

Efficient adaptive response surface method using intelligent space exploration strategy

Teng Long · Di Wu · Xiaosong Guo · G. Gary Wang · Li Liu

Received: 12 May 2014 / Revised: 2 November 2014 / Accepted: 3 December 2014
© Springer-Verlag Berlin Heidelberg 2015

Abstract This article presents a novel intelligent space exploration strategy (ISES), which is then integrated with the adaptive response surface method (ARSM) for higher global optimization efficiency. ISES consists of two novel elements for space reduction and sequential sampling: i) Significant design space (SDS) identification algorithm, which is developed to identify the promising design space and balance local exploitation and global exploration during the search, and ii) An iterative maximin sequential Latin hypercube design (LHD) sampling scheme and tailored termination criteria. Moreover, an adaptive penalty method is developed for handling expensive constraints. The new global optimization strategy, notated as ARSM-ISES, is then tested with numerical benchmark problems on optimization efficiency, global convergence, robustness, and algorithm execution overhead. Comparative results show that ARSM-ISES not only outperforms the original ARSM and IARSM, in general it also converges to better optima with fewer function evaluations and less algorithm execution time as compared to state-of-the-art metamodel-based design optimization algorithms including MPS, EGO, and MSEG. For high dimensional (HD) problems, ARSM-ISES shows promises as it performs better

on chosen test problems than TR-MPS, which is especially designed for solving HD problems. ARSM-ISES is then applied to the optimal design of a lifting surface of hypersonic flight vehicles. Finally, main features and limitations of the proposed algorithm are discussed.

Keywords Response surface method · Metamodel-based design optimization · Global optimization · Intelligent space exploration strategy · Significant design space · Sequential sampling · Constrained optimization

1 Introduction

Optimization technologies have been more and more extensively applied in engineering design problems to improve design quality and shorten design cycle. Nowadays, most modern engineering design problems involve computationally expensive functions (e.g., computational fluid dynamics models, finite element analysis models, etc.). In the past two decades, metamodeling techniques have become attractive to reduce computational cost for expensive function based optimizations. Computationally inexpensive metamodels are constructed to approximate and take place of expensive functions in simulation-based optimizations. Plenty of optimization methodologies assisted by metamodeling technologies, called metamodel-based design optimization (MBDO) (Wang and Shan 2007) or surrogate-based analysis and optimization (SBAO) (Queipo et al. 2005), have been developed in recent years.

Generally, metamodels are constructed based on a set of known samples produced by various sampling methods. In order to obtain more information of expensive simulations in the design space with minimum number of samples, sampling

T. Long · L. Liu

Key Laboratory of Dynamics and Control of Flight Vehicle, Ministry of Education, Beijing Institute of Technology Beijing, Beijing 100081, China

T. Long (✉) · D. Wu · X. Guo · L. Liu

Aircraft Synthesis Design Group, School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China
e-mail: bitryu@gmail.com

T. Long

e-mail: tenglong@bit.edu.cn

G. G. Wang

School of Mechatronic Systems Engineering, Simon Fraser University, Surrey, BC V3T 0A3, Canada

methods are developed to satisfy the space-filling and projective uniform properties (Simpson et al. 2004). To produce high quality samples, the promising optimal Latin Hypercube Design (LHD) sampling methods have been developed based on several optimal criteria such as Maximin distance criterion, Entropy criterion, and Centered L_2 discrepancy criterion (Jin et al. 2005). For instance, Ye et al. (2000) employed the columnwise-pairwise (CP) algorithm for constructing optimal symmetrical LHDs. Jin et al. (2005) presented an efficient optimal LHD sampling method using an enhanced stochastic evolutionary algorithm. Viana et al. (2010a) presented TPLHD to obtain near optimal LHDs without going through the expensive optimization process. Another optimal LHD sampling method using successive local enumeration (SLE) was proposed by Zhu et al. (2012a).

Thus far, a number of metamodeling methods have been developed and successfully applied in engineering design problems (Wang and Shan 2007; Simpson et al. 2008; Shan and Wang 2010). Typical metamodels include the polynomial response surface model (RSM), Kriging model, radial basis function (RBF), support vector machine (SVM), etc. Comparison of performance among different metamodels was presented in the references (Simpson et al. 1998; Jin et al. 2001; Long 2009). Although studies have been conducted to enhance approximation capability of metamodels, it is still rather difficult to precisely approximate computation-intensive models using sparsely scattered samples in the entire large design space, especially for high dimensional (HD) and high nonlinear problems.

Recently, adaptive or sequential metamodeling techniques are used for MBDO studies and applications. The generic execution process of most adaptive MBDO methodologies can be summarized as below. Metamodels employed in optimization are gradually updated by using sequential bias samples to improve the approximation accuracy in the promising regions where the true global optimum probably exists. The metamodels then lead the search to the global optimum. Different metamodel management mechanisms are proposed for promising region identification, space reduction, sequential bias sampling, and metamodel updating, which are essential elements of MBDO methods that determine their overall performance. In the past two decades, a number of studies have been reported on MBDO algorithms. A brief review of adaptive MBDO algorithms is presented as follows.

The primitive adaptive MBDO methodology adds a single potential optimum from approximation-based optimization at each iteration for updating metamodels until convergence. Lewis (1996) and Alexandrov et al. (1998) introduced a trust region framework to manage approximations with proved convergence property. Gano et al. (2006) proposed a metamodel updating management scheme using trust region ratio (TR-MUMS) to periodically update a Kriging scaling model, which decreased the computational expense of variable fidelity optimization. Pérez et al. (2002) introduced a trust region-based

adaptive experimental design (AED) strategy to reduce the number of required sample points to maintain the accuracy of local approximation model during the optimization procedure. However, TR-MUMS and AED need gradient information of the objective and constraints to construct the scaling model.

Efficient global optimization (EGO) algorithm was reported by Jones et al. (1998). In EGO, sequential bias samples are designated to update the Kriging metamodel in terms of the expected improvement (EI) criterion. Sasena (2002) inspected several infilling sampling criteria for EGO and proposed super-EGO to enhance the flexibility and efficiency for constrained problems (Sasena et al. 2005). Viana et al. (2010b, 2013) introduced a multiple surrogate efficient global optimization methodology (MSEGO) based on the notion of EGO, which increased a few samples instead of only one sample at each iteration, based on information from various metamodels.

In another direction, Wang et al. (2004) presented the mode pursuing sampling (MPS) method to produce more samples towards the global optimum using cumulative probability estimation according to the RBF metamodel of the objective function. A comparative study between MPS and GA is presented by Duan et al. (2009). Sharif et al. (2008) then developed an extended version of MPS for discrete variable problems.

Fuzzy clustering was also applied to locate the promising regions. In this area, Wang and Simpson (2004) presented a hierarchical metamodel-based global optimization methods using fuzzy clustering for design space reduction. In this method, a global metamodel using RSM or Kriging is first built to produce plenty of inexpensive samples for clustering and space contraction, and then expensive samples are generated in an irreducible space to build a local Kriging metamodel for seeking the global optimum. Zhu et al. (2012b) proposed an adaptive RBF metamodel-based global optimization methodology using fuzzy c-mean clustering method, and then applied this method to optimize the aerodynamic-thermal-structural coupled performance of the lifting surface of a hypersonic aircraft (Zhu et al. 2012c). Li et al. (2013) developed a more elaborated design space reduction method based on fuzzy clustering using pseudo reduction processes to enhance global exploration performance.

In addition, according to the investigation on RSM for structural optimization, Roux et al. (1998) pointed out that the location and size of the smaller region of interest largely influence the fitting quality of RSM and corresponding optimal solution compared with other factors including the number of construction samples and selection of the best regression equation. Hence, for RSM-based global optimization algorithms, it is crucial to effectively and efficiently identify the region of interest that contains the actual global optimum. Wang et al. (2001) proposed the adaptive response surface method (ARSM) using cutting planes for space reduction. Furthermore, an improved ARSM (IARSM) inherited LHD samples to save the computational cost (Wang 2003). Based

on the standard ARSM, some further enhancement studies and engineering applications were reported. Wang et al. (2008) developed ARSM based on particle swarm optimization intelligent sampling method to optimize sheet metal forming process. To overcome the limitations of the cutting plane approach, a pseudo-ARSM was presented by Panayi et al. (2009), which employed iteratively the reweighted least-square method to enhance the quality of RSM for piston skirt profiles optimization. Long et al. (2012a) proposed a significant design space (SDS) approach to develop an enhanced adaptive response surface method (EARSM), which was applied to the aero-structural coupled optimization of a high aspect ratio wing (Long et al. 2012b).

For more detailed information about the state-of-the-art of adaptive MBDO algorithms, some comprehensive literature reviews are highly recommended (Queipo et al. 2005; Simpson et al. 2008; Forrester and Keane 2009; Shan and Wang 2010).

Recently, some work has been carried out to handle expensive constraints. For example, Kazemi et al. (2011) developed a constraint importance mode pursuing sampling algorithm (CiMPS) for problems with expensive objective and constraint functions. Regis (2011) proposed a constrained local metric stochastic RBF (ConstrLMSRBF) for optimizations involving expensive objective and constraints, where multiple RBF metamodels were built for expensive objective and constraints to select promising points according to the feasibility, optimality, and a distance-based local metrics. A constrained optimization algorithm by radial basis function interpolation (COBRA) and an extension of ConstrLMSRBF independent of feasible initial samples were then developed (Regis 2014). Parr et al. (2012) constructed Pareto sets according to the expected improvement and probability of feasibility to identify infill samples for expensive constrained problems. Bichon et al. (2013) developed a constrained EGO using augmented Lagrangian method for reliability-based design optimization. Although some progresses have been achieved, the development of effective mechanisms for general MBDO algorithms to handle expensive constraints is still an important challenge.

In this paper, a novel and efficient ARSM algorithm using our proposed intelligent space exploration strategy (ISES), notated as ARSM-ISES, is developed. ISES package is composed of two novel algorithms, including SDS identification algorithm, as well as sequential sampling scheme and the termination criteria.

The rest of this article is arranged as follows. Section 2 presents a short review of ARSMs to summarize the features and limitations of ARSM. In Section 3, development of ARSM-ISES is detailed, including iterative sequential

maximin LHD sampling scheme for improving samples' quality, new SDS identification algorithm for design space reduction, specific termination criteria and an adaptive penalty function method for dealing with expensive constraints. In Section 4, some numerical benchmark problems are employed to test the performance of ARSM-ISES through comparison with ARSM, IARSM, and other well-known MBDO methodologies. And then ARSM-ISES is applied to minimize the weight of a hypersonic flight vehicle's lifting surface subject to flutter speed constraint to show its practicability for real-world engineering problems involving expensive constraints. Features and limitations of the proposed method are discussed in Section 5. Finally, conclusions and future work are given.

2 Review of ARSMs

ARSM is based on a second order RSM, described below

$$\hat{y} = \beta_0 + \sum_{i=1}^{n_v} \beta_i x_i + \sum_{i=1}^{n_v} \beta_{ii} x_i^2 + \sum_{i=1}^{n_v-1} \sum_{j=i+1}^{n_v} \beta_{ij} x_i x_j \quad (1)$$

Where β are the coefficients and n_v is the number of variables. A threshold objective value is chosen to function as a "cutting plane" with which a reduced design space is obtained through calling two global optimization processes to find the lower and upper bounds for each variable. The metamodeling and space reduction process continues till convergence. Although ARSM possesses some merits for expensive global optimizations, such as high efficiency and capability of global optimization (Wang 2003), the cutting plane approach causes major limitations of ARSM on optimization efficiency and global exploration capability. In addition, ARSM is not capable of handling optimizations with expensive constraints.

First, the cutting plane approach invokes $2n_v$ auxiliary global optimization processes to identify boundaries of the reduced design space. Furthermore, the approach of choosing a threshold to determine the cutting plane is empirical and *ad hoc*.

