$ is expected to be improved; note that, here the time of the optimization process is short because of the lack of expensive function evaluations. After terminating each of two optimizers, array $\mathbf{y}(\mathbf{x})$ is called to evaluate the true responses at the current solution and record those in the database.

Step 3. (lines 11–20): This step is the same as lines 7–16 of Algorithm 2(d).

5. Computational experiments

5.1. Two illustrative examples

Two graphical test problems while their feasible regions are disconnected are employed to indicate that how KASRA acts; disconnected feasible regions are intentionally considered to show the applicability of KASRA. The first test problem is to minimize a modified version of the Branin (MBR) function [13], defined over the range $0 \leq x_i \leq 1$, $i = 1, 2$, subject to a normalized version of the Six-hump Camelback (SC) function known as Gomez#3 function [20,21], with an additional sine wave to increase the modality presented by Parr et al. [14]. This problem has an active

Algorithm 2(b) The pseudo codes to update a new infill point (when $\mathbf{x}^*$ is infeasible)

Input: (1) $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{1+k}(\mathbf{x})]^T$: a vector involves 1 objective and k constraints deterministic expensive-to-evaluate black-box functions whose function values may be the output of a complex computer simulation; (2) $\mathbf{X}_s$, $\mathbf{Y}_s$: a set of selected sample points with their responses; (3) $D_c \subset D$: the current design sub-space; (4) *sw*: a flag for running Alg. 2(b).

Output: (1) $\{\mathbf{x}^{\text{infill}}, \mathbf{y}(\mathbf{x}^{\text{infill}})\}$: a new infill point; (2) *sw*.

Begin
  if $s \le 30$ **then**
    $g = 50, p = 60$.
  elseif $30 < s \le 60$ **then**
    $g = 40, p = 50$.
  elseif $60 < s \le 80$ **then**
    $g = 30, p = 50$.
  elseif $80 < s \le 100$ **then**
    $g = 15, p = 40$.
  elseif $s > 100$ **then**
    $g = 10, p = 40$.
  end
  With $\mathbf{X}_s$ and $\mathbf{Y}_s$, maximize Eq.6 using genetic algorithm with maximum generation g and population size p to estimate hyperparameters $\boldsymbol{\theta}$ and $\mathbf{p}$ for each $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$.
  Construct the Kriging-based surrogate models of objective and constraint functions $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$, via Eq.7.
  Calculate an estimated MSE via Eq.8.
  Maximize sum of $P[F(\mathbf{x})]$ of each constraint in Eq.14 using genetic algorithm with maximum generation $g = 50$, and population size $p = 100$, to update a new infill point as $\mathbf{x}^{\text{infill}}$.
  Call $\mathbf{y}(\mathbf{x})$ to obtain the true responses of the new sampled point as $\mathbf{y}(\mathbf{x}^{\text{infill}})$.
  if $\{\sum \max(0, y_j^{(\text{infill})}): j = 2, \ldots, k+1\} \le 0$; ($\mathbf{x}^{\text{infill}}$ is feasible) **then**
    $sw = 1$.
  end
End

Algorithm 2(c) The pseudo codes to update new infill point (when $\mathbf{x}^*$ is feasible)

Input: (1) $D = [\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$: original search space; (2) $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{1+k}(\mathbf{x})]^T$: a vector involves 1 objective and k constraints deterministic expensive-to-evaluate black-box functions whose function values may be the output of a complex computer simulation; (3) $\mathbf{X}_s$, $\mathbf{Y}_s$: a set of selected sample points with their responses; (4) $D_c \subset D$: the current design sub-space; (5) D_{sh}: the shrunken design sub-space; (6) $w^{\max} = 0.5$: maximum weighting factor; (7) $w^{\min} = 0$: minimum weighting factor;

Output: (1) a new infill point $\{\mathbf{x}^{\text{infill}}, \mathbf{y}(\mathbf{x}^{\text{infill}})\}$.

Begin
  if $s \le 30$ **then**
    $g = 50, p = 60$.
  elseif $30 < s \le 60$ **then**
    $g = 40, p = 50$.
  elseif $60 < s \le 80$ **then**
    $g = 30, p = 50$.
  elseif $80 < s \le 100$ **then**
    $g = 15, p = 40$.
  elseif $s > 100$ **then**
    $g = 10, p = 40$.
  end
  With $\mathbf{X}_s$ and $\mathbf{Y}_s$, maximize Eq.6 using genetic algorithm with maximum generation g and population size p to estimate hyperparameters $\boldsymbol{\theta}$ and $\mathbf{p}$ for each $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$.
  Construct the Kriging-based surrogate models of objective and constraint functions $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$, via Eq.7.
  Calculate an estimated MSE via Eq.8.
  V_D = volume (D), V_c = volume (D_c), V_{sh} = volume (D_{sh}).
  $C \leftarrow (w^{\min} - w^{\max}) / (V_D - V_{sh})$.
  $w \leftarrow (C \times (V_c - V_{sh})) + w^{\max}$.
  Maximize weighted CE[$I(\mathbf{x})$] via the combination of Eq.13 and Eq.15 using genetic algorithm with maximum generation $g = 50$, and population size $p = 100$, to update a new infill point as $\mathbf{x}^{\text{infill}}$.
  Call $\mathbf{y}(\mathbf{x})$ to obtain the true responses of the new sampled point as $\mathbf{y}(\mathbf{x}^{\text{infill}})$.
End

global optimum at [0.9406, 0.3171] with an objective function value of 12.0050. The second test problem is to minimize SC function defined over the range $-1 \le x_i \le 1, i = 1, 2$ subject to sine terms in the constraint function as used by Dong et al. [34], where an active global optimum located at [0.1093, −0.6235] with an objective function value of −0.9711. The functions of the illustrative examples are in the supplementary material, Appendix A. Since the constraint functions are entirely difficult to

Algorithm 2(d) The pseudo codes to update new infill point in phase 2, stage 1

Input: (1) $D = [\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$: original search space; (2) $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{1+k}(\mathbf{x})]^T$: a vector involves 1 objective and k constraints deterministic expensive-to-evaluate black-box functions whose function values may be the output of a complex computer simulation; (3) $D_{sub} = [\mathbf{lb}^{sub}, \mathbf{ub}^{sub}] \subset D$: a sub-region for searching at phase 2; (4) $\mathbf{X}_s = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(s)}]^T \in D_{sub}$, $\mathbf{Y}_s$: a set of selected sample points with their responses; (5) $\mathbf{x}^*$: current best point;
Output: (1) $\{\mathbf{x}^{infill}, \mathbf{y}(\mathbf{x}^{infill})\}$: a new infill point; (2) D_{sub}.

Begin
  With $\mathbf{X}_s$ and $\mathbf{Y}_s$, maximize Eq.6 using genetic algorithm with maximum generation $g = 50$, and population size $p = 60$, to estimate hyperparameters $\boldsymbol{\theta}$ and $\mathbf{p}$ for each $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$.
  Construct the Kriging-based surrogate models of objective and constraint functions $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$, via Eq.7.
  Calculate an estimated MSE via Eq.8.
  Maximize CE[$I(\mathbf{x})$] via Eq.12 and Eq.15 using genetic algorithm with maximum generation $g = 50$, and population size $p = 100$, to update a new infill point as $\mathbf{x}^{infill}$.
  Call $\mathbf{y}(\mathbf{x})$ to obtain the true responses of the new sampled point as $\mathbf{y}(\mathbf{x}^{infill})$.
  for $i \leftarrow 1$ to d **then**
    if $x_i^{infill} \le lb_i + (0.02 \times (ub_i^{sub} - lb_i^{sub}))$ **then**
      $lb_i^{sub\_upd} = lb_i - (0.5 \times (ub_i^{sub} - lb_i^{sub}))$
    end
    if $x_i^{infill} \ge ub_i - (0.02 \times (ub_i^{sub} - lb_i^{sub}))$ **then**
      $ub_i^{sub\_upd} = ub_i + (0.5 \times (ub_i^{sub} - lb_i^{sub}))$
    end
  end
  $D_{sub} := [\mathbf{Lb}^{sub\_upd}, \mathbf{Ub}^{sub\_upd}] \cap [\mathbf{a}, \mathbf{b}]$.
End

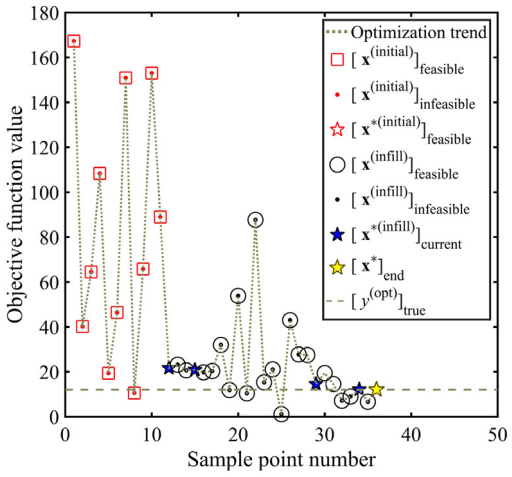
Algorithm 2(e) The pseudo codes to update new infill point in phase 2, stages 2–10

Input: (1) $D = [\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$: original search space; (2) $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{1+k}(\mathbf{x})]^T$: a vector involves 1 objective and k constraints deterministic expensive-to-evaluate black-box functions whose function values may be the output of a complex computer simulation; (3) $D_{sub} = [\mathbf{lb}^{sub}, \mathbf{ub}^{sub}] \subset D$: a sub-region for searching at phase 2; (4) $\mathbf{X}_s = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(s)}]^T \in D_{sub}$, $\mathbf{Y}_s$: a set of selected sample points with their responses; (5) $\mathbf{x}^*$: current best point; (6) NFE_l: number of function evaluation at phase 2;
Output: (1) $\{\mathbf{x}^{infill}, \mathbf{y}(\mathbf{x}^{infill})\}$: a new infill point; (2) D_{sub}.

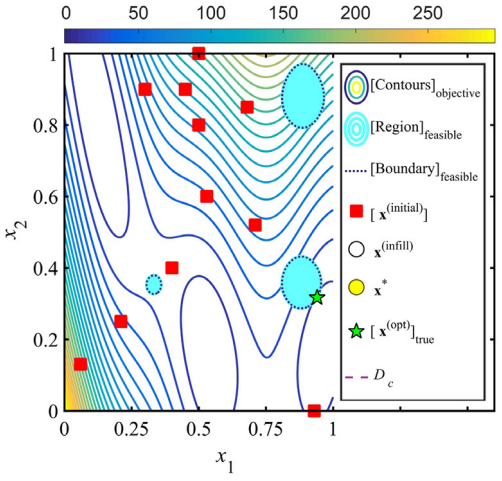
Begin
  With $\mathbf{X}_s$ and $\mathbf{Y}_s$, maximize Eq.6 using genetic algorithm with maximum generation $g = 50$, and population size $p = 60$, to estimate hyperparameters $\boldsymbol{\theta}$ and $\mathbf{p}$ for each $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$.
  Construct the Kriging-based surrogate models of objective and constraint functions $\{\hat{y}_j(\mathbf{x}): j = 1, \ldots, k+1\}$, via Eq.7.
  if NFE_l is even **then**
    Call exponential tunneling optimizer and execute a global search from starting point $\mathbf{x}^*$ on $\hat{\mathbf{y}}(\mathbf{x})$ for finding global optima of surrogate functions as $\mathbf{x}^{infill}$.
    Call $\mathbf{y}(\mathbf{x})$ to obtain the true responses of the new sampled point as $\mathbf{y}(\mathbf{x}^{infill})$.
  elseif NFE_l is odd **then**
    Call SQP optimizer and execute a local search from starting point $\mathbf{x}^*$ on $\hat{\mathbf{y}}(\mathbf{x})$ for finding local optima of surrogate functions as $\mathbf{x}^{infill}$.
    Call $\mathbf{y}(\mathbf{x})$ to obtain the true responses of the new sampled point as $\mathbf{y}(\mathbf{x}^{infill})$.
  end
  for $i \leftarrow 1$ to d **then**
    if $x_i^{infill} \le lb_i + (0.02 \times (ub_i^{sub} - lb_i^{sub}))$ **then**
      $lb_i^{sub\_upd} = lb_i - (0.5 \times (ub_i^{sub} - lb_i^{sub}))$
    end
    if $x_i^{infill} \ge ub_i - (0.02 \times (ub_i^{sub} - lb_i^{sub}))$ **then**
      $ub_i^{sub\_upd} = ub_i + (0.5 \times (ub_i^{sub} - lb_i^{sub}))$
    end
  end
  $D_{sub} := [\mathbf{Lb}^{sub\_upd}, \mathbf{Ub}^{sub\_upd}] \cap [\mathbf{a}, \mathbf{b}]$.
End

model accurately, surrogate-based methods have challenges with these problems.

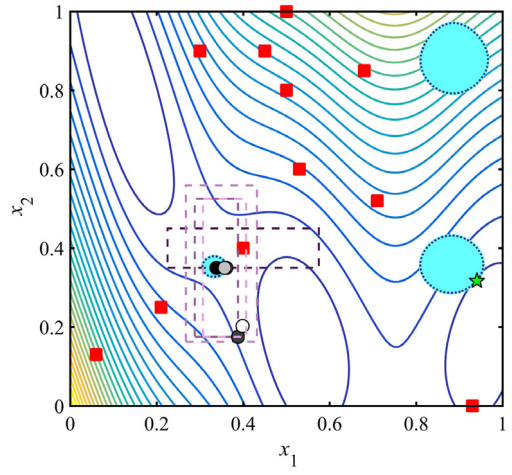
Figs. 2(a) and 3(a) show the optimization trends of KASRA on these two examples. Totally, 36 and 44 expensive function evaluations (involve initial samples) are needed to find respectively the global optimum for first and second problems. Figs. 2(b) and 3(b) present a contour plot of the objective function with the bounded feasible space and eleven initial sample points as well as the location of the global optimum for both of these problems. The placement of the initial sample points affects the infill sampling performance [14,15] due to an initial point, by pure luck, positioned close to the global optimum is a guide for the search algorithm to accelerate convergence. In this case, the possible inefficiencies of the algorithm are not well understood. It is the reason that a "worse initial sample points" without including points near to global optimum selected for both problems. A sequence of updated $\mathbf{x}^{infill}$ is distinguishable with a black–white color spectrum from darkness to light as depicted in Figs. 2(c)–(f)



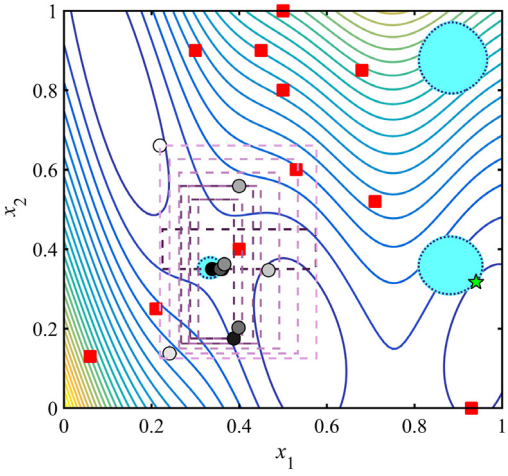
(a) Optimization trend of KASRA on constrained-MBR problem



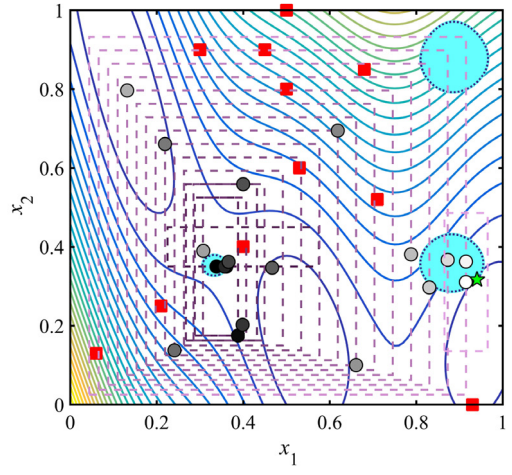
(b) Objective contour plot, initial samples, feasible regions, and location of the global optimum of constrained-MBR problem



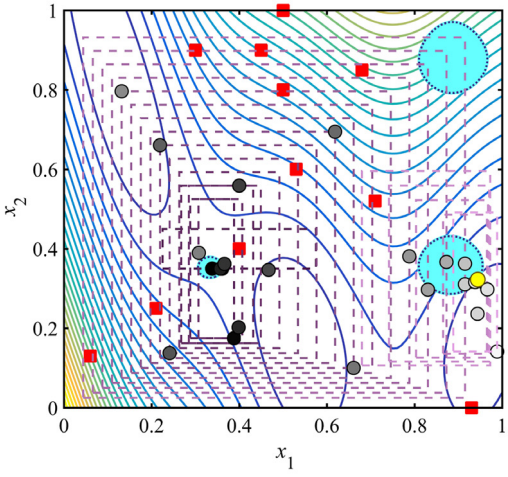
(c) Iterations 1–5



(d) Iterations 6–10



(e) Iterations 11–20

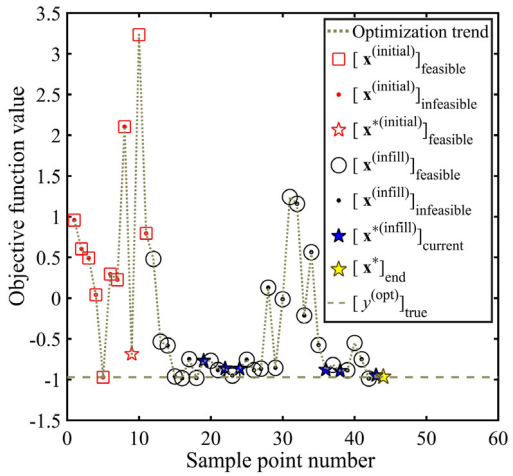


(f) Iterations 21–25

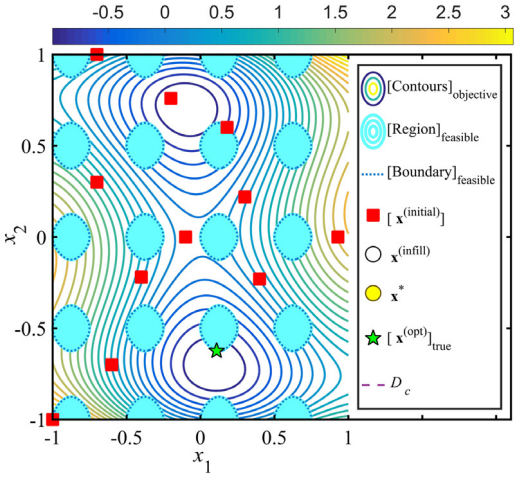
Fig. 2. Iterative results of KASRA on constrained-MBR problem.

and 3(c)–(f); the same approach has been applied to detect the sequence of produced box-shaped regions D_c.