Secondly, for multimodal cases, the fitting quality of RSMs at initial iterations is generally not good so that the actual global optimal solution very likely locates outside the reduced design space determined by the cutting plane. The eliminated regions containing the true global optimum are never explored at the subsequent iterations. Hence, ARSM has a theoretical pitfall of missing the global optimum. Figure 1 depicts this phenomenon of ARSM missing the global optimum on a one-dimensional example. In Fig. 1, the pentagram indicates the true global optimum of the function, and the

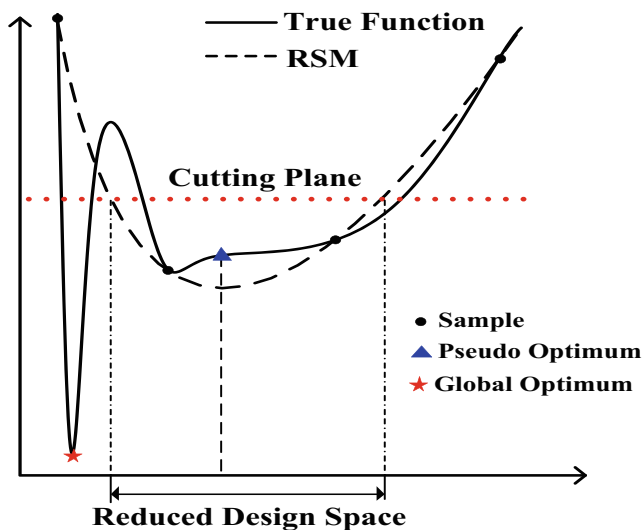


Fig. 1 Illustration of the cutting plane approach missing the global optimum

triangle is the pseudo-optimum obtained from the current RSM. Sample points for fitting the RSM are represented by solid circles. For this problem, the cutting plane is indicated by the dotted line. It is clear that the true global optimum is eliminated from the reduced design space and eventually missed in this ARSM optimization.

Additionally, the studies on ARSM (Wang 2003; Wang et al. 2001) are under an assumption that all the constraints are computationally inexpensive. However, lots of real-world engineering optimization problems are subject to expensive constraints. Hence, existing ARSM and IARSM cannot be directly applied to such problems with both expensive objective and constraints.

From the previous discussion, it is evident that the cutting plane approach for space reduction is the problem. Though several ARSM variants (e.g., IARSM and pARSM) have been developed to improve the efficiency, most existing variants of ARSM inherit the cutting plane approach for space reduction. In order to further improve the performance of ARSM, this paper presents a novel ISES for space reduction and RSM update. Besides, development of an adaptive penalty function method enables ARSM-ISES to be applicable to expensive constrained optimizations.

3 ARSM using intelligent space exploration strategy

In this section, an efficient adaptive response surface method using intelligent space exploration strategy,

namely ARSM-ISES, is presented in detail. At first, the overall procedure of ARSM-ISES is introduced. Then the novel algorithms in ISESE package, including the new SDS identification algorithm, and the sequential sampling scheme and the tailored termination criteria, are described. The adaptive penalty function method is also given for handling expensive constraints.

3.1 Overall procedure of ARSM-ISES

Similar to the standard ARSM, Fig. 2 illustrates the flowchart of ARSM-ISES with highlighted new modules developed in this work. The cutting plane approach for ARSM is replaced by our developed ISES. More detailed description of ISES will be presented in following sections.

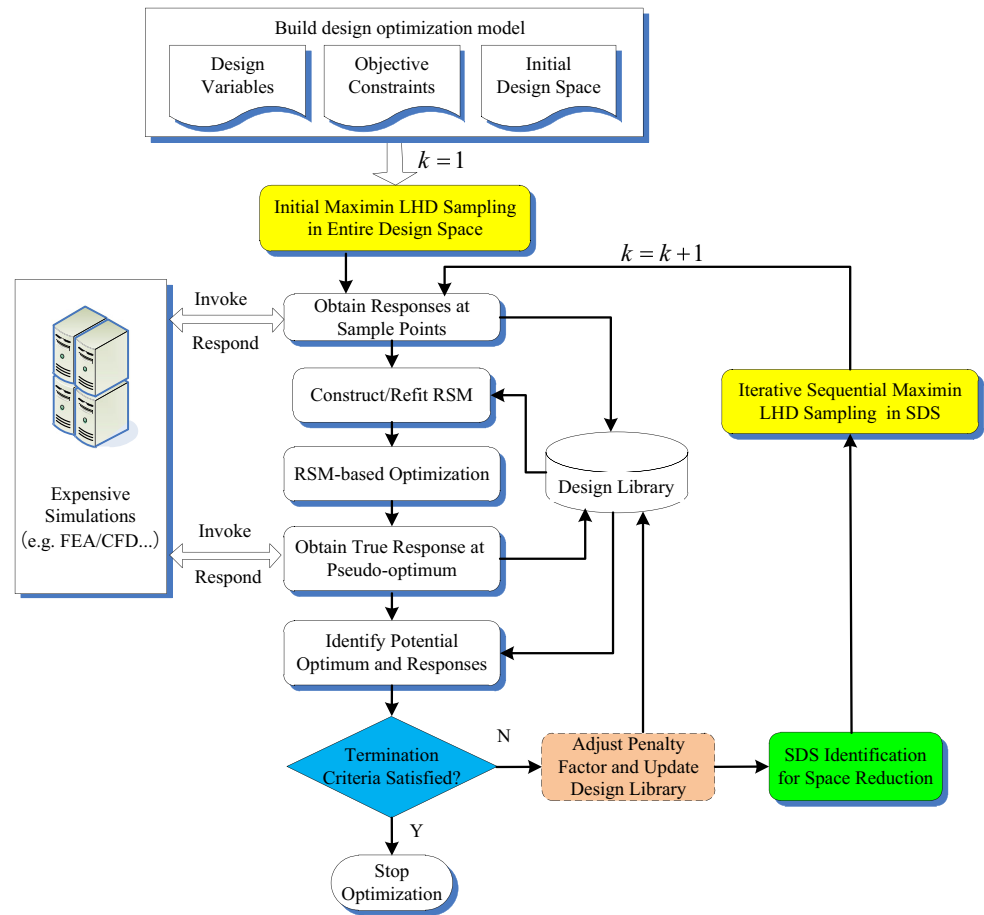
According to Fig. 2, the overall iterative procedure of ARSM-ISES is given as follows.

Step 1. Build the optimization model. A common engineering optimization problem can be formulated in (2)

$$\begin{aligned} & \text{find } \mathbf{x} = [x_1, x_2, \dots, x_{n_v}]^T \\ & \min f(\mathbf{x}) \\ & \text{s.t. } g_j(\mathbf{x}) \leq 0 \quad (j = 1, 2, \dots, m) \\ & \quad x_i^{lb} \leq x_i \leq x_i^{ub} \quad (i = 1, 2, \dots, n_v) \end{aligned} \quad (2)$$

where $\mathbf{x} = [x_1, x_2, \dots, x_{n_v}]^T$ is a vector of design variables; $f(\mathbf{x})$ is the objective function and $g_j(\mathbf{x})$ is the j -th constraint. The initial design space is defined by the upper and lower bounds of design variables. In this work, $f(\mathbf{x})$ is always evaluated based on expensive simulations, while $g_j(\mathbf{x})$ may be an expensive or cheap constraint for different problems. If constraints are inexpensive, then no approximation or special treatment of constraints is needed. But in case of using expensive constraints, a specific constraint handling mechanism is required to obtain a feasible optimal solution with limited evaluations of expensive constraints. Besides, we have to set tuning parameters to configure ARSM-ISES. Set the iteration counter $k=1$ and start ARSM-ISES.

Step 2. Sample points are generated through a maximin LHD sampling method in the initial design space. For the sake of improving RSM metamodel's quality and acquire more information in unexplored regions, samples with good space-filling and projective properties are preferred. Considering sampling quality and efficiency, the maximin LHD sampling method of

Fig. 2 Overall flowchart of ARSM-ISES

MATLAB *lhsdesign* function using ‘maximin’ criterion with 100 iterations is adopted to collect initial samples. The number of initial sample points is equal to two times of the number of RSM’s unknown coefficients, namely, $n_{is} = 2p = (n_v + 1)(n_v + 2)$.

- Step 3. Expensive simulations are invoked to evaluate responses at the initial or newly-added samples. The responses include both objective function and expensive constraints. All the sample points and corresponding true responses are stored in the design library for later use.
- Step 4. RSM is constructed or refitted with the samples and their responses in current design space. If m_1 expensive constraints are involved, we construct a RSM to approximate a merit function instead of building several RSMs for objective and constraints separately. Merit function $\phi(\mathbf{x})$ in (3) comprises of the original objective $f(\mathbf{x})$ to indicate optimality and an additive penalty term to manifest feasibility. The proposed adaptive penalty

function method for handling expensive constraints will be explained later.

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \lambda \sum_{j=1}^{m_1} \max(g_j(\mathbf{x}), 0) \quad (3)$$

Note that we do not claim that for general MBDO algorithms the use of a single metamodel for the merit function is superior to building separate metamodels for objective and constraints. Because ISES identifies the promising regions in terms of the objective function information, the model uncertainty of separate RSMs for expensive constraints, especially at the initial iterations, may cause the SQP sub-optimization in Step 5 to yield an infeasible pseudo-optimum far away from the boundaries of the feasible domains, and even make ARSM-ISES abnormally converge to an infeasible solution. In this work, using a single

RSM for the merit function is preferable to prevent ARSM-ISES from being trapped in infeasible domains.

- Step 5. RSM-based sub-optimization in the current design space is carried out to obtain a pseudo-optimum $\mathbf{x}_{ps}^{*(k)}$. Note that at initial iterations, the pseudo-optimum may not be the true optimal and even not the best point in the current design library due to poor fitting quality of RSM. Since the second order RSM theoretically has only one optimum, the sequential quadratic programming (SQP) is employed to perform the RSM-based search. Although the identical SQP algorithm is used (e.g., MATLAB *fmincon* function), there are three circumstances of SQP sub-optimization in terms of the computational cost of constraints. If no expensive constraint appears, SQP sub-optimization is performed to minimize the original objective subject to the true cheap constraints and boundary constraints. If all the constraints are computationally expensive, SQP sub-optimization is conducted to minimize the merit function, comprising of the original objective and expensive constraints, subject to the boundary constraints. If both expensive and cheap constraints are involved, only the expensive constraints are used to formulate the merit function, and the cheap constraints and boundary constraints are directly invoked in SQP sub-optimizations. Hence, SQP sub-optimizations in this step gradually improve and eventually ensure the feasibility defined by the expensive constraints by using the merit function as the objective.
- Step 6. The computationally intensive simulations are called to evaluate the true responses at the pseudo-optimum, which is then recorded in the design library.
- Step 7. The best design point with the lowest objective or merit function value in current design library is elected as the potential optimum $\mathbf{x}_{po}^{*(k)}$. Although current $\mathbf{x}_{po}^{*(k)}$ may also differ from the true global optimum, it is recognized to be the closest to our desired solution. At initial iterations, the poor accuracy of RSM in a large design space makes the potential optimum be different from the pseudo-optimum. However, as optimization proceeds, the accuracy of RSM will be gradually improved in more
- limited design space, and $\mathbf{x}_{ps}^{*(k)}$ is expected to be identical to $\mathbf{x}_{po}^{*(k)}$.
- Step 8. If the termination criteria are satisfied, ARSM-ISES process stops and outputs the last potential global optimum $\mathbf{x}_{po}^{*(k)}$ as the final result. Otherwise, the process jumps to Step 9 and continues. An effective assembly of termination criteria will be detailed in Section 3.5.
- Step 9. If expensive constraints are involved, an adaptive penalty function method is used to adjust the penalty factor according to the present information, and then update all the merit function values using the renewed penalty factor and original responses of $f(\mathbf{x})$ and $g_j(\mathbf{x})$. If no expensive constraint appears, this step is skipped.
- Step 10. SDS for the next iteration is determined by using the new SDS identification algorithm. This step is the most important and interesting part of the proposed method. The new algorithm to identify SDS will be described in Section 3.3.
- Step 11. Inside the renewed SDS for the next iteration, a set of sequential samples are collected using an iterative maximin sequential LHD sampling scheme (IMS-LHD). IMS-LHD inherits the previously generated samples located inside the new ISES to save the computational cost. Thus, the number of the new sample points satisfies $0 \leq n_{new} \leq p$. Then, the iteration counter increases, $k=k+1$, and the process jumps to Step 3 and continues.