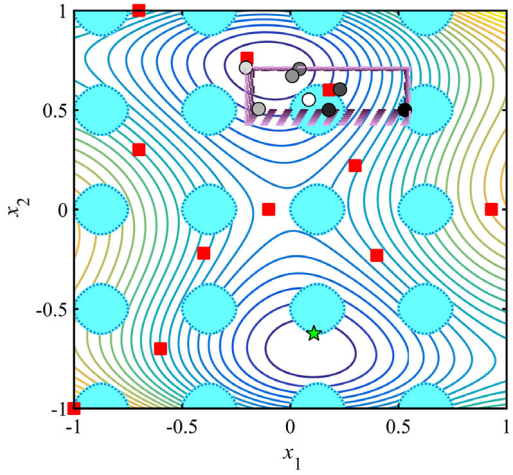
As Fig. 2(b) shows, there is no feasible point between the initial samples for the first problem. In this situation, the best point to start KASRA procedure is the one with the least constraints violation, the point in the vicinity of the smallest feasible region. According to Algorithm 2(a), the sub-space must be shrunken to surround the current best point $\mathbf{x}^*$. For this purpose, by considering the $n_{\text{close}} = 5$ points near the current $\mathbf{x}^*$, the hyperparameters for the kriging model of the objective function are calculated as



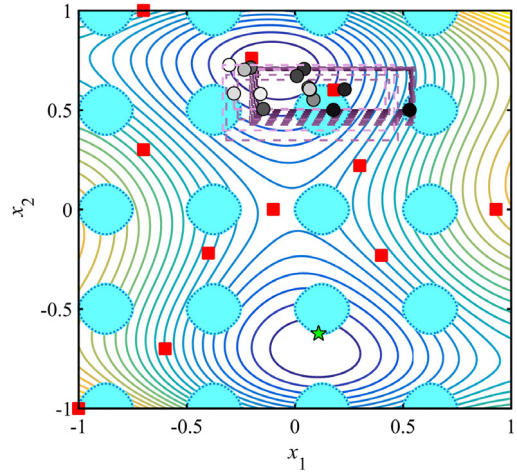
(a) Optimization trend of KASRA on constrained-SC problem



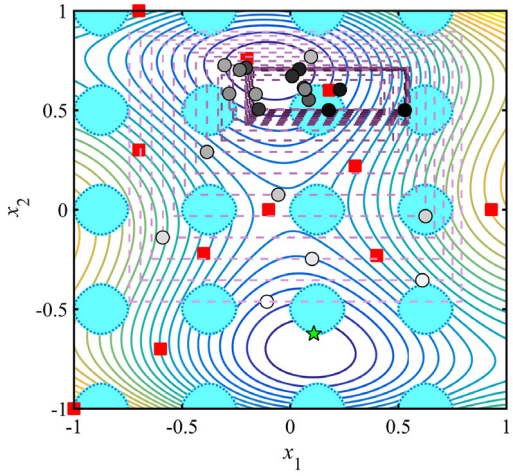
(b) Objective contour plot, initial samples, feasible regions, and location of the global optimum of constrained-SC problem



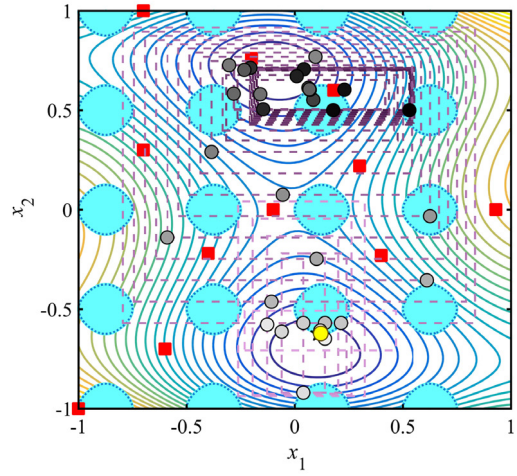
(c) Iterations 1–8



(d) Iterations 9–16



(e) Iterations 17–24



(f) Iterations 25–33

Fig. 3. Iterative results of KASRA on constrained-SC problem.

$\theta_1 = 0.0010$ and $\theta_2 = 380.5836$. Because the second variable is more active, the shrunken box size $D_c = D_{sh}$ will be 35% and 10% of the original intervals for the first and second variables, respectively, i.e. more focus on the second variable (to see the detailed results and to use the magnification feature of the figures, use the files reported to the Ref. [42]). The objective and constraint meta-models are constructed in this box, and the first infill point is updated using Algorithm 2(b).

From Figs. 2(a) and (c), it is clear that the first infill point is feasible with coordinate, objective, and constraint function values

[0.3381, 0.3500], 21.6754, and −0.1501 respectively, therefore this point is selected as the current $\mathbf{x}^*$ and a new shrunk box $D_c = D_{sh}$ needs to be rebuilt with this center. This time the values of the hyperparameters are calculated as $\theta_1 = 41.3348$ and $\theta_2 = 0.0010$, so the shrunken box size $D_c = D_{sh}$ will be 10% and 35% of the original intervals for the first and second variables, respectively, i.e. more focus on the first variable. The second infill point then is searched in this box, which resulted in an infeasible point with coordinate, objective, and constraint function values [0.3881, 0.1750], 23.1328, and 5.3904, respectively. As a result, the current $\mathbf{x}^*$ remains unchanged, and the new expanded box $D_c = D_{ex}$ is created as the previous box expands.

The third infill point with coordinate, objective, and constraint function values [0.3609, 0.350], 20.6111, and 0.0023, respectively, is another infill point, so the new box expands again. As the search process for the new box continues, this time a point is found with coordinate, objective, and constraint function values [0.3569, 0.3500], 20.8358, and −0.0399 respectively, whose value of the objective function is less than the previous best point; as a result, the fourth infill point chosen as the current $\mathbf{x}^*$, see Fig. 2(a).

Finding a new $\mathbf{x}^*$ requires a shrunk box $D_c = D_{sh}$ around it. Using Algorithm 2(a), the values of the hyperparameters are $\theta_1 = 165.1636$ and $\theta_2 = 6.3101$, so the shrunken box size $D_c = D_{sh}$ will be 10% and 35% of the original intervals for the first and second variables, respectively, see Fig. 2(c). The fourth infilling point is a local minimum for this small feasible region, and other feasible regions are expected to be explored in subsequent iterations as the search boxes grow larger; these regions are discovered in update 16, as shown in Fig. 2(e). However, according to Figs. 2(a) and (e), the sixteenth infill point with coordinate, objective, and constraint function values [0.8305, 0.2971], 27.7240, and 0.2503, respectively, is still an infeasible point.

The seventeenth infill point with coordinate, objective, and constraint function values [0.8729, 0.3667], 27.4533, and −1.0698, respectively, is feasible near to the correct solution. Nevertheless, its objective function value is no better than the current $\mathbf{x}^*$. The eighteenth infill point with coordinate, objective, and constraint function values [0.9153, 0.3107], 14.5489, and −0.3571, respectively, has a better objective function value, therefore is a new $\mathbf{x}^*$. As a result, a new shrunk box $D_c = D_{sh}$ needed to be created according to the hyperparameters $\theta_1 = 4.6881$ and $\theta_2 = 3.2862$ which are 10% and 35% of the original intervals for the first and second variables, respectively, see the lower right corner of Fig. 2(e). The boxes are expanded as shown in Fig. 2(f) and two infeasible infill points 21 and 22 updated in these boxes. Point 23 is a new $\mathbf{x}^*$ with coordinate, objective, and constraint function values [0.9384, 0.3165], 12.1735, and −0.0382, respectively. So a new shrunk box $D_c = D_{sh}$ matches the hyperparameters $\theta_1 = 43.0651$ and $\theta_2 = 12.7555$ are 10% and 35% of the original intervals for the first and second variables, respectively. Point 24 is an infeasible point with coordinate, objective, and constraint function values [0.9884, 0.1415], 6.6104, and 7.3966, respectively. Finally, point 25 with coordinate, objective, and constraint function values [0.9441, 0.3226], 12.0837, and −0.0076, respectively, has an error of 0.655% to the true global optimum (from Eq. (17)) as demonstrated by a yellow star and a yellow circle in Fig. 2(a) and (f) respectively, which results at the end of the search process.

Unlike the first problem, KASRA procedure in the second problem begins from an initial sample point that is feasible as shown in Fig. 3(b). This ninth initial point with coordinate, objective, and constraint function values [0.1800, 0.6000], −0.6862, and −0.0795, respectively, is depicted by a red hollow star in Fig. 3(a). According to Algorithm 2(a) and by considering the $n_{\text{close}} = 5$ points near the current $\mathbf{x}^*$, the hyperparameters for the kriging model of the objective function are calculated as $\theta_1 = 0.5844$ and $\theta_2 = 13.6792$. Because the second variable is more active, the shrunken box size $D_c = D_{sh}$ will be 35% and 10% of the original intervals for the first and second variables, respectively.

In the following, the box size $D_c = D_{ex}$ expands until the eighth infill point is found with coordinate, objective, and constraint function values [0.0856, 0.5518], −0.7707, and −0.6751, respectively. Since the value of the objective function at this point is less than the previous best point, it is chosen as the current $\mathbf{x}^*$. Thus, given the values of the hyperparameters, $\theta_1 = 5.6514$ and $\theta_2 = 8.2680$ for the kriging model of the objective function, a shrunk box $D_c = D_{sh}$ is created with size 35% and 10% of the original intervals for the first and second variables, respectively.

The process of expanding and shrinking of the boxes continues with the new infill points until the local minimum of the adjacent feasible area exploited as point 13. The thirteenth point is chosen as the current $\mathbf{x}^*$ with coordinate, objective, and constraint function values equal to [0.0678, 0.6051], −0.8689, −0.0002 respectively. Given the values of the hyperparameters $\theta_1 = 3.1227$ and $\theta_2 = 5.1396$, the new box size $D_c = D_{sh}$ is contracted at 35% and 10% of the original intervals for the first and second variables, respectively, see Fig. 3(a) and (d).

In the following, because no better infill point is found, the box size is expanded until the desirable area is found in iteration 24, as shown in Fig. 3(e)—here KASRA ignores other feasible regions because their objective function values are not less than the function value of the current best point.

As the search box grows, the 25th infill point with coordinate, objective, and constraint function values [0.1395, −0.5699], −0.8797 and −0.6213 respectively, is chosen as the current $\mathbf{x}^*$, see Fig. 3(a) and (f). As a result, the size of the new shrunk box $D_c = D_{sh}$, concerning the obtained hyperparameters $\theta_1 = 2.7626$ and $\theta_2 = 2.3492$, equals 10% and 35% of the original interval for the first and second variables, respectively. Also, the 27th and 32nd infill points are chosen as the current $\mathbf{x}^*$, respectively, according to their value. Finally, by finding the 33rd infill point with coordinate, objective, and constraint function values equal to [0.1206, −0.6222], −0.9664, and −0.0335 respectively, the search process ends, because the solution is close enough to the true global optimum (the error percentage to the true global optimum is 0.487% from in Eq. (17)), Figs. 3(a) and Fig. 3(f) show it by a yellow star and a yellow circle respectively.

KASRA has not jumped to its second phase in any of these examples, because the objective and constraint functions do not have much nonlinear behavior in the vicinity of the global optimum coordinates, which results in the locally well-approximated KBSMs. It should be mentioned that the process could be a little shorter or longer, depending on the starting point location.

5.2. Experimental setup

The performance of KASRA is studied in the sense of global convergence capability, efficiency, and robustness on a broad set of benchmark nonlinear constrained optimization problems including twenty-two mathematical cases and ten engineering applications having various properties. The formulations, dimensions, design ranges, best known objective value, best-known solution, and the related literature of the benchmarks are presented in the supplementary material, Appendices B and C. Since the proposed approach inherently has statistical properties, each test case has been run several times to eliminate randomness matter. It is worth noting that many other studies conducted verification tests on only a handful of optimization problems while this study executes extensive tests to cover the constrained optimization problems that commonly encountered in the real-world engineering design.

Table 1
Mean number of simulations (for 30 trials) to obtain a feasible point and to reach the target feasible objective value starting from an LHD of size $d + 1$ for the different algorithms.

Problem	Best known value	Target value	KASRA		The best published results from comparison of COBRA-Local, COBRA-Global, Extended ConstrLMSRBF, SDPEN, and NOMADm-DACE algorithms			
			Feasible	Target	Feasible	Algorithm	Target	Algorithm
G1	−15	−14.85	**4.4 (3.08)**	197.3 (24.75)	15(0)	COBRA	>**120.27 (1)**	NOMADm-DACE
G3 (MOD)	−26.5	−27.0858	49.27 (19.40)	363.10 (56.43)	**23.53 (0.20)**	COBRA	**141.70 (8.60)**	Extended ConstrLMSRBF
G5 (MOD)	5126.50	5150	23.37 (9.38)	63.70 (13.08)	**6.37 (0.09)**	COBRA	**12.90 (0.46)**	COBRA-Local
G6	−6961.8139	−6800	11.10 (3.92)	**41.23 (19.55)**	**10.90 (0.30)**	COBRA	53.57 (13.95)	COBRA-Local
G7	24.3062	25	**29.87 (7.81)**	70.67 (13.02)	39.83 (2.90)	Extended ConstrLMSRBF	**58.90 (1.55)**	NOMADm-DACE
G8	−0.0958	−0.09	7.9 (5.03)	**26.63 (11.46)**	5 (0)	NOMADm-DACE	30.30 (2.79)	COBRA-Local
G9	680.6301	1000	**4.03 (4.81)**	**25 (18.35)**	21.50 (1.85)	COBRA	>88.63 (1)	SDPEN
G10 (MOD)	7049.25	8000	48.8 (30.48)	**185.87 (28.59)**	**22.80 (1.51)**	COBRA	>193.67 (1)	COBRA-Global
G13 (MOD)	0.0035	0.005	**4.5 (3.93)**	**88.47 (17.51)**	8.60 (0.67)	Extended ConstrLMSRBF	97.03 (2)	NOMADm-DACE
G16	−1.9052	−1.75	29.77 (11.12)	61.2 (16.67)	**14.70 (2.37)**	COBRA	**38.40 (3.63)**	Extended ConstrLMSRBF
G18	−0.8660	−0.8	**20.53 (4.42)**	**125.60 (51.94)**	86.23 (5.66)	NOMADm-DACE	>190.33 (14)	NOMADm-DACE
G19	32.6556	40	**3.93 (2.10)**	**391 (54.11)**	16.50 (0.50)	COBRA	698.53 (75.25)	COBRA-Local
G24	−5.5080	−5	1.97 (1.43)	11.33 (4.36)	**1.33 (0.09)**	COBRA, SDPEN, & Extended ConstrLMSRBF	**9.00 (0)**	COBRA
PVD (MOD)	5804.38	6000	27.63 (16.81)	**91.00 (32.00)**	**7.87 (0.35)**	COBRA	155.43 (38.19)	COBRA-Global
WBD (MOD)	1.6952	2.5	**21.83 (12.89)**	**58.27 (22.60)**	25.00 (4.10)	Extended ConstrLMSRBF	164.57 (12.23)	COBRA-Local
SRD	2994.47	2995	10.90 (4.79)	**28.33 (6.14)**	**9.47 (0.11)**	COBRA	33.53 (1.57)	COBRA-Global

The experiments of KASRA for the test cases are executed in Matlab™ R2016b (9.1) on the computer equipped with an Intel Core i5-6600U CPU (3.50 GHz) and 8 GB memory. The acquired statistical results were compared with the collection of published results transcribed from other papers due to the lack of publicly available software to implement the competitor methods. The selected alternative methods divided into two categories. The first category includes Extended ConstrLMSRBF (Constrained Local Metric Stochastic Radial Basis Function) [43,44] and two version of COBRA (Constrained Optimization By RAdial basis function interpolation) called COBRA-Local and COBRA-Global, that all use RBF surrogates in a two-phase structure where the first phase detects a feasible point while the second phase improves this point [44]. Two other alternative methods are SDPEN [45] and NOMADm-DACE [46]. SDPEN is a derivative-free extension of a sequential penalty method for nonlinear programming without using the surrogates [44]. NOMADm-DACE is a Matlab™ version of NOMAD [47], which implements the Mesh Adaptive Direct Search (MADS) algorithm running with a DACE surrogate [48]. The significant features of the selected algorithms in the first category are that they can be used for HEB constrained problems where even there is no feasibility in the initial samples. The best-published results of these algorithms that were published by Regis [44] have been used for the first set of comparisons in Table 1. These results are gained by running 30-times on each test problem to detect a feasible point and pass a particular target value. Each algorithm is allowed up to 500 function evaluations for the test problems (one call to evaluate $f(x)$ and $g(x)$ as one function evaluation). All algorithms begin with an independent LHD of size $d + 1$ whose points are all infeasible. The second category includes MSSR (Multi-Start Space Reduction) [32], MS is the MSSR without space reduction strategies [32], and SCGOSR (Surrogate-based Constrained Global Optimization using Space Reduction) [34], that all use the Kriging surrogates. SCGO and RBFCGOSR also are included in the second set of comparisons. SCGO is the SCGOSR algorithm ignoring space reduction. RBFCGOSR is the same as SCGOSR while the surrogate model uses the cubic RBF. The best-published results from the execution of these algorithms by Dong [34] have been employed for the second set of comparisons in Table 2. These results are obtained by running 10-times on each test problem to reach a particular target value. The computational budget is up to 500 function evaluations. Here too, one call returns the values of the objective and all constraint functions.

6. Results and discussion

6.1. Comparisons for the first alternative methods

The related raw datasets of this work including the comparative and complementary studies are available at Mendeley Data Repository [42].

Table 1 marked the best statistical results with boldface after comparing KASRA with COBRA-Local, COBRA-Global, Extended ConstrLMSRBF, SDPEN, and NOMADm-DACE. The symbol "$>$" indicates that at least one of the trials cannot detect the target value within 500 function evaluations. The number inside the parenthesis is the standard deviation of the mean or otherwise the number of trials that failed to reach the target value. The "mean" is calculated by adding up all the NFE of each trial and then divide by the number of trials. For example, the number of samples to capture the feasible solution of G1 for 30 trials are

Table 2
Statistical NFE, best values, and CPU time (for 10 trials) to reach the target feasible objective value for the different algorithms.