3.2 Iterative Maximin sequential LHD sampling scheme

In general, sequential sampling has to balance the local exploitation and global exploration. In this work, such a balance is achieved by collaboration of SDS identification algorithm, sequential sampling scheme, and the tailored termination criteria. Also, evenly distributed samples in sequential sampling phases are beneficial for exploring a promising design space. Good space-filling property is also desired to prevent the sequential samples from getting too crowded for reducing the risk of encountering singularity when using some nearly coincident samples to refit a RSM metamodel.

To generate sequential samples far from the existing ones in a reduced design space, an iterative maximin sequential LHD sampling scheme is developed, notated as IMS-LHD, whose process is summarized in

Algorithm 1. The steps of IMS-LHD algorithm are presented as follows.

Algorithm 1 IMS-LHD

Input: the design library at the current iteration $[S, F]$; number of design variables n_v ; maximum number of iterations for choosing maximin samples MAX_ITER; current design space (i.e., SDS) V_{SDS} ; objective tolerance ε ; and minimum design space factor σ .

Output: a set of new samples S_{new} ; a set of samples in the current SDS S_{SDS} ; renewed SDS V_{SDS} ; and renewed minimum design space factor σ .

Begin

```

1   $[S_{ex}, F_{ex}, n_{ex}] \leftarrow \text{InheritExistingSamples}(V_{SDS}, S, n_v)$ ;
2   $n_{new} = p - n_{ex}$ ;
3  If  $n_{new} > 0$  then
4     $[S_{new}, d_{mxm}] \leftarrow \text{GenerateNewSamples}(x_{lb}^k, x_{ub}^k, S, F, \text{criteria}, \text{MAX\_ITER})$ ;
5    If  $d_{mxm} \leq \varepsilon$  then
6       $[V_{SDS}, \sigma] \leftarrow \text{ExpandSDS}(V_{SDS}, \sigma)$ ;
7      jump to 1
8    end
9     $S_{SDS} \leftarrow S_{ex} \cup S_{new}$ ;
10 else
11    $[S_{SDS}] \leftarrow \text{SelectPromisingSamples}(S_{ex}, F_{ex}, p)$ ;
12    $S_{new} = \text{NULL}$ ;
13 end
14 return  $[S_{new}, S_{SDS}, V_{SDS}, \sigma]$ ;

```

- Step 1. (Inherit existing samples, lines 1–2): All existing samples located inside the current design space are inherited for updating the RSM. If $n_{new} > 0$, n_{new} samples will be generated in subsequent steps. While, if $n_{new} \leq 0$, no new sample to be generated.
- Step 2. (Generate new samples, lines 3–4): To improve the space-filling property of newly-added and existing samples, an iterative election process in terms of maximin distance criterion d_{mxm} , as shown in (4) is performed to pick the best sample set found within MAX_ITER iterations. At first, $10p$ candidate samples are produced by *lhsdesign* function, and then at each iteration, d_{mxm} value of the sample set consisting of n_{ex} existing samples and n_{new} randomly selected candidate samples is computed and recorded. When MAX_ITER iteration is reached, the candidate sample set with the largest d_{mxm} is elected. In this work, MAX_ITER is set to 100.

$$d_{mxm} = \max_{1 \leq i < j \leq n_s} d_{\min}(x_1, x_2, \dots, x_{n_s}) \quad (4)$$

$$d_{\min}(x_1, x_2, \dots, x_{n_s}) = \min_{0 < i < j \leq n_s} d_{ij} = \min_{0 < i < j \leq n_s} \|x_i - x_j\|$$

- Step 3. (Distance inspection and design space expansion, lines 5–9): If the best samples pass the distance inspection, namely $d_{mxm} > \varepsilon$, no operation on design space expansion is required, and the current samples are regarded as the final results of IMS-LHD. Otherwise, if $d_{mxm} \leq \varepsilon$, it is revealed that the generated samples are too crowded, which is likely to cause singular or ill-conditioned matrices in refitting a RSM. In that case, we have to discard the crowded samples, and then double the minimum design space factor ($\sigma=2$) to enlarge the current SDS. After that, turn to line 1 to restart IMS-LHD.
- Step 4. (Pick promising samples, lines 11–13): If more than p sample points exist inside V_{SDS} , only p best promising sample points with lower objective or merit function values are selected from S_{ex} to refit RSM. We discard the other non-promising samples for two reasons. First, RSM constructed based on promising samples rapidly leads to the local optimum in V_{SDS} . On the other hand, CPU time spent on updating RSM can be saved by using limited samples, especially for HD problems.
- Step 5. (Output results, lines 14): The finally results of IMS-LHD are returned.

In order to demonstrate the benefits of IMS-LHD, IMS-LHD and *lhsdesign* function are employed to generate sequential samples in a unit hypercube design space containing some existing samples for problems of 2–20 dimensions. In this test, it is assumed that $0.5p$ initial samples have already existed, and $0.5p$ sequential samples need to be generated. To eliminate random variation, for each problem 50 consecutive runs using two sequential sampling schemes are performed. For fair comparison, in each run $0.5p$ randomly designated initial samples are kept the same for IMS-LHD and *lhsdesign*. The average and minimum values of maximin distance for all the testing problems are illustrated in Fig. 3. As can be seen from Fig. 3, the proposed IMS-LHD apparently improves the space-filling property of samples compared with *lhsdesign*. Additionally, the average value and minimal value from IMS-LHD are almost the same, which proves its good robustness. In contrast, *lhsdesign* produces inferior samples, especially for low dimensional problems.

Apart from aforementioned benefits to generate evenly distributed sequential samples, the sample inheritance mechanism of IMS-LHD can dramatically save computational burden. Besides, compared with the inherited LHD sampling scheme in IARSM (Wang 2003), IMS-LHD mainly focuses on quality improvement of samples and is easier for implementation.

3.3 New significant design space identification algorithm

The new SDS identification algorithm is the most important and creative technique in ISES to improve optimization efficiency and global exploration capability of ARSM-ISES. This new SDS identification algorithm is developed based on the primitive SDS approach first introduced by the authors (Long

et al. 2012a, b). ISES inherits the basic concept of SDS approach to implement space reduction without performing global optimization.

Before presenting the new SDS identification algorithm, we first introduce the definition of significant design space as below.

Definition 1 Significant design space (SDS) is a relatively small hypercube sub-region where local or global optima probably locate. SDS is determined by two components, namely, center (C) and size (L). Mathematical formulation of the region defined by SDS is given in (5). During the optimization process, SDS is automatically moved, contracted, or enlarged according to the known information, such as the size of the current design space, position of the best sample, and fitting quality of the current RSM.

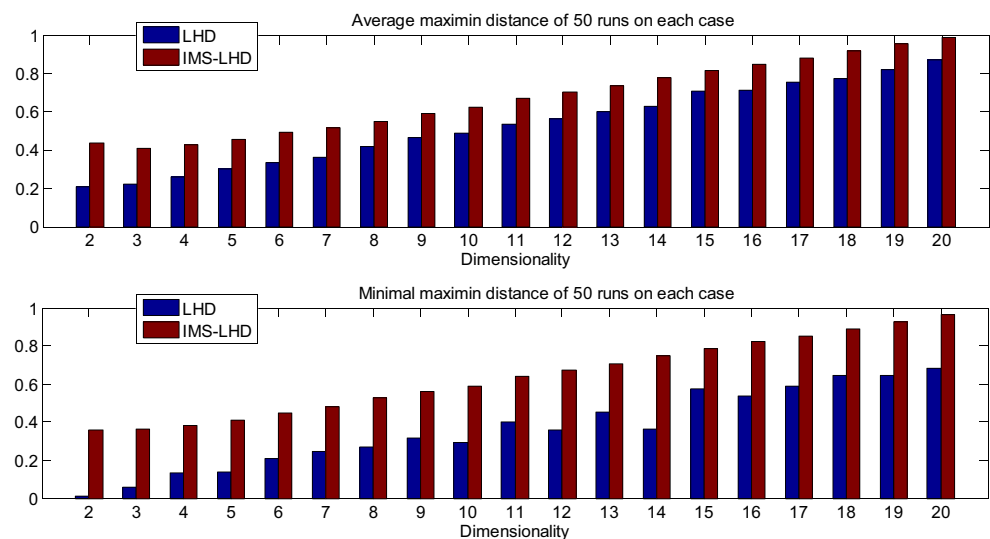
$$\mathbf{V}_{SDS} = \left\{ \mathbf{x} \mid x_i^{lb(S)} \leq x_i \leq x_i^{ub(S)}, 1 \leq i \leq n_v \right\} \quad (5)$$

$$\text{where } x_i^{lb(S)} = \max(x_i^{lb}, C_i - L_i);$$

$$x_i^{ub(S)} = \min(x_i^{ub}, C_i + L_i);$$

There are several basic guidelines to develop the algorithm for identifying SDS at each iteration. First, identified SDS has to be small enough to obtain good fitting quality of RSM and to speed up local exploitation. Second, SDS is expected to thoroughly utilize existing expensive samples to pursue the global optimum. Third, once acceptable fitting quality of RSM is achieved, the size of SDS should be enlarged to avoid premature convergence and being trapped at local optima. According to the above guidelines, an optimality

Fig. 3 Comparisons of *lhsdesign* and IMS-LHD for sequential sampling



and accuracy driven concept is developed to identify SDS, i.e., when poor fitting quality is observed, the size of SDS is reduced for boosting approximation accuracy, and once RSM provides good approximation, the size of SDS is enlarged for better global exploration. The

pseudo codes of the new SDS identification algorithm at the k -th iteration are listed in **Algorithm 2**.