Problem	Best known value	Target value	KASRA						The best published results between SCGOSR, RBFCGOSR, SCGO, MSSR, MS, and MSRBF algorithms					
			NFE				Best value	CPU time (s)	NFE				Best value	Algorithm
			Min	Median	Max	Mean			Min	Median	Max	Mean		
G4 (MOD)	−31025.56	−31025	47	52.5	**57**	**51.8**	[−31025.56, −31025.53]	1020.3	**32**	**35.5**	174	53.9	[−31026, −31025]	SCGOSR
G6	−6961.8139	−6960	36	**39.5**	**55**	**40.40**	[−6961.8, −6960.3]	342.4	**27**	44.5	71	46.4	[−6961.8, −6961.2]	RBFCGOSR
G7	24.3062	25	**79**	**85**	**90**	**84.50**	[24.3765, 24.7965]	1435.3	102	199.5	239	178.2	[24.3149, 24.9969]	SCGOSR
G8	−0.09583	−0.09580	29	**42.5**	**49**	**41.2**	[−0.09583, −0.09582]	514.5	**24**	47.5	80	51.8	[−0.0958, −0.0958]	SCGOSR
G9	680.6301	1000	**13**	**15.5**	**47**	**22**	[761.92, 994.21]	447.9	60	102.5	189	109.1	[828.79, 999.87]	MSSR
TSD	0.01267	0.0128	**42**	**48**	**54**	**47.8**	[0.01267, 0.01277]	675.8	43	69	114	75.7	[0.01267, 0.01278]	SCGOSR
WBD	1.6952	1.8	**50**	**87**	**101**	**69.90**	[1.7252, 1.7991]	1273.5	72	97	153	101.9	[1.7249, 1.7888]	SCGOSR
PVD	5885.33	6000	52	55	70	56	[5886.8, 5998.5]	691.8	**23**	**28**	**49**	**29.7**	[5885.4, 5965.3]	MS
SRD	2994.47	3000	38	**43**	**46**	**42**	[2997.2, 2999.9]	1159.1	**35**	60.5	272	88.1	[2994.5, 2997.8]	SCGOSR
SCBD	62791	65 000	95	**100.5**	**116**	**104.40**	[63571.1, 64960.9]	1425.0	**62**	119.5	297	152.5	[62861, 64895]	SCGOSR

[2, 8, 3, 11, 4, 3, 4, 4, 2, 3, 11, 5, 2, 2, 1, 3, 4, 2, 3, 1, 13, 5, 9, 3, 4, 3, 2, 6, 3, 6], and so the mean number of simulations to reach feasibility is 4.4 (detailed results of other problems are available at the Mendeley Data Repository in Ref. [42]). The number of samples to capture the feasible region is affected by the distribution of samples in the initial LHD. In other words, if one of the initial samples stands near the feasible region, it is more likely that this region is found faster. Indeed, the shrunken box around this point (the one with the least constraint violation) will probably comprise the feasible area, and as a result, the feasible point is quickly found.

Table 1 shows that KASRA is able to enter the feasible region faster than 5 alternative algorithms on 7 test problems (G1, G7, G9, G13-MOD, G18, G19, WBD-MOD) and can be compared with the best results attained by the alternative methods on 4 other problems (G6, G8, G24, and SRD). However, the COBRA algorithm performs better to reach the feasibility on G3-MOD, G5-MOD, G16, G10-MOD, and PVD-MOD. It can be caused by starting the search algorithm in KASRA from a point far from the feasible region; this issue is possible to be improved by increasing the size of LHD. Of course, the unavailability and, naturally, the different initial LHD in the compared papers can influence these results. Anyway, it is obvious that KASRA algorithms are quite robust to reach a feasible point in all trials on test cases mentioned in Table 1. As can be seen from the results in Table 1, KASRA algorithm has shown its advantages on stability and efficiency to pass the target value in all trials on more test problems. The convergence performance of the proposed method in term of NFE is faster than 5 alternative methods on 10 test problems (G6, G8, G9, G13-MOD, G18, G19, PVD-MOD, WBD-MOD, and SRD) and is comparable with the best results attained by the alternative methods on 2 other problems (G7 and G24). It is obvious that, KASRA has an impressive speed of convergence on G9, G18, G19, PVD-MOD, and WBD-MOD compared to the alternative algorithms. However, as the observation on G1, G3-MOD, G5-MOD, and G16 in Table 1 shows, the more conservative search to capture the target objective value in KASRA did not always translate into less NFE.

6.2. Comparisons for the second alternative methods

Table 2 details the statistical NFE alongside the variation ranges of KASRA and the best results from SCGOSR, RBFCGOSR, SCGO, MSSR, MS, and MSRBF algorithms to compare the optimization robustness, convergence, and efficiency. From Table 2, it can be found that KASRA algorithm was performed very well compared to the five various competitive algorithms on 9 out of 10 test problems (all except PVD). KASRA is quite robust to reach a specific target value for all trials in less than 116 simulations. The convergence performance of the proposed method in term of maximum and mean NFE are much better than the best results attained by the alternatives on G7, G9, WBD, SRD, and SCBD in reaching the target value. Besides, the data in Table 2 explains that KASRA gets the best median optimal solutions for 8 out of 10 test problems (all except G4-MOD and PVD). Since KASRA uses a more conservative search, it is natural to expect that the results in terms of minimum NFE should not be always better than the comparison algorithms on all test cases. However, the minimum NFE of KASRA is either better or competitive with the best results attained by the alternative methods, except for PVD and SCBD. It should be noted that KASRA compared to alternatives can successfully find the target value with fewer function evaluations for problems with more dimensions and constraints.

The average runtime to reach a specific target value for each of the test problems is listed in Table 2. In this way, the readers can compare CPU time for problems with different dimensions and various constraints (see the computer specifications used for this work in Section 5.2.). From Table 2, the higher the number of variables and constraints and the more nonlinear the problem functions, the longer the time of solution. The most CPU time is related to G7 with 10 dimensions and 8 constraints and the least CPU time is related to G2 with 2 dimensions and 2 constraints.

Intuitively, alternatives can obtain a more accurate result than KASRA, while the proposed algorithm mostly can be more efficient with fewer NFE. It is possible that the results get closer to the real global optimum value if the algorithm is allowed to continue, especially using phase 2. The next section proves this matter by running more experiments, where the algorithm will stop when Eq. (17) satisfied.

6.3. Complementary experiments

Further mathematical and engineering experiments are set up to show the extensive applicability of KASRA and its ability to find a more accurate solution. In Table 3, KASRA provides the statistical terms of 10 trials including best, worst, median, mean, and standard deviation on both NFE and the best feasible objective value, to reach the permissible relative error in Eq. (17) within 300 function evaluations. The size of the initial LHD is 11 and $2 \times (d + 1)$ for problems with $d \leq 4$ and $d \geq 5$ respectively.

It is obvious that KASRA can easily get much close to the real global optima on 14 out of 15 test problems (all except G2-MOD). As can be seen from the results in Table 3, the errors of the mean calculated with Eq. (17) for objective values indicate that the algorithm is well-suited to reach the accuracy better than 1% in NFE less than 172; here the best performance is related to the SB problem with the error of 0.173% at the maximum 70 function evaluations. The best known optimum value of the G2-MOD problem reported -0.4 in some literature [43,44]. With this value, KASRA is successfully implemented on this extremely non-linear optimization problem, but if the real optimum value -0.7473 is considered, the algorithm will be trapped into the local optima and at the best trial, the objective value will be reduced to -0.65645 in 300 function evaluations budget. It is remarkable that KASRA is stable and efficient on all of the engineering design applications and the maximum error of the mean is 0.623% at NFE less than 100.

All previous numerical experiments and comparison studies performed with minimum and maximum space reduction ($sr^{\min}$ and $sr^{\max}$) 10% and 35% respectively. Algorithm KASRA was tested with various percentages of space reduction to investigate the effectiveness of size $sr^{\min}$ and $sr^{\max}$ in the performance of phase 1 including: (1) $sr^{\min} = 5\%$, $sr^{\max} = 15\%$; (2) $sr^{\min} = 10$, $sr^{\max} = 35\%$; (3) $sr^{\min} = 25\%$, $sr^{\max} = 50\%$. Fig. 4 shows the Average Progress Curves (APC) of the three space reductions on the ten engineering test cases; APC-1, APC-2, and APC-3 state the curves with space reductions (1), (2), and (3). Table 4 provides statistics on the best feasible objective values realized after 50 function evaluations (regardless of the initial samples) during 10-trials; there is an exception for SCBD problem with 100 function evaluations. The initial LHD samples with size $(d + 1)$ are the same for different space reductions. The average progress curve is a graph of the mean of the best feasible objective function value versus the number of function evaluations [44]. Error bars show 95% t-confidence intervals of the mean to indicate variability in the results.

Five statistical terms including best, worst, median, mean, and standard deviation of the best feasible objective values are ranked for different $sr^{\min}$ and $sr^{\max}$ in Table 4. The total ranking of each term reported in Table 4. Intuitively, the best performance is obtained for $sr^{\min} = 25\%$, $sr^{\max} = 50\%$ in all terms except

Table 3
Statistics on NFE and the best feasible objective value (for 10 trials), to reach the permissible relative error 1% using KASRA algorithm.

Problem	Best known value	NFE					Objective value					
		Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std	Error of mean (%)
CGO	−0.9711	46	97	68.5	78.7	18.221	−0.971	−0.962	−0.968	−0.968	0.004	0.336
HESSE	−310	101	172	143.5	142.1	24.205	−309.99	−306.92	−307.99	−308.58	1.018	0.458
LEGO*	0	82	95	82.5	87.8	4.826	3.34×10^{-7}	8.99×10^{-6}	5.32×10^{-6}	5.09×10^{-6}	2.754×10^{-6}	0.00051
G2 (MOD)	−0.74731	300	300	300	300	0	−0.65645	−0.23240	−0.45649	−0.44463	0.137	40.50
G4	−30665.54	24	52	36	40	10.604	−30481.90	−30359.59	−30389.60	−30404.92	40.282	0.850
G12	−1	16	27	21.5	22.1	3.479	−0.99997	−0.99072	−0.99960	−0.99734	0.004	0.266
G23-MOD	−3900.00	102	134	119	114.9	11.426	−3894.29	−3862.55	−3877.61	−3880.19	12.215	0.508
TCD	66.1922	26	98	42.5	54.2	23.030	66.2180	66.8508	66.4712	66.6029	0.229	0.620
TBOT	8.5086	53	97	62.5	69.3	12.885	8.5086	8.5926	8.5457	8.5543	0.028	0.537
GTCD	12089.91	20	51	39	34	10.583	12123.90	12202.57	12155.45	12165.24	33.211	0.623
WPG	0.9051	52	67	59	59	5.967	0.9055	0.9140	0.9092	0.9103	0.003	0.578
SB	320.77	52	70	65	61.8	6.106	320.77	322.29	320.77	321.32	0.625	0.173

*The optimization procedure is terminated when the relative error was less than or equal to 0.001% for the LEGO problem.

Table 4
Statistics on the best feasible objective value of ten engineering benchmark problems for 10 trials obtained by KASRA algorithm for various percentages of minimum and maximum space reduction.

sr^{min}	sr^{max}		TCD	TBOT	TSD	GTCD	PVD	WPG	WBD	SB	SRD	SCBD	Total ranking
5%	15%	Best	96.94	8.53	0.003	12110.73	6545.86	0.91	2.27	329.16	3062.36	65507.804	21
		Worst	1388.73	12.88	0.171	13981.90	17426.52	2.29	9.70	581.39	4817.39	74259.455	30
		Median	796.59	8.68	0.015	12470.85	9693.24	1.68	3.73	476.82	4210.03	67168.118	25
		Mean	338.82	9.53	0.026	13025.84	10127.30	1.32	4.11	388.94	3767.97	69468.819	30
		Std	384.40	1.34	0.051	707.43	3631.77	0.47	2.14	85.41	626.23	2497.59	30
10%	35%	Best	66.74	8.59	0.003	12100.07	7188.45	0.91	2.02	324.46	3004.02	63683.86	**16**
		Worst	178.55	10.27	0.016	12816.06	12265.02	1.24	3.91	383.94	3266.18	68393.50	18
		Median	67.17	9.65	0.015	12122.01	8037.70	1.05	2.60	331.20	3038.54	66394.75	16
		Mean	81.51	9.34	0.010	12221.52	8598.14	1.03	2.73	355.93	3060.87	65705.63	18
		Std	34.59	0.67	0.006	216.22	1549.27	0.13	0.52	23.71	76.23	1545.19	17
25%	50%	Best	67.24	8.58	0.003	12111.64	7256.99	0.92	1.82	323.10	3004.68	62364.36	18
		Worst	98.13	9.64	0.110	12171.43	12976.03	1.13	2.92	349.22	3066.90	67945.53	**12**
		Median	71.08	8.99	0.016	12147.81	10331.39	1.03	2.23	344.12	3008.94	63672.26	**17**
		Mean	74.26	8.90	0.022	12141.60	9163.70	0.96	2.24	336.79	3021.95	64538.19	**12**
		Std	9.35	0.31	0.032	20.18	2013.57	0.06	0.43	7.33	17.58	1915.95	**13**

best; the total ranking of the term best belongs to $sr^{min} = 10\%$, $sr^{max} = 35\%$.

From Fig. 4, the mean of the best feasible objective function value obtained by APC-3 is better than other curves on all benchmarks. It consistently converges well toward the optimum value and goes down faster than the APC-1 and APC-2; for SCBD-problem, the objective function value will approach optimum value using the phase 2 of KASRA, as previously done in Table 3. However, there is no significant discrepancy between APC-2 and APC-3 on TCD, TBOT, PVD, WPG, WBD, and SB in Fig. 4(a), (b), (e)–(h).

Fig. 4 exhibits, both of APC-1 and APC-2 have been able to do their main task in Phase 1, which is to achieve the region of the optimal solution. The APC-1 on 5 of the 10 test problems (GTCD, WPG, WBD, SRD, and SCBD) illustrates the fact that small reduction space may not lead to finding a promising region in phase 1 particularly for the higher-dimensional problem in a very limited sampling budget. Though, the reduction space with larger sr^{min} and sr^{max} might produce poorly estimated surrogates for HEB problems with the dimensions higher than 10 and so miss global optimum zone in phase 1. Generally speaking, KASRA is partially sensitive to the selection of the maximum and minimum space reduction, according to the results of Table 4 and APCs of Fig. 4. The better results were obtained for $sr^{min} = 25\%$, $sr^{max} = 50\%$ and therefore, it is possible that better results can be obtained compared to the results listed in Tables 1, 2, and 3 by taking $sr^{min} = 25\%$, $sr^{max} = 50\%$.

6.4. Limitation of the proposed method

Though the desirable performance of KASRA is demonstrated, it is still possible that KASRA misses the true global optima for some test cases (G2-MOD). The cause may be due to the surrogate models not correctly approximate the trend of the true functions if the behavior of objective and constraint functions is too non-linear and the feasible region of the problem is very tight. This means, hyperparameters of the surrogate models are not correctly estimated because the global sub-optimizer such as the genetic algorithm does not perform its task properly. Moreover, KASRA is not predestinated to be used for equality constraints—efforts to find a feasible point will be failed in a reasonable budget (up to 500 NFE).

In addition, it assumed that the objective and constraint function values are always available; it may be possible to use the approximate function values of surrogates to recover the failure simulation. The other limitation is that the computational experiments of comparative studies used manual parameter settings. It might still be possible to improve the performance on the given problems by choosing the other parameters for the proposed algorithm. Finally, KASRA only is considered for the constrained

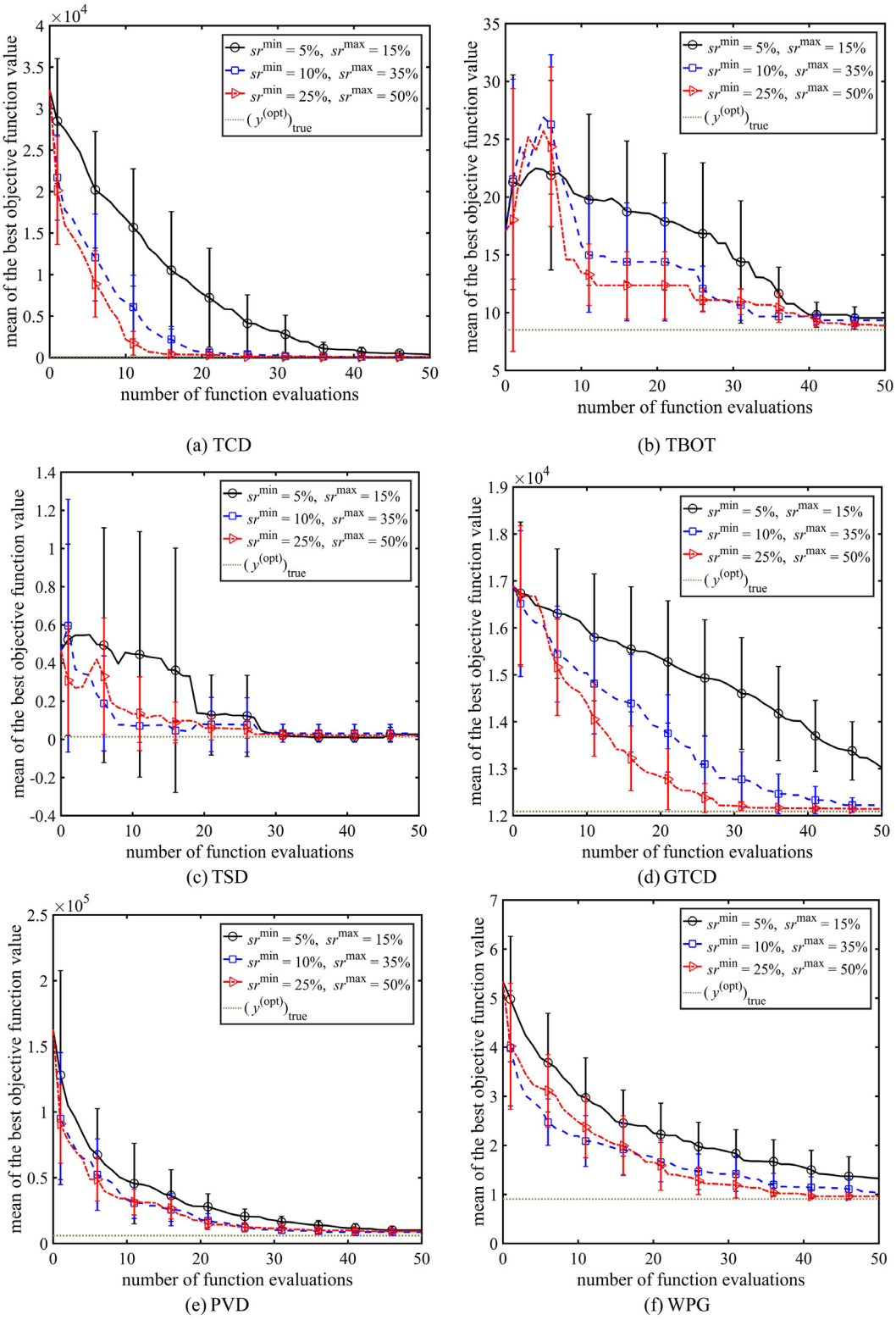


Fig. 4. Mean of the best feasible objective function values in 10 trials, obtained by various percentages of space reduction on engineering test problems. Error bars are 95% t-confidence intervals about the mean.

optimization problems with single objective function and continuous variables. These issues will all be addressed in the future work.