Step 1. (Identify unexplored SDS, lines 1–8). If improvement on the objective has not been achieved ($N_{ni} \geq 1$),

Algorithm 2 SDS Identification Algorithm

Input: bounds of initial design space $[x^{lb(1)}, x^{ub(1)}]$; design library at current iteration $[S, F]$; accuracy of current RSM ξ ; acceptable accuracy of RSM ξ_a ; set of SDS' obtained in previous iterations $\Theta = [V_{SDS}^{(1)}, V_{SDS}^{(2)}, \dots, V_{SDS}^{(k)}]$; current sizing factor $\mathcal{G}^{(k)}$; potential optimal point found at current iteration $x_{po}^{*(k)}$; pseudo optimal point found at current iteration $x_{ps}^{*(k)}$; true objective value at pseudo optimal point $f_{ps}^{*(k)}$; RSM objective value at pseudo optimal point $\hat{f}_{ps}^{*(k)}$; minimum design space factor σ ; objective tolerance ε ; number of non-improvement iterations N_{ni}

Output: updated SDS $V_{SDS}^{(k+1)} = [x^{lb(k+1)}, x^{ub(k+1)}]$; updated sizing factor for the next iteration $\mathcal{G}^{(k+1)}$

Begin

```

1  if  $N_{ni} \geq 1$  then
2     $[V_{UXS}] \leftarrow \text{IdentifyUnexploredSDS}(S, \Theta, x^{lb(1)}, x^{ub(1)}, \sigma)$ ;
3    if  $V_{UXS} \neq \emptyset$ 
4       $V_{SDS}^{(k+1)} = V_{UXS}$ ;
5      go to line26;
6    else
7      go to line10
8    end
9  else
10    $\xi = \text{abs}((f_{ps}^{*(k)} - \hat{f}_{ps}^{*(k)}) / f_{ps}^{*(k)})$ ;
11   if  $\xi \geq 5\xi_a$  then
12      $\mathcal{G}^{(k+1)} = 1 / \log(\xi / \xi_a)$ ;
13   else if  $\xi \leq \xi_a / 5$ 
14      $\mathcal{G}^{(k+1)} = \log(\xi_a / \xi)$ ;
15   else
16      $\mathcal{G}^{(k+1)} = 1$ ;
17   end
18    $C^{(k+1)} = x_{po}^{*(k)}$ ;
19    $L^{(1)} \leftarrow \text{GetSpaceLength}(V_{SDS}^{(1)})$ ;
20    $L^{(k)} \leftarrow \text{GetSpaceLength}(V_{SDS}^{(k)})$ ;
21    $L^{(k+1)} = \max(\mathcal{G}^{(k+1)} L^{(k)}, \sigma L^{(1)})$ ;
22    $x^{lb(k+1)} = \max((C^{(k+1)} - 0.5 L^{(k+1)}), x^{lb(1)})$ ;
23    $x^{ub(k+1)} = \min((C^{(k+1)} + 0.5 L^{(k+1)}), x^{ub(1)})$ ;
24    $V_{SDS}^{(k+1)} = [x^{lb(k+1)}, x^{ub(k+1)}]$ ;
25  end
26 end
27 return  $[V_{SDS}^{(k+1)}, \mathcal{G}^{(k+1)}]$ ;

```

an operation to identify an unexplored SDS (V_{UXS}) will be triggered. Unexplored SDS can help ARSM-ISES to escape from local optima and increase the probability of detecting the true global optimum. More detailed description for

unexplored SDS will be presented later. If V_{UXS} is not empty, **Algorithm 2** terminates and returns V_{UXS} as the identified SDS for the next iteration, otherwise, turn to line 10 to build a regular SDS.

- Step 2. (Assess fitting quality of the current RSM, line 10) If the current iteration successfully decreases the objective or an empty V_{UXS} is found, a regular SDS need to be constructed. The fitting quality of the current RSM is used for space operations. R-square has been widely used for evaluating accuracy of RSM. However, since only p samples are used to fit RSM, RSM actually interpolates all sample points and the R-square value always reaches its maximum value of 1. Thus, R-square is not applicable to assessing accuracy of an interpolating RSM here. The relative error at the pseudo-optimum $\xi = \text{abs}\left(\left(f_{ps}^{*(k)} - \hat{f}_{ps}^{*(k)}\right) / f_{ps}^{*(k)}\right)$ is used instead to assess the quality of RSM. Because ξ directly reflects the accuracy of RSM at the pseudo-optimum that will gradually become identical to the local or even global optimum as optimization proceeds, space operations based on ξ can effectively ensure RSM's fitting quality in the neighborhood around the optima.
- Step 3. (Space contraction operation, lines 11–12) A sizing factor $\vartheta^{(k)} > 0$ is introduced to control the size of SDS according to the accuracy of RSM. When a large ξ is observed, the space contraction operation is required to shrink SDS for improving accuracy of RSM at the next iteration. For this purpose, we need to design a proper contraction operator to restrict the size factor as $0 < \vartheta^{(k)} \leq 1$, which should be a simple monotonically decreasing function with respect to ξ at the pseudo-optimum. In the primitive SDS approach (Long et al. 2012a, b), a ratio function is employed to compute the sizing factor, i.e., $\vartheta^{(k)} = \xi_a / \xi$. Although the ratio function meets the basic requirement for a contraction operator, it may become so aggressive that the design space is over shrunk when very poor RSMs are built at initial iterations. The dotted curve in Fig. 4 illustrates that the sizing factor is rapidly decreased to trivial values as ξ / ξ_a increases. Since RSM usually produces a large ξ / ξ_a at the beginning

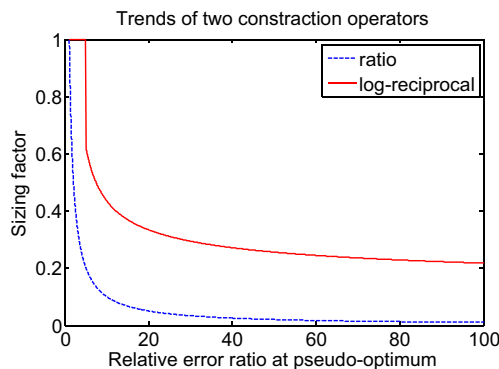


Fig. 4 Plot of two contraction operators

of ARSM-ISES optimization, the ratio function is very likely to eliminate the global optimum due to the overly greedy space contraction operation. A more effective log-reciprocal contraction operator is thus developed to determine the sizing factor as $\vartheta^{(k+1)} = 1 / \log(\xi / \xi_a)$. From observation of Fig. 4, the decreasing trend of the sizing factor obtained by the new contraction operator is much milder, which can improve the RSM accuracy, meanwhile, greatly reduce the risk of losing the global optimum. To avoid negative sizing factors, only when $\xi \geq 5\xi_a$, a log-reciprocal contraction operator is called to reduce the sizing factor.

- Step 3. (Space expansion operation, lines 13–14) For better global exploration, the space expansion operation is essential to enlarge SDS when the current RSM provides good approximation. In case of $\xi \leq \xi_a / 5$, the log expansion operator $\vartheta^{(k+1)} = \log(\xi_a / \xi) \geq 1$ is adopted to update the sizing factor, which increases the size of SDS at the next iteration for wider global exploration. The space expansion operation is beneficial to reducing the probability of missing the true global optimum.
- Step 4. (Space conservation operation, lines 15–17) If the current RSM provides a moderate but acceptable fitting quality, i.e., $\xi_a / 5 < \xi < 5\xi_a$, the space conservation operation is carried out to maintain the current size for constructing the next SDS. Although the size of next SDS will not change, the renewed center forces the SDS to move towards promising regions according to Step 5.
- Step 5. (Identification of the new SDS center, line 18) In metamodel-based optimizations without any prior knowledge, acquired expensive samples are the most credible and precious information for space exploration. Even though the potential optimal point may differ from the true global optimum, particularly at initial iterations, it is the most promising point under the current known information. Hence, the potential optimal point is used as the center of the next SDS. During ARSM-ISES optimization, the updated center drives SDS to continuously move towards the true global or local optimum.
- Step 6. (Construction of a regular SDS, lines 19–26) After the size and center have been determined in previous steps, a regular SDS can be constructed for the next iteration. The actual size of SDS in each variable component is computed by the size of the current SDS multiplied by the sizing factor. Because the center of SDS may locate near the bounds of the initial design space, SDS must be limited within the lower and upper bounds of each variable component

to prevent SDS from exceeding the initial design space. The minimum design space factor σ is used to prevent SDS from being over small.

Step 7. (Final results, line 27) Return the identified SDS as the design space for the next iteration, as well as the updated sizing factor.

Referring to Step 1, a particular SDS called unexplored SDS ($\mathbf{V}_{UXS}$) and its identification algorithm are proposed to explore scarcely tapped regions. Definitions of unexplored samples and unexplored SDS are presented in **Definition 2** and **Definition 3**, respectively.

Definition 2 The expensive samples in a design library that locate outside all the exploited SDS' excluding the first SDS (i.e., the initial design space) are regarded as unexplored samples (notated as $\mathbf{S}_{UXP}$), which can be expressed as (6). n_s is the total number of expensive samples, and k is the iteration counter.

$$\mathbf{S}_{UXP} = \left\{ \mathbf{x}^j \mid \forall j, \forall i, \mathbf{x}_i^j > \mathbf{x}_i^{ub(q)} \vee \mathbf{x}_i^j < \mathbf{x}_i^{lb(q)}; j = 1, \dots, n_s; i = 1, \dots, n_v; q = 2, \dots, k \right\} \quad (6)$$

Definition 3 As a special SDS, an unexplored significant design space $\mathbf{V}_{UXS}$ is determined by unexplored samples defines a hypercube sub-region. Unexplored SDS holds the same mathematical formulation as a regular SDS in (5).

Because only a small number of samples are generated at initial iterations, the limited information and poor approximation capability of RSM for multimodal functions probably make certain promising samples become unexplored. Hence, when non-improvement is found at certain iteration, it is rational to give unexplored samples opportunities to guide RSM-based search in rarely tapped regions. The methodology to identify an unexplored SDS is summarized in **Algorithm 3**.

Step 1. (Collect all the unexplored samples, lines 1–5): Unexplored samples are selected from the current design library through comparing coordinates of each sample with lower and upper bounds of existing SDS'. If less than two unexplored samples are collected (i.e., $n_{ux} < 2$), no non-null unexplored SDS can be achieved.

Step 2. (Determine the center of unexplored SDS, lines 6–7): If $n_{ux} \geq 2$, a non-null unexplored SDS can be constructed, and meanwhile intersections with previous regular SDS' should be minimized. All the unexplored samples are used as vertices to form an irregular homogeneous polyhedron. Then, the centroid of that polyhedron is computed as the center of the unexplored SDS. Note that the polyhedron is only employed to find the center of the unexplored SDS that is still a hypercube with a size determined in the next step.

Algorithm 3 Unexplored SDS identification algorithm

Input: lower and upper bounds of initial design space $[\mathbf{x}^{lb}, \mathbf{x}^{ub}]$; design library at current iteration $[\mathbf{S}, \mathbf{F}]$; a set of SDS' obtained in previous iterations $\Theta = [\mathbf{V}_{SDS}^{(1)}, \mathbf{V}_{SDS}^{(2)}, \dots, \mathbf{V}_{SDS}^{(k)}]$; minimum design space factor σ ; sizing factor $\mathcal{G}^{(k)}$;

Output: unexplored SDS $\mathbf{V}_{UXS} = [\mathbf{C}_{UXS}, \mathbf{L}_{UXS}]$; sizing factor $\mathcal{G}^{(k+1)}$

Begin

1 $[\mathbf{S}_{UXP}, n_{ux}] \leftarrow \text{CollectUnexploredSamples}(\mathbf{S}, \Theta)$;

2 **if** $n_{ux} < 2$

3 $\mathbf{V}_{UXS} = \emptyset$;

4 $\mathcal{G}^{(k+1)} = \mathcal{G}^{(k)}$;

5 **go to** line 16

6 **else**

7 $\mathbf{C}_{UXS} \leftarrow \text{ComputeCentroid}(\mathbf{S}_{UXP})$;

8 $\mathbf{L}_{UXS} \leftarrow \text{ComputerMeanDist}(\mathbf{S}_{UXP})$;

9 $\mathbf{L}^{(1)} \leftarrow \text{GetSpaceLength}(\mathbf{V}_{SDS}^{(1)})$;

10 $\mathbf{L}_{UXS} = \max(\mathbf{L}_{UXS}, \sigma \mathbf{L}^{(1)})$;

11 $\mathbf{x}_{uxs}^{lb} = \max((\mathbf{C}_{UXS} - 0.5 \mathbf{L}_{UXS}), \mathbf{x}^{lb(1)})$;

12 $\mathbf{x}_{uxs}^{ub} = \min((\mathbf{C}_{UXS} + 0.5 \mathbf{L}_{UXS}), \mathbf{x}^{ub(1)})$;

13 $\mathbf{V}_{UXS} = [\mathbf{x}_{uxs}^{lb}, \mathbf{x}_{uxs}^{ub}]$;

14 $\mathcal{G}^{(k+1)} = 1.0$;

15 **end**

16 **return** $[\mathbf{V}_{UXS}, \mathcal{G}^{(k+1)}]$;

- Step 3. (Determine the size of the unexplored SDS, lines 8–10): Unlike determining a regular SDS, the size of unexplored SDS is independent of the size of the current SDS, but is decided in terms of the distances among unexplored samples. The average value of distances between unexplored samples is used as the size of the unexplored SDS. In addition, the size of an unexplored SDS should be larger than the minimal size determined by the minimum design space factor σ .
- Step 4. (Construct the unexplored SDS, lines 11–15): By using the size and center, the unexplored SDS can be constructed in a same manner as a regular SDS. It can be found that no sizing factor is required to identify an unexplored SDS. Generally speaking, an unexplored SDS is larger than the current SDS. Once the unexplored SDS is successfully constructed, the sizing factor is reset to 1.0.
- Step 5. (Output final results, line 16): Return the unexplored SDS as the design space for the next iteration, as well as the renewed sizing factor.

Figure 5 demonstrates the concept of constructing an unexplored SDS for a 2-D problem. In Fig. 5, the hollow circles indicate the explored samples gathering around the local optima (as marked by the triangles) at

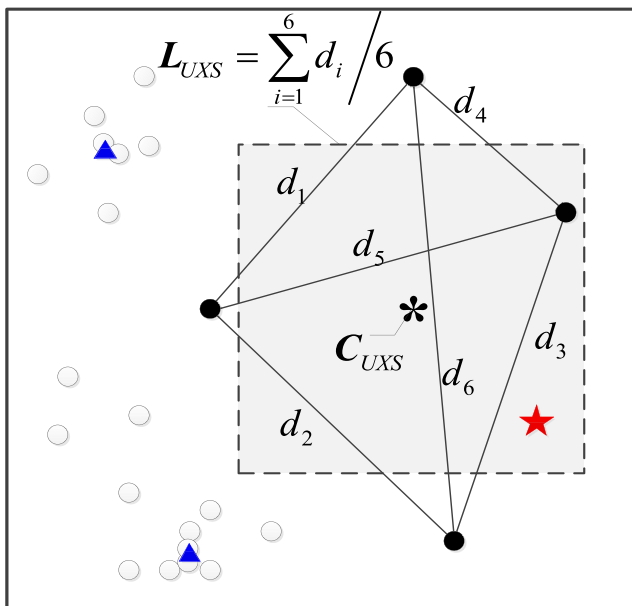


Fig. 5 Illustration of unexplored SDS for a 2-D problem

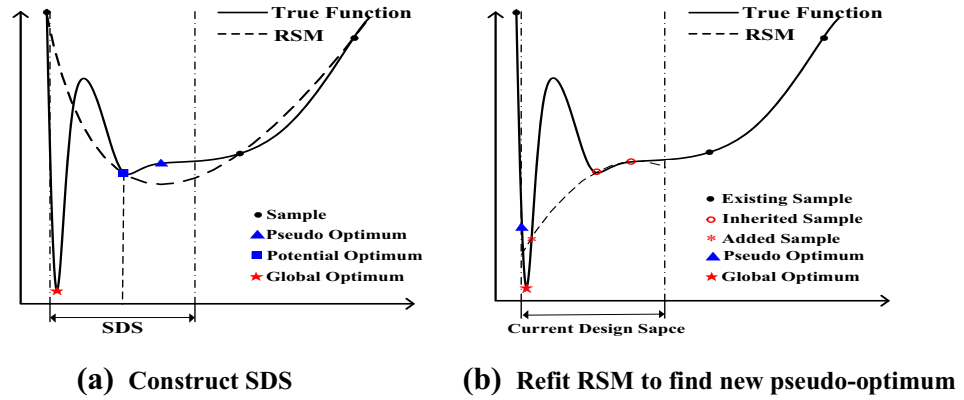
previous iterations, and the filled circles depict four unexplored samples that determine a special homogeneous polyhedron (i.e., quadrilateral for this problem). The centroid of this quadrilateral is computed as the center of the unexplored SDS, as indicated by the asterisk. The average value of the six distances between every two unexplored samples is calculated to determine the size of this unexplored SDS. The final unexplored SDS is the shadowed region inside the dotted square, which covers most area of the quadrilateral governed by the unexplored samples and contains the global optimum (as indicated by the pentagram).

From aforementioned description of ARSM-ISES, it is clear that no auxiliary global optimization is needed to identify regular and unexplored SDS, which results in higher algorithm execution performance. More importantly, with the intelligent SDS management scheme, the probability of missing the actual global optimum can be reduced. Take the same 1-D example for instance to illustrate the benefits of ISES, as depicted in Fig. 6. Unlike the cutting plane discarding the region containing the true global optimum as depicted in Fig. 1, SDS determined by ISES captures the actual global optimum shown in Fig. 6a. Figure 6b shows that two existing samples (expressed as hollow circles) are inherited to recalibrate RSM after the first fitting. Due to the behavior of IMS-LHD, one sequential sample far from inherited samples is added, depicted by the asterisk. Based on the refitted RSM, the new pseudo-optimum is found on the lower bound of SDS. Note that if the Hessian matrix of RSM is positive definite, the pseudo-optimum locates inside the SDS; otherwise, it locates on the bound of SDS. The next SDS using the newly-added sample as center gets much closer to true global optimum. Moreover, Fig. 6 indicates another important trend that as optimization continues, the pseudo-optimum successively approaches to the potential optimum until the two coincide, which also conforms to our expectation.

3.4 Novel termination criteria of ARSM-ISES

In algorithm development, it is preferable to devise criteria to terminate the optimization process automatically in an effective manner. In this paper, a specific assembly of criteria to exam whether ARSM-ISES should terminate is proposed according to unique features of ISES, whose pseudo codes are summarized in **Algorithm 4**.

Fig. 6 Illustration of ISES without missing the true global optimum. **a** Construct SDS; **b** Refit RSM to find new pseudo-optimum



As can be seen from **Algorithm 4**, the developed termination criteria are much complicated than popular rules of examining the objective tolerance and maximum number of

function evaluations used in MBDO algorithms. More detailed description of the proposed termination criteria are presented as below.

Algorithm 4 Termination Criteria

Input: objective value at current potential optimum $f_{po}^{*(k)}$; objective value at last potential optimum $f_{po}^{*(k-1)}$; relative error at current pseudo-optimum ξ ; acceptable relative accuracy of RSM ξ_a ; objective tolerance ε ; number of function evaluation N_{fe} ; allowed maximum number of function evaluations MAX_NFE; number of non-improvement iterations N_{ni} ; allowed maximum number of non-improvement iterations MAX_NNI.

Output: termination flag $bStop$; renewed number of non-improvement iterations N_{ni} .

Begin

```

1   $bStop = false;$ 
2  if  $N_{fe} \geq MAX\_NFE$  then
3     $bStop = true;$ 
4    go to line 20;
5  end
6   $\Delta = abs\left(\frac{f_{po}^{*(k)} - f_{po}^{*(k-1)}}{f_{po}^{*(k)}}\right);$ 
7  if  $\Delta \leq \varepsilon$  or  $abs(\Delta f_{po}^{*(k)}) \leq 0.1\varepsilon$  then
8    if  $\xi \leq \xi_a$  and  $N_{ni} \geq 2$  then
9       $bStop = true;$ 
10     go to line 20;
11   elseif  $N_{ni} \geq MAX\_NNI$ 
12      $bStop = true;$ 
13     go to line 20;
14   else
15      $N_{ni} = N_{ni} + 1;$ 
16   end
17 else
18    $N_{ni} = 0;$ 
19 end
20 return  $[bStop, N_{ni}]$ ;

```

- Step 1. (Set default termination flag, line 1): The default value of termination flag *bStop* is set to *false*, which indicates that optimization process has to continue until the *true* value is observed.
- Step 2. (Maximum number of function evaluations criterion, lines 2–5): If the number of function evaluation reaches or exceeds the allowed maximum number of function evaluations MAX_NFE, the *bStop* is change to *true* and returned.
- Step 3. (Tolerance, accuracy, and stall criteria, lines 6–10): Relative difference of objective values at two consecutive iterations is computed, i.e. $\Delta = \text{abs}((f_{po}^{*(k)} - f_{po}^{*(k-1)})/f_{po}^{*(k)})$. If Δ is less than user-defined tolerance ε or the absolute difference is less than 0.1ε , the tolerance criterion is met. The use of absolute difference can stop ARSM-ISES once the search process approaches to zero value for some problems whose global optimal objective value is zero. For most optimization methods, once the tolerance criterion is satisfied, optimization process should be stopped. However, the potential optima found at two consecutive iterations very likely remain the same due to the poor fitting quality of RSM. Thus, the tolerance criterion may cause premature convergence. To correctly assess convergence, an additional accuracy criterion based on RSM accuracy is introduced. When both accuracy criterion and tolerance criterion are met, it indicates that ARSM-ISES has converged to a local or global optimum. In order to utilize the merit of unexplored SDS for improving the probability of finding global optimum, a stall criterion (i.e., $N_{ni} \geq 2$) is added to force ARSM-ISES to conduct at least one unexplored SDS operation. Therefore, in this step, *bStop* becomes true only if tolerance, accuracy, and stall criteria are all satisfied.
- Step 4. (Maximum stall criterion, lines 11–13): Maximum stall criterion is introduced to prevent wasting expensive samples in a trivial neighborhood of an optimum especially for some problems with zero-value extrema. On the premise that the tolerance criterion is satisfied, if N_{ni} exceeds the maximum stall limitation, *bStop* is set to be true.
- Step 5. (Update the number of non-improvement iterations, lines 14–19): If the tolerance criterion is satisfied, the number of non-improvement iterations is increased by one. Otherwise, if any improvement is obtained, it is reset to be zero.
- Step 6. (Output results, line 20): Return the renewed termination flag and number of non-improvement iteration. If *bStop* is true, ARSM-ISES optimization terminates and then output the current potential optimum.

3.5 Adaptive penalty function method

The original ARSM and some variants have not provided effective expensive constraint handling mechanisms; they are only applicable to optimizations with boundary and cheap constraints. To extend the application scope of ARSM-ISES, an adaptive penalty function method is proposed to handle expensive constraints that frequently appear in real-world engineering applications.