It is worth mentioning, an algorithm that universally operates the best for all optimization problems may be out of reach. This paper has really tried to confirm that the proposed idea is a promising approach for computationally expensive black-box constrained global optimization and performs reasonably well on a broad set of benchmark testing problems.

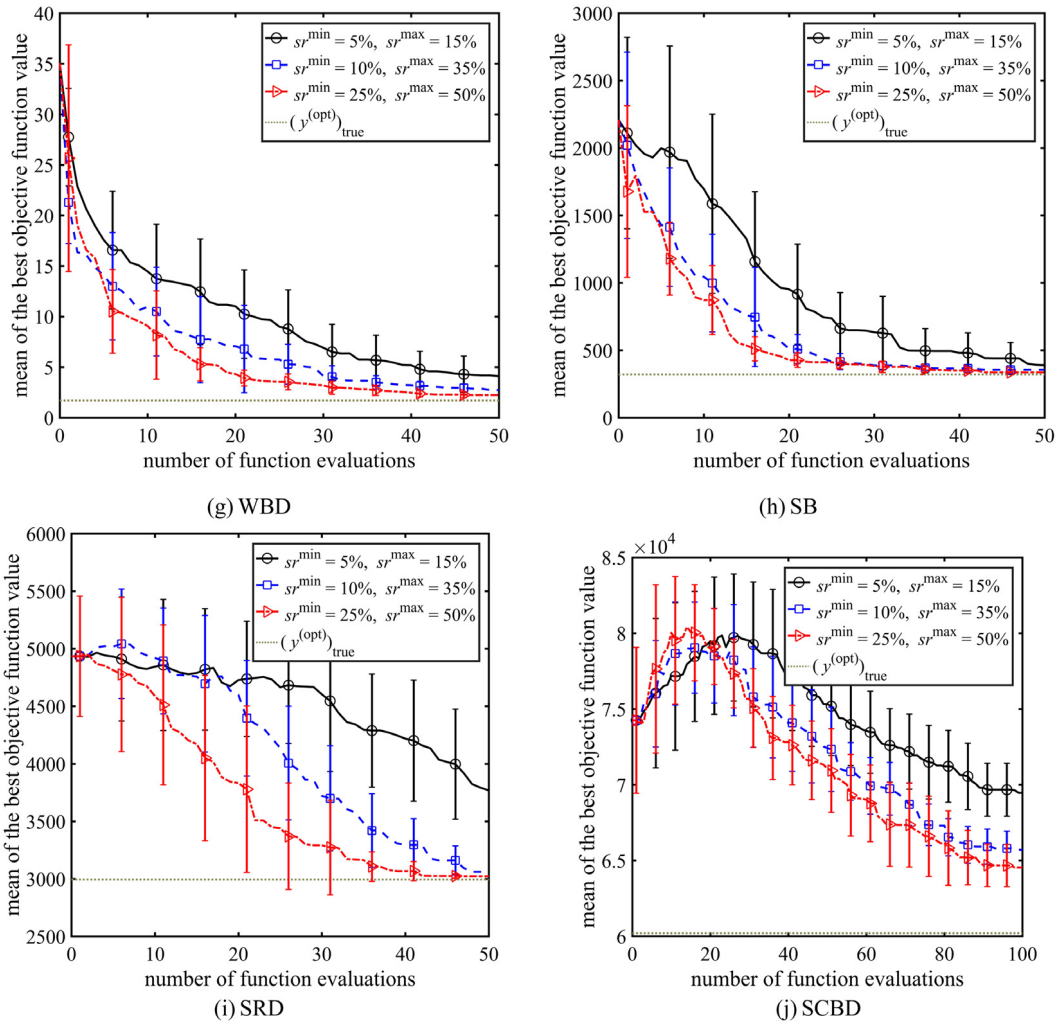


Fig. 4. (*continued*).

7. Conclusion

This paper introduced the Kriging-based Adaptive Space Reduction Algorithm, (KASRA), which is an extension of the Efficient Global Optimization (EGO) methodology. KASRA is designed to handle the computationally expensive black-box optimization problems with single objective function subject to inequality constraints; these problems frequently appear in the real-world engineering design where only a relatively limited number of function evaluations (simulations) can be performed.

KASRA is a two-phase surrogate-based approach that uses Kriging models for the objective and each of the constraint functions. The first phase aims at identifying a promising region of the search space that involves global optimum. The key elements of this phase are: (1) finding a feasible point if all initial points are infeasible; (2) adaptively reducing the design space based on the activity of the decision variables; (3) searching the subspaces based on the weighed constrained expected improvement criterion which dynamically adjusts a reasonable weight ratio of the exploration and exploitation; (4) expanding sub-spaces to decrease the risk of missing the promising region and avoid getting trapped in a local valley.

The second phase of the algorithm is a multi-stage local optimization program in the vicinity of the current best solution to improve it. The second phase gets started when the best point of the first phase adequately does not approach to the true global optimum. The budget needed to evaluate the functions should be separated between these two phases so that the local region (adjacent zone) of global optimum with high probability gets trapped in the first phase. Otherwise, the implementation of the second phase is not useful, and only the accuracy of the local solution will be increased.

The performance of KASRA is compared and investigated with the alternative methods on 32 widely used test problems whose dimensions range from 2–20 with 1–38 inequality constraints. The alternatives have been divided into two sets: (1) COBRA-Local, COBRA-Global, Extended ConstrLMSRBF, SDPEN, and NOMADm-DACE; (2) SCGOSR, RBFCGOSR, SCGO, MSSR, MS, and MSRBF. In the first set, the algorithms are compared in terms of the number of function evaluations required to find a feasible region and to pass a given target value, therefore, all initial points are deliberately considered infeasible. In the second set of comparisons, the convergence rate and efficiency are assessed. Moreover, complementary experiments have been considered to indicate the extensive applicability of KASRA and its capability to find the more exact solution.

The numerical results express that KASRA is quite robust to achieve a feasible region and potentially has a significant capacity to reach the target value. Besides, global convergence and optimization efficiency of the proposed method generally outperforms several existing algorithms on most of the test problems. KASRA also shows its strong robustness to find a more accurate

solution on the complementary experiments. Although KASRA is very promising surrogate-based method for high-multimodal expensive black-box constrained optimization problems, especially for engineering applications, further works are required to generally determine its user-specified parameters such as the size of minimum and maximum space reduction ($sr^{\min}$ and $sr^{\max}$). Besides, it is needed to extend the full potential of this method for large scale optimization problems and the problems with multi-objective functions dealing with equality constraints.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106154.

CRediT authorship contribution statement

Hossein Akbari: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Afshin Kazerooni:** Resources, Writing - original draft, Writing - review & editing, Supervision, Project administration.

Acknowledgment

The authors would like to thank the computing center of SRTTU University for sincere cooperation in providing facilities.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2020.106154.

References

[1] S. Shan, G.G. Wang, Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions, Struct. Multidiscip. Optim. 41 (2) (2010) 219–241, http://dx.doi.org/10.1007/s00158-009-0420-2.

[2] Ya. D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget, Sci. Rep. 8 (1) (2018) 1–9, http://dx.doi.org/10.1038/s41598-017-18940-4.

[3] L. Leifsson, S. Koziel, Simulation-Driven Aerodynamic Design using Variable-Fidelity Models, Imperial College Press, 2015, http://dx.doi.org/10.1142/9781783266296_fmatter.

[4] S.S. Garud, I.A. Karimi, M. Kraft, Design of computer experiments: A review, Comput. Chem. Eng. 106 (2017) 71–95, http://dx.doi.org/10.1016/j.compchemeng.2017.05.010.

[5] V.K. Ky, C. D'Ambrosio, Y. Hamadi, L. Liberti, Surrogate-based methods for black-box optimization, Int. Trans. Oper. Res. 24 (3) (2016) 393–424, http://dx.doi.org/10.1111/itor.12292.

[6] J.P.C. Kleijnen, Regression and Kriging metamodels with their experimental designs in simulation: A review, European J. Oper. Res. 256 (1) (2017) 1–16, http://dx.doi.org/10.1016/j.ejor.2016.06.041.