Wang and Simpson (2004) employed a predefined static penalty factor to handle expensive constraints. However, the appropriate value of a static penalty factor is problem-dependent, and hard to set. The adaptive method for adjusting the penalty factor is more appealing. In this work, the initial penalty factor is set to 1. At each iteration, the maximum violation of expensive constraints $s_{\max}$ is used to determine

Table 1 List of tuning parameters required to run ARSM-ISES

Tuning parameters	Requirement	Recommended values	Values used in this paper
Number of initial samples	$n_{is} \geq p$	$[p, 3p]$	$2p$
Number of refitting samples	$n_{rf} \geq p$	$[p, 3p]$	p
Acceptable error of RSM	$\xi_a > 0$	$[0.005, 0.2]$	0.01
Initial minimum space factor	$\sigma > 0$	$[0.001, 0.1]$	0.05
Objective tolerance	$\varepsilon > 0$	$[0.001, 0.01]$	0.005
Constraint violation tolerance	$s_{tol} \geq 0$	$[0.0, 0.01]$	0.001
Maximum number of function evaluations	MAX_NFE > 1	$\geq 100n_{is}$	$500(n_v \leq 6)$ $10000(n_v > 6)$
Maximum stall number	MAX_NNI ≥ 1	$[2, 10]$	3
Initial penalty factor	$\lambda^{(1)} > 0$	$[1, 10]$	1
Increment factor for penalty	$\mu > 1$	$[1, 10]$	2
Maximum penalty factor	$\lambda_{\max} > 0$	$[100, 1000]$	500

Table 2 Numerical benchmark problems

Category	Function	# of design variables	Initial design space	Analytic global optimum solution
Low dimensional problems	SE	2	$x_{1,2} \in [0, 5]$	-1.457
	PK	2	$x_{1,2} \in [-3, 3]$	-6.551
	SC	2	$x_{1,2} \in [-2, 2]$	-1.032
	BR	2	$x_1 \in [-5, 10]; x_2 \in [0, 15]$	0.397
	RS	2	$x_{1,2} \in [-1, 1]$	-2.000
	GF	2	$x_{1,2} \in [-5, 5]$	0.000
	GP	2	$x_{1,2} \in [-2, 2]$	3.000
	GN	2	$x_{1,2} \in [-100, 100]$	0.000
	HN	6	$x_{1,2}, \dots, x_6 \in [0, 1]$	-3.322
High dimensional problems	HD1	10	$x_{1,2}, \dots, x_{10} \in [-3, 2]$	0.000
	R10	10	$x_{1,2}, \dots, x_{10} \in [-5, 5]$	0.000
	F16	16	$x_{1,2}, \dots, x_{16} \in [-1, 1]$	25.875

whether the penalty factor should be increased or not. If the maximum constraint violation is larger than a user-defined tolerance s_{tol} , the penalty factor is increased according to a predefined increment factor $\mu > 1$, so that the next iteration will focus on enhancing feasibility. In addition, the upper limitation of the penalty factor is imposed to avoid numerical difficulties. If $s_{max} < s_{tol}$, we suppose that the current penalty factor is competent to limit the search inside or at least near the feasible region. In that case, no increment of penalty factor is needed and further improvement on objective is emphasized at the next iteration. The formula to adjust the penalty factor can be expressed as

$$\lambda^{(k+1)} = \begin{cases} \min(\mu\lambda^{(k)}, \lambda_{max}) & s_{max} \geq s_{tol} \\ \lambda^{(k)} & s_{max} < s_{tol} \end{cases} \quad (7)$$

Since the objective function and constraints probably involve different orders of magnitude, normalization is very important to make the adaptive penalty function method and ARSM-ISES more efficient and robust. In current work, the mean values of absolute values of the objective and constraints from initial samples are used for normalization.

It is worth noting that for optimization problems using expensive constraints, objective function values at potential and pseudo optimal points in all the aforementioned algorithms should be replaced by corresponding merit function values. Besides, an additional constraint violation criterion (i.e., $s_{max} \leq s_{tol}$) has to be imposed in Step 3 of **Algorithm 4** to ensure a feasible solution.

3.6 Tuning parameters of ARSM-ISES

It is commonly recognized that a good and practical MBDO algorithm should have fewer and easily configured tuning parameters. Here, we will summarize tuning parameters of ARSM-ISES and corresponding recommended values.

Although quite a few input parameters are required in aforementioned algorithms, most of them are intermediate results produced during ARSM-ISES optimization process. Most of 11 tuning parameters listed in Table 1, such as the number of samples to build and refit metamodels, tolerance, maximum number of function evaluations, etc., are also commonly used in other MBDO algorithms. In fact, only the acceptable error of RSM, initial minimum space factor, maximum stall number, and the parameters for adjusting the penalty factor are unique inputs for ARSM-ISES. According to our

Table 3 Parameter configurations for EGO, MSEGO, and MPS

Parameter	Value	
	EGO/MSEGO	MPS
# of initial samples	12 ($n_v=2$); 56 ($n_v=6$); $10n_v$ ($n_v>6$)	$2p+1-n_v$
Maximum # of iterations	40	N/A
# of increasing samples in each iteration	EGO: 1 MSEGO: 3	n_v or $1.5n_v$
# of cheap points in each iteration	N/A	10000
Difference coefficient	N/A	0.01
Acceptable difference	0.005	N/A

As a default setting, Kriging, RBF, and Shepard metamodels are used in MSEGO

Table 4 Comparison results with ARSM and IARSM

Func.	Global optimum obtained					Number of function evaluation (N_f)				
	ARSM	IARSM	IARSM	ARSM-ISES		ARSM	IARSM	IARSM	ARSM-ISES	
		I	II	Best	Median		I	II	Best	Mean
SC	-0.866	-1.026	-1.029	-1.032	-1.032	100	39	44	25	32
BR	2.099	0.417	0.398	0.398	0.398	50	15	36	29	37.5
RS	-2.000	-1.417	-1.854	-2.000	-2.000	9	17	60	27	31
GF	0.609	0.444	0.082	0.000	0.000	144	29	46	35	57
GP	3.210	3.250	3.000	3.000	3.004	70	30	77	28	37.1
HN	-3.320	-2.652	-2.456	-3.322	-3.322	1248	158	105	142	188.6

experience, recommended values and used values to test ARSM-ISES in Section 4 are presented in this table as well.

To collect more information, $2p$ initial samples are used for reducing the probability of missing global optimum. Since the number of refitting samples mainly influences optimization efficiency, to save expensive function calls, only p samples are used to refit RSM at each iteration. Any acceptable error meeting the requirements of engineering application can be used for configuring ARSM-ISES. In this work, we set acceptable error to be 5 %. The initial minimum space factor is set to be 0.05, which can also be automatically adjusted in IMS-LHD. For a good tradeoff between efficiency and global exploration capability, the maximum stall number is set to be three. The maximum number of function evaluations is set to be large enough to prevent ARSM-ISES from premature termination.

4 Benchmark problems and engineering application

A number of benchmark problems and a real-world engineering application are used to test the performance of ARSM-ISES in terms of global convergence, number of function

evaluations, robustness, algorithm execution overhead, and capability of handling expensive constraints.

4.1 Description of numerical benchmarks and algorithm setup

Here a number of multimodal numerical benchmarks are used to test ARSM-ISES. Basic information of those numerical benchmark problems are listed in Table 2. SE and PK indicate the Sasena function (Viana et al. 2013) and Peaks function (Zhu et al. 2012a; Li et al. 2013) respectively. Formulae of all the numerical testing functions are presented in Appendix.

If the number of design variables is more than 10, the problem is considered to be high dimensional (HD), otherwise low dimensional (LD). To demonstrate the performance of the proposed algorithm, several existing MBDO algorithms including ARSM, IARSM, MPS, EGO, and MSEG0 are used for comparison. Tuning parameters of ARSM-ISES are configured in light of the last column in Table 1. Table 3 details the parameter configurations for EGO, MSEG0, and MPS. The number of initial samples of EGO and MSEG0 for 2-D and 6-D problems are the same as those in the reference (Viana et al. 2013). Because the maximum numbers of

Table 5 Global exploration capability comparison on LD benchmarks

Func.	ARSM-ISES		MPS		EGO		MSEG0	
	Var. range	Median	Var. range	Median	Var. range	Median	Var. range	Median
SE	[-1.457, 2.866]	-1.457	[-1.457, 6.538]	-1.457	[-1.456, -1.436]	-1.453	[-1.456, -1.454]	-1.456
PK	[-6.551, -6.550]	-6.551	[-6.551, -3.040]	-6.551	[-6.550, -6.383]	-6.550	[-6.498, -5.979]	-6.498
SC	[-1.032, -1.030]	-1.032	[-1.032, -1.029]	-1.032	[-1.032, -1.031]	-1.031	[-1.024, -0.987]	-1.024
BR	[0.398, 0.399]	0.398	[0.398, 1.393]	0.398	[0.398, 0.400]	0.398	[0.398, 0.431]	0.398
RS	[-2.000, -1.395]	-2.000	[-2.000, -1.516]	-2.000	[-1.375, -1.283]	-1.375	[-1.874, -1.636]	-1.874
GF	[0.000, 0.003]	0.000	[0.000, 1.214]	0.000	[0.966, 3.480]	0.966	[0.001, 0.035]	0.001
GP	[3.000, 30.032]	3.004	[3.005, 1.089E3]	3.105	[7.581, 43.353]	7.581	[3.002, 3.014]	3.002
GN	[0.000, 0.000]	0.000	[0.000, 6.223]	0.000	[0.459, 0.459]	0.459	[0.176, 0.627]	0.177
HN	[-3.322, -3.193]	-3.322	[-3.322, -3.194]	-3.322	[-3.316, -3.308]	-3.313	[-3.208, -3.052]	-3.145

Table 6 Number of function evaluations comparison on LD benchmarks

Func.	ARSM-ISES		MPS		EGO		MSEGO	
	Var. range	Mean	Var. range	Mean	Var. range	Mean	Var. range	Mean
SE	[22,35]	29.4	[12,71]	39.1	[52,52]	52(26.0 ^b)	[70,123]	109.6
PK	[22,55]	35.4	[20,57]	39.8	[26,52]	42.6	[129,132]	130.4
SC	[25,38]	31.7	[24,47]	32.9	[27,37]	32.6(16.5 ^b)	[130,132]	131.2
BR	[29,58]	39.8	[14,174]	69.2	[32,41]	36.1(28.0 ^a ,77.5 ^b)	[36,132]	112.6
RS	[27,39]	31.2	[25,122]	52.1	[52,52]	52.0	[131,132]	131.4
GF	[35,74]	57.0	[56,145]	100.4	[52,52]	52.0	[132,132]	132.0
GP	[28,64]	37.1	[9,590]	137.2	[52,52]	52(32.0 ^a)	[101,132]	120.4
GN	[32,45]	38.8	[9,330]	110.2	[52,52]	52.0	[132,132]	132.0
HN	[142,288]	188.6	[365,1091]	613.4	[66,74]	68.8(121.0 ^a)	[176,176]	176.0

EGO results marked by ^a are cited from Jones et al. (1998), each of which was reported from a particular EGO run; results marked by ^b are cited from Sasena (2002), which were median values from 10 EGO runs. All the results marked by ^a and ^b are the numbers of function evaluations when EGO found a solution whose objective has less than 1 % deviation from the theoretical global optimum

iterations for EGO and MSEGO recommended by Viana et al. (2013) are too small to find the global optima for most benchmarks, we increased the maximum numbers of iterations. Moreover, we modified the EGO and MSEGO codes (Viana et al. 2013) to stop when the maximum number of iterations is reached or the result is very close to the analytic global optimum (difference is less than 0.005).

Parameters of MPS in Table 3 are configured by using the default setup suggested in MPS code (Wang et al. 2004). Note that when R-square of RSM is less than 0.9, MPS only increases n_v new samples, otherwise, extra $0.5n_v$ new samples are produced to fit RSM. Some particular termination limitations for HD problems are explained later. Since no codes of ARSM and IARSM are available, we directly cite the results from existing publications for comparison. For the purpose of eliminating randomness, we performed each algorithm for consecutive and independent 10 runs on all the numerical benchmarks. All the experiments for numerical benchmarks are executed on the computer equipped with Core i5-3230M CPU (2.60 GHz) and 12 GB memory.

4.2 Testing ARSM-ISES on LD problems

At first we carried out a comparative study with existing ARSM variants. Although results of ARSM, IARSM I, and IARSM II from the paper (Wang 2003) may be the best values from several trials, the best, median, and mean results of obtained by using ARSM-ISES are summarized in Table 4 for comparison.

The best and median values of obtained global optima in Table 4 show that ARSM-ISES can successfully capture the analytic global optima for all the testing problems. While, the results from ARSM and IARSMs are different from the true optimal for some benchmark problems. N_{fe} is an important metric to indicate the efficiency of MBDO algorithms. ARSM-ISES employs fewer N_{fe} to find the true global optimum for SC and GP than ARSM and IARSM II. For RS and GF, N_{fe} values of IARSM I are less than those of ARSM-ISES. However, IARSM I fails to find the true global optima for those functions. Only for BR, IARSM I succeeds in finding the true global optimum with a lower N_{fe} . It is worth noting that for RS, ARSM converges to the true global optimal solution with only nine function evaluations because the

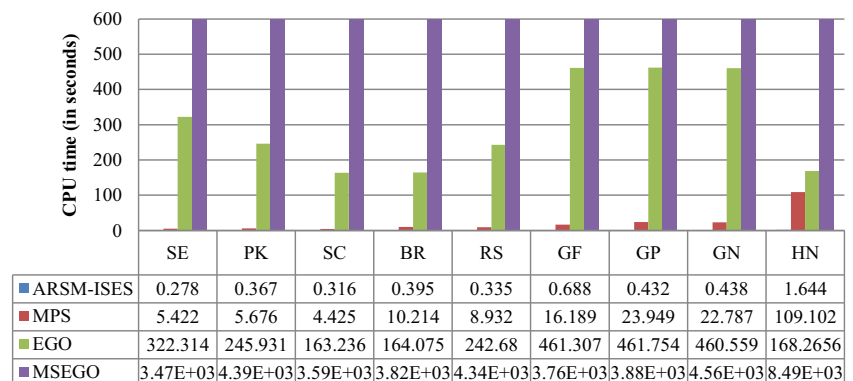
Fig. 7 Plot of algorithm execution overhead (in seconds) on LD benchmark problems

Table 7 Global exploration capability comparison on HD problems

Func.	ARSM- ISES		MPS		TR-MPS	
	Var. range	Median	Var. range	Median	Var. range	Median
F16	[25.875,25.887]	25.875	[29.387,30.615]	29.387	N/A	25.912
R10	[2.714,66.174]	3.147	[70.057,272.384]	70.057	N/A	9.570
HD1	[0.505,0.557]	0.5193	[3.326,5.854]	3.326	N/A	2.019

center sample from the CCD sampling scheme used in ARSM happens to be coincident with the analytic global optimum (0.0, 0.0) of RS. Although no algorithm execution time of ARSM and IARSM was reported, it is believed that they are much more time-consuming than ARSM-ISES, due to the fact that the cutting plane approach needs to call lots of global optimization subroutines to determine the boundaries of the reduced spaces. Therefore, in general ARSM-ISES outperforms ARSM and IARSMs in terms of global exploration capability and efficiency.

To sufficiently test the capability of our proposed algorithm, additional comparative studies are conducted with other well-known MBDO algorithms including MPS, EGO, and MSEGO on LD testing problems. Table 5 details the median values and variation ranges of global optimal solutions obtained by using different MBDO algorithms.

Table 6 shows the mean values and variation ranges of N_{fe} from various competitive algorithms to compare optimization efficiency. As can be observed, ARSM-ISES uses the least mean N_{fe} for most numerical benchmarks. EGO and MSEGO usually cannot stop until the maximum number of iterations is reached, even if we have added an effective (but advantageous over the competitors) termination criterion to tell EGO and MSEGO the analytic global optimum *a priori*. For SC, BR, and HN, EGO shows competitive or even higher efficiency than ARSM-ISES because of the prior knowledge of the analytical optimum. In addition, several reported EGO results from literatures (Jones et al. 1998, Sasena et al. 2002) are also used for comparison, as shown in parentheses. The differences of EGO results are probably caused by the different methods used for maximizing the EI criterion. For example, Jones et al. (1998) used the branch-and-bounds method, and DIRECT was employed by Sasena (2002), while Viana's code used differential evolution algorithm to maximize EI. Compared

with the published results of EGO, ARSM-ISES shows comparable efficiency in finding global optima. Note that fewer N_{fe} of ARSM-ISES may be attained if it terminates once a solution with an objective value within 1 % difference from the theoretical global optimum is found.

In regard to variation ranges of N_{fe} , it can be seen that the minimum values of N_{fe} from MPS are much smaller than ARSM-ISES when optimizing SE, BR, GP, and GN. Nevertheless, those smaller N_{fe} is caused by premature convergence of MPS as shown in the Var. Range column in Table 5. For instance, MPS employs 9 function evaluations to obtain 1.089E3 for GP, which is prohibitively far away from the true global optimum. Although EGO can find near-global optimum with the least N_{fe} for BR, GF and HN, it is not so efficient for solving the rest of benchmark problems, especially when considering the algorithm execution time as discussed later. In addition, the results for GF and GN obtained by using ARSM-ISES prove that the tailored termination criteria are effective and efficient to solve such problems with zero-value global optimal solution.

Thus far, most publications in MBDO field only use N_{fe} to assess the efficiency, while the CPU time consumed to run MBDO algorithm is generally ignored. However, the authors argue that good MBDO algorithms should reduce both N_{fe} and algorithm execution time for shortening the optimization cycle. Actually, it has been found that the computational cost for executing some MBDO algorithms is rather intensive and the prohibitive elapsed time cannot be simply ignored especially for HD problems. Figure 7 shows the bar plot and detailed values of mean algorithm execution time. Note that the figure does not show the high values of MSEGO, since it is cut-off at 600 s; it does not show the time for ARSM-ISES as it is comparatively too small. For 2-D problems, ARSM-ISES only needs less than 1 s for executing algorithm itself. For 6-D

Table 8 Comparison of number of function evaluations on HD problems

Func.	ARSM-ISES		MPS		TR-MPS	
	Var. range	Mean	Var. range	Mean	Var. range	Mean
F16	[462,916]	661.0	[916,931]	921.0	N/A	726.3
R10	[1023,4197]	2638.0	[4198,4204]	4200.6	N/A	6483.3
HD1	[802,2006]	1408.6	[2007,2012]	2008.8	N/A	7137.0

Table 9 Comparison of algorithm execution time (in seconds) on HD problems

Func.	ARSM-ISES		MPS	
	Var. range	Mean	Var. range	Mean
F16	[10.62,26.88]	17.29	[173.71,183.39]	176.860
R10	[15.52,68.02]	37.45	[2.698E4,2.917E4]	2.802E4
HD1	[8.04,20.71]	18.28	[1.615E3,1.784E3]	1.722E3

Table 10 Summary of optimization results obtained by using ASRM-ISES and CiMPS on engineering benchmark problems

Case	Optimal Solution		Nfe		Nce		MCV	
	CiMPS	ARSM-ISES	CiMPS	ARSM-ISES	CiMPS	ARSM-ISES	CiMPS	ARSM-ISES
SD	0.0127 (N/A)	0.0127 (0.0207)	21 (N/A)	123 (117.033)	1911 (N/A)	123 (117.033)	0.0012 (N/A)	6.838E-4 (-0.003)
PVD	7163.739 (N/A)	7197.652 (7928.012)	37 (N/A)	168 (154.800)	335 (N/A)	168 (154.800)	<1.0E-7 (N/A)	9.978E-4 (-0.005)

Numbers in parentheses are average values

problem, the elapsed time is still less than 2 s. In general, the algorithm overhead of ARSM-ISES is two or three orders of magnitude smaller than that of other algorithms. Although MPS and EGO uses more CPU time, several minutes also can be possibly ignored. However, MSEG0 exhibits the worst performance in algorithm execution time. More than one hour is required for simple 2-D problems, and for the 6-D problem the time increases to more than 2.5 h. Such long elapsed time seriously impacts the total optimization time, which cannot be simply ignored.

The metamodel type, sequential sampling method, and metamodel updating strategy directly influence the algorithm execution time. Because RSM is the simplest metamodel method and no global optimization is required in the novel ISES, and RSM-based SQP search is very fast, ARSM-ISES shows high efficiency in algorithm execution time. On the contrary, construction, and evaluation of metamodels used in MPS, EGO, and MSEG0 depend on computing distances between every two samples. When the scale of sample set grows, the CPU time needed to build, update, and evaluate the metamodels increases accordingly. Besides, the heavy computational burden of global optimization processes used to determine the sequential infilling samples dramatically increases the algorithm overhead of EGO and MSEG0. MSEG0 is much worse than EGO due to the use of multiple metamodels.

The results on LD problems prove that ARSM-ISES outperforms not only the previous ARSM variants but also current well-known MBDO algorithms (i.e., MPS, EGO, and MSEG0) in terms of the overall performance in global exploration capability, Nfe , algorithm execution time, and robustness.

4.3 Testing ARSM-ISES with HD problems

Although this work does not specifically focus on efficiently solving HD engineering problems, ARSM-ISES has also been tested using three HD benchmark problems because of the increasing popularity of HD problems encountered in industry.

In this section, ARSM-ISES uses the same tuning parameters for LD problems. Unfortunately, EGO, MSEG0, and MPS encounter difficulties when solving HD problems. EGO and MSEG0 theoretically support solving HD problems, however, the prohibitive algorithm execution time and slow convergence speed become the primary bottleneck hampering their application to HD problems. For example, for the HD1 function, EGO spent more than 80 h to find a mediocre result of 6.427, which differs from the true global optimal by a large margin. Although MSEG0 has not been tested yet, according to the results for LD problems, it is expected that MSEG0 has to spend much more time to run HD optimizations. Therefore, considering the unacceptable efficiency, EGO and MSEG0 are not used for comparison on HD problems.

In fact, MPS also has difficulties for HD optimizations. It has been reported that MPS took 35 and 53 h for solving R10 and HD1 functions respectively and abnormally terminated due to running out the available memory (Cheng and Wang 2012). During our experiments, we also found that MPS rapidly slowed down as the number of samples increased. To save the execution time, MPS is forced to stop when Nfe reaches or exceeds the maximum Nfe from using ARSM-ISES for R10 and HD1 functions. Because F16 is much easier to be optimized, a regular process of MPS is applied. Moreover, an improved version of MPS based on trust region called TR-MPS (Cheng and Wang 2012), which is particularly designed

Fig. 8 Profile and FEA model of lifting surface

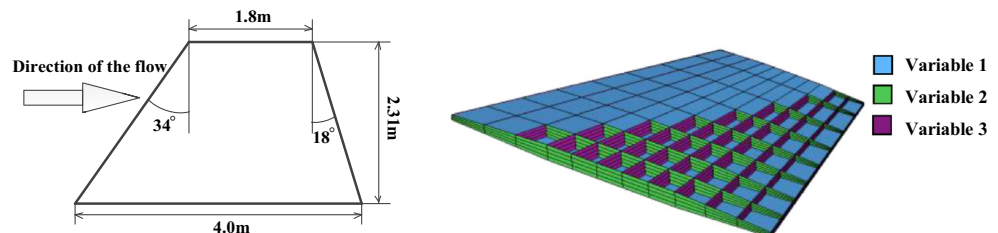


Table 11 Summary of optimization results obtained by using ARSM-ISES

Run #	Optimal design	Optimal weight (kg)	Optimal flutter speed	<i>Nfe</i>	CPU time (min)
1	[1.551,0.877,2.374]	75.552	1808	97	36.2
2	[1.690,0.740,1.001]	71.755	1800	110	41.1
3	[1.598,0.585,1.701]	72.041	1800	69	25.8
4	[1.715,0.637,0.928]	71.809	1800	64	24.0
5	[1.566,0.526,2.190]	73.365	1800	81	30.3
6	[1.683,0.513,1.108]	71.089	1800	96	35.9
7	[1.542,0.820,2.381]	74.986	1801	67	25.0
8	[1.696,0.505,1.046]	71.220	1800	88	33.0
9	[1.618,1.433,1.325]	74.228	1800	75	27.9
10	[1.660,0.506,1.250]	71.504	1800	71	26.6

to enhance the efficiency of MPS for HD optimizations, is also adopted to compare with ARSM-ISES.

Comparison results for global exploration capability, *Nfe*, and algorithm execution time are shown in Tables 7, 8 and 9 respectively. The results of TR-MPS are collected from (Cheng and Wang 2012), which did not present any variation ranges and CPU time. The data in Table 7 manifests that ARSM-ISES obtains the best median optimal solutions for all the HD problems. The variation ranges show that ARSM-ISES finds better results using the same or even lower *Nfe*, and exhibits higher robustness than MPS.