[7] T.B. Beielstein, M. Zaefferer, Model-based methods for continuous and discrete global optimization, Appl. Soft Comput. 55 (2017) 154–167, http://dx.doi.org/10.1016/j.asoc.2017.01.039.

[8] A. Bhosekar, M. Ierapetritou, Advances in surrogate based modeling, feasibility analysis, and optimization: A review, Comput. Chem. Eng. 108 (2018) 250–267, http://dx.doi.org/10.1016/j.compchemeng.2017.09.017.

[9] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. 13 (4) (1998) 455–492, http://dx.doi.org/10.1023/A:1008306431147.

[10] A.I.J. Forrester, A.J. Keane, Recent advances in surrogate-based optimization, Prog. Aerosp. Sci. 45 (1–3) (2009) 50–79, http://dx.doi.org/10.1016/j.paerosci.2008.11.001.

[11] H. Liu, Y.S. Ong, J. Cai, A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design, Struct. Multidiscip. Optim. 57 (1) (2018) 393–416, http://dx.doi.org/10.1007/s00158-017-1739-8.

[12] L. Pronzato, W.G. Müller, Design of computer experiments: space filling and beyond, Stat. Comput. 22 (3) (2012) 681–701, http://dx.doi.org/10.1007/s11222-011-9242-3.

[13] A.I.J. Forrester, A. Sóbester, A.J. Keane, Engineering Design Via Surrogate Modelling: A Practical Guide, Wiley, 2008, http://dx.doi.org/10.2514/4.479557.

[14] J.M. Parr, A.I.J. Forrester, A.J. Keane, C.M.E. Holden, Enhancing infill sampling criteria for surrogate-based constrained optimization, J. Comput. Methods Sci. Eng. 12 (1–2) (2012) 25–45, http://dx.doi.org/10.3233/JCM-2012-0402.

[15] J.M. Parr, A.J. Keane, A.I.J. Forrester, C.M.E. Holden, Infill sampling criteria for surrogate-based optimization with constraint handling, Eng. Optim. 44 (10) (2012) 1147–1166, http://dx.doi.org/10.1080/0305215X.2011.637556.

[16] H. Liu, J. Cai, Y.S. Ong, An adaptive sampling approach for Kriging metamodeling by maximizing expected prediction error, Comput. Chem. Eng. 106 (2017) 171–182, http://dx.doi.org/10.1016/j.compchemeng.2017.05.025.

[17] R.G. Regis, C.A. Shoemaker, Improved strategies for radial basis function methods for global optimization, J. Global Optim. 37 (1) (2007) 113–135, http://dx.doi.org/10.1007/s10898-006-9040-1.

[18] R.G. Regis, Trust regions in Kriging-based optimization with expected improvement, Eng. Optim. 48 (6) (2015) 1037–1059, http://dx.doi.org/10.1080/0305215X.2015.1082350.

[19] J. Arora, Introduction To Optimum Design, fourth ed., Academic Press, 2017, http://dx.doi.org/10.1016/C2013-0-15344-5.

[20] C. Barron, S. Gomez, The Exponential Tunneling Method, Technical Report, IIMAS-UNAM, (1)(3), 1991, pp. 1–23.

[21] L. Castellanos, S. Gomez, A New Implementation of the Tunneling Methods for Bound Constrained Global Optimization, IIMAS-UNAM, (10)(59), 2000, pp. 1–18.

[22] G. Wu, W. Pedrycz, P.N. Suganthan, H. Li, Using variable reduction strategy to accelerate evolutionary optimization, Appl. Soft Comput. 61 (2017) 283–293, http://dx.doi.org/10.1016/j.asoc.2017.08.012.

[23] G.G. Wang, Z. Dong, P. Aitchison, Adaptive response surface method – A global optimization scheme for approximation-based design problems, Eng. Optim. 33 (6) (2001) 707–733.

[24] G.G. Wang, T. Simpson, Fuzzy clustering based hierarchical metamodeling for design space reduction and optimization, Eng. Optim. 36 (3) (2004) 313–335, http://dx.doi.org/10.1080/03052150310001639911.

[25] G.G. Wang, S. Shan, Design space reduction for multi-objective optimization and robust design optimization problems, SAE Trans. J. Mater. Manuf. 113 (5) (2004) 101–110, http://dx.doi.org/10.4271/2004-01-0240.

[26] V.V. de Melo, A.C.B. Delbem, D.L.P. Junior, F.M. Federson, Improving global numerical optimization using a search-space reduction algorithm, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, England, 2007, pp. 1195–1202, http://dx.doi.org/10.1145/1276958.1277191.

[27] A. Younis, R. Xu, Z. Dong, Approximated unimodal region elimination-based global optimisation method for engineering design, Int. J. Prod. Dev. 9 (1/2/3) (2009) 164–187, http://dx.doi.org/10.1504/IJPD.2009.026178.

[28] A. Younis, Z. Dong, Metamodelling and search using space exploration and unimodal region elimination for design optimization, Eng. Optim. 42 (6) (2010) 517–533, http://dx.doi.org/10.1080/03052150903325540.

[29] J. Gu, G.Y. Li, Z. Dong, Hybrid and adaptive meta-model-based global optimization, Eng. Optim. 44 (1) (2012) 87–104, http://dx.doi.org/10.1080/0305215X.2011.564768.

[30] T. Long, D. Wu, X. Guo, G.G. Wang, L. Liu, Efficient adaptive response surface method using intelligent space exploration strategy, Struct. Multidiscip. Optim. 51 (6) (2015) 1335–1362, http://dx.doi.org/10.1007/s00158-014-1219-3.

[31] H. Liu, S. Xu, X. Wang, Sequential sampling designs based on space reduction, Eng. Optim. 47 (7) (2015) 867–884, http://dx.doi.org/10.1080/0305215X.2014.928816.

[32] H. Dong, B. Song, Z. Dong, P. Wang, Multi-start space reduction (MSSR) surrogate-based global optimization method, Struct. Multidiscip. Optim. 54 (4) (2016) 907–926, http://dx.doi.org/10.1007/s00158-016-1450-1.

[33] H. Dong, B. Song, P. Wang, Z. Dong, Hybrid surrogate-based optimization using space reduction (HSOSR) for expensive black-box functions, Appl. Soft Comput. 64 (2018) 641–655, http://dx.doi.org/10.1016/j.asoc.2017.12.046.

[34] H. Dong, B. Song, Z. Dong, P. Wang, SCGOSR: Surrogate-based constrained global optimization using space reduction, Appl. Soft Comput. 65 (2018) 462–477, http://dx.doi.org/10.1016/j.asoc.2018.01.041.

[35] D.R. Jones, A taxonomy of global optimization methods, J. Global Optim. 21 (4) (2001) 345–383, http://dx.doi.org/10.1023/A:1012771025575.
[36] D.G. Krige, A statistical approaches to some basic mine valuation problems on the witwatersrand, J. Chem. Metall. Min. Soc. S. Afr. 52 (6) (1951) 119–139.
[37] G. Matheron, Principles of geostatistics, Econ. Geol. 58 (8) (1963) 1246–1266, http://dx.doi.org/10.2113/gsecongeo.58.8.1246.
[38] J. Sacks, W.J. Welch, T.J. Mitchell, H.P. Wynn, Design and analysis of computer experiments, Statist. Sci. 4 (4) (1989) 409–423.
[39] A. Sóbester, S.J. Leary, A.J. Keane, On the design of optimization strategies based on global response surface approximation models, J. Global Optim. 33 (1) (2005) 31–59, http://dx.doi.org/10.1007/s10898-004-6733-1.
[40] M. Schonlau, Computer Experiments and Global Optimization (Ph.D. thesis), University of Waterloo, 1997.
[41] E. Vazquez, J. Bect, Convergence properties of the expected improvement algorithm, J. Statist. Plann. Inference 140 (11) (2010) 3088–3095, http://dx.doi.org/10.1016/j.jspi.2010.04.018.
[42] H. Akbari, A. Kazerooni, Dataset for: KASRA: A Kriging-based adaptive space reduction algorithm for global optimization of computationally expensive black-box constrained problems, Mendeley data, v1, 2019. http://dx.doi.org/10.17632/bnv8fn48g6.1.
[43] R.G. Regis, Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions, Comput. Oper. Res. 38 (5) (2011) 837–853, http://dx.doi.org/10.1016/j.cor.2010.09.013.
[44] R.G. Regis, Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points, Eng. Optim. 46 (2) (2014) 218–243, http://dx.doi.org/10.1080/0305215X.2013.765000.
[45] G. Liuzzi, S. Lucidi, M. Sciandrone, Sequential penalty derivative-free methods for nonlinear constrained optimization, SIAM J. Optim. 20 (5) (2010) 2614–2635, http://dx.doi.org/10.1137/090750639.
[46] S. Le Digabel, Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm, ACM Trans. Math. Software 37 (4) (2011) 44:1–44:15.
[47] M.A. Abramson, NOMADm Version 4.6 User'S Guide, Department of Mathematics and Statistics, Air Force Institute of Technology, 2007.
[48] S.N. Lophaven, H.B. Nielsen, J. Søndergaard, DACE: A Matlab Kriging Toolbox, Version 2.0, Technical Report Informatics and Mathematical Modelling, (IMM-TR)(12), Technical University of Denmark, 2002, pp. 1–34.