From observation of Table 8, ARSM-ISES is the most efficient in terms of *Nfe*. For F16, the efficiency improvement of ARSM-ISES is not very evident, which only ranges from 10 to 30 %. The efficiency merit of ARSM-ISES becomes

more significant when HD problems get more complicated. When optimizing R10 and HD1, MPS always reaches the maximum *Nfe*. Compared with TR-MPS, the mean *Nfe* needed by ARSM-ISES is reduced by 59 and 80 % for R10 and HD1 respectively. More importantly, even the maximum *Nfe* from using ARSM-ISES is still fewer than the mean *Nfe* of TR-MPS.

As can be seen from Table 9, ARSM-ISES needs less than one minute to execute the algorithm for HD problems. For F16, MPS uses more than 10 times as much CPU time as ARSM-ISES. Moreover, even under the limitation of maximum *Nfe*, MPS still needs 2–3 orders of magnitude more CPU time compared with ARSM-ISES when solving HD1 and R10 problems. Although the algorithm execution time of TR-MPS is not reported, ARSM-ISES is believed to be more time efficient than TR-MPS, because TR-MPS uses two dynamic MPS processes.

From the aforementioned comparative studies on the test problems, ARSM-ISES shows great promises for solving HD problems. More rigorous tests will be performed in future work to further examine the performance of ARSM-ISES for HD problems.

4.4 Testing ARSM-ISES on engineering benchmark problems

Two well-known engineering benchmark problems are used to test the performance of ARSM-ISES to solve expensive constrained problems. The spring design problem (SD) aims at minimizing the weight of a tension/compression spring subject to four performance constraints. Another pressure vessel design problem (SVD) is to minimize the total cost of

Fig. 9 Convergence histories of lifting surface optimization using ARSM-ISES. **a** Objective function, **b** Constraint violation, **c** Penalty factor, **d** Relative error of RSM

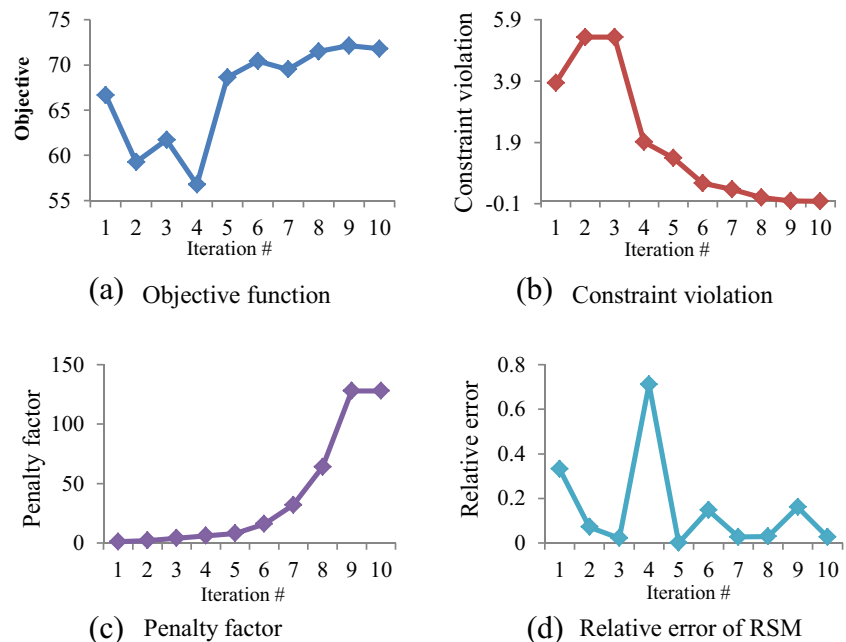


Table 12 Summary of optimization results obtained by using SQP

Run #	Optimal design	Optimal weight (kg)	Optimal flutter speed	Nfe	CPU time (min)
1	[1.529,2.610,2.220]	81.769	1800	530	195.4
2	[1.155,1.851,1.112]	57.809	1565	8	3.0
3	[0.670,1.866,2.134]	47.254	1302	8	3.0
4	[1.620,0.576,1.479]	71.278	1800	606	246.4
5	[0.600,2.092,2.316]	45.743	1232	8	3.0
6	[0.949,1.441,2.167]	54.690	1480	8	3.0
7	[1.646,1.369,1.159]	73.967	1800	514	191.0
8	[1.857,1.351,0.567]	78.089	1800	604	225.6
9	[0.666,2.080,1.332]	42.186	1210	8	3.0
10	[1.320,1.454,1.231]	62.793	1659	8	3.0

the pressure vessel subject to six performance constraints. The formulations of SD and SVD problems are also presented in [Appendix](#). More details about SD and SVD problems can be found in the article (Kazemi et al. 2011). All the constraints in both problems are assumed to be computationally expensive. Thus, besides using Nfe to count the number of objective evaluations, the number of constraint evaluations (Nce) is also recorded to measure the efficiency. Moreover, maximum constraint violation (MCV) (i.e., the maximum value of all the constraints) at the optimal solution is used to check the feasibility. A positive MCV indicates an infeasible solution, in contrary, a solution with non-positive MCV is feasible.

Thirty runs of ARSM-ISES are performed on each problem. Table 10 summarizes the optimization results obtained by ARSM-ISES, as well as by CiMPS that is an extended variant of MPS specially designed for optimizations with expensive objective and expensive constraints (Kazemi et al. 2011). Since Kazemi et al. (2011) only reported the best solution from 30 CiMPS runs for each problem, the best solutions found by ARSM-ISES are used here for comparison. As can be seen from Table 10, for both problems, ARSM-ISES finds

comparable feasible optimal solutions with those of CiMPS. Although CiMPS needs fewer objective evaluations (i.e., Nfe), ARSM-ISES considerably reduces the constraint evaluations (i.e., Nce) compared with CiMPS. Considering both Nfe and Nce , the total computational cost of ARSM-ISES is much cheaper than CiMPS.

To further discuss the performance of ARSM-ISES for expensive constrained problems, the average values (including optimal solution, Nfe , Nce , and MCV) are also listed in Table 10. First, all the 30 runs of ARSM-ISES successfully reach the feasible solutions, since all the MCV is negative or less than the constraint violation tolerance (0.001). The mean values of Nfe and Nce are even smaller than those to obtain the best solutions. However, the mean values of optimal solution attained by the proposed method have certain deviations from the best solutions. In those experiments, ARSM-ISES occasionally converges to conservative feasible solutions with small values of MCV (e.g., -0.04). Hence, the current ARSM-ISES incorporated with the adaptive penalty function has a limitation that the searching process may be trapped in some feasible but sub-optimal solutions. To overcome this limitation, a more elaborated approach to update the penalty factor is required.

Table 13 Summary of optimization results obtained by using GA

Run #	Optimal design	Optimal weight (kg)	Optimal flutter speed	Nfe	CPU time (min)
1	[1.641,0.594,1.720]	73.615	1825	1656	625.6
2	[1.821,0.500,0.738]	73.871	1816	1656	629.0
3	[1.662,0.609,1.226]	71.467	1803	1656	625.6
4	[1.861,0.580,0.726]	75.666	1833	1656	623.6
5	[1.710,0.521,1.014]	71.561	1804	1656	630.4
6	[1.666,0.595,1.215]	71.472	1804	1656	621.6
7	[1.708,0.574,1.214]	72.923	1824	1656	623.4
8	[1.682,0.609,1.136]	71.680	1806	1656	624.6
9	[1.698,0.564,1.120]	71.963	1811	1656	628.2
10	[1.716,0.624,0.925]	71.712	1800	1656	623.4
11	[1.711,0.764,1.500]	75.648	1852	125(263)	109.2

4.5 Engineering application with ARSM-ISES

In this section, an engineering application to optimize a lifting surface of a hypersonic aircraft is employed to demonstrate the capability of ARSM-ISES in solving real-world expensive engineering optimization problems. The profile of the lifting surface referred to the work by McNamara et al. (2008) is shown in Fig. 8. This optimization problem can be formulated in (8). The design objective is to minimize the weight of the lifting surface W subject to a constraint of the flutter speed V_f . Three internal structure parameters of the lifting surface are defined as design variables, including skin thickness, sizing parameters of beams and ribs as shown in Fig. 8. In this problem, it is assumed that the aerodynamic profile of this lifting surface is invariable. And a finite element analysis (FEA) model consisting of 956 shell elements and 540 nodes is constructed to evaluate the weight and flutter speed. Aluminum is used as the material. Since both the objective and constraint are evaluated based on the FEA model, we can use this application to further verify the capability of ARSM-ISES in solving engineering optimization problems involving computationally intensive objective and constraints. This optimization is executed on DELL OptiPlex 380 computer equipped with Core 2 Duo CPU (2.93 GHz) and 2G memory. ARSM-ISES is consecutively performed 10 runs to reduce random variations. The optimization results of ARSM-ISES are reported in Table 11.

$$\begin{aligned} \min \quad & W \\ \text{s.t.} \quad & V_f \geq 1800 \text{ m/s} \\ & 0.5 \text{ mm} \leq x_{1,2,3} \leq 3 \text{ mm} \end{aligned} \quad (8)$$

As can be seen from Table 11, ARSM-ISES has successfully found feasible optimal designs with limited Nfe for all the 10 trials. since the objective and constraint are embedded in the same FEA simulation, thus here Nfe is the total number of function calls for evaluating both the objective and constraint. Moreover, except for the 1st run, the optimal designs in the rest runs exactly locate on the boundary of the active

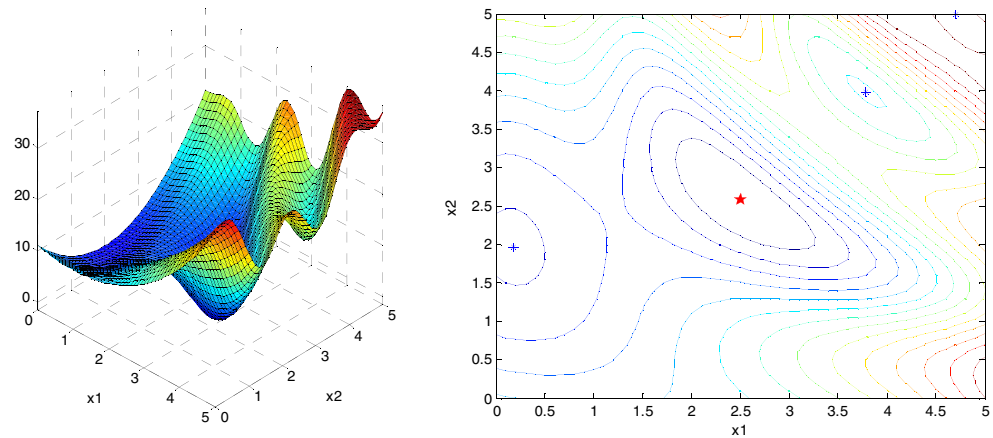
inequality constraint $V_f=1800$. Therefore the results of Table 11 manifest that the proposed adaptive penalty function method is effective.

We take the 4-th run to explain the optimization process of ARSM-ISES. Convergence histories of some important and interesting parameters are given in Fig. 9. Figure 9a shows that the objective decreases at the first four iterations to improve the optimality, and then it is pulled up to ensure feasibility. Such a trend of objective convergence well conforms to the basic behavior of classic penalty methods used in nonlinear constrained optimizations. Figure 9b exhibits the constraint violation computed based on the normalized flutter speed. It is evident that the proposed adaptive penalty function method forces constraint violation to decrease gradually as the iteration continues. At the last two iterations, constraint violation becomes slightly negative, which indicates the feasible optimal design has been found. The convergence curve of Fig. 9c shows that the penalty factor is continuously increased for the first nine iterations, while it keeps constant for the last cycle when a feasible design has been achieved at the 9th iteration. This is attributed to the mechanism of not increasing the penalty factor when maximum constraint violation is acceptable as described in (7). As can be seen from Fig. 9d, the relative error of RSM at each pseudo-optimum shows an overall decline trend and finally become acceptable when ARSM-ISES terminates. However, in Fig. 9d three significant rises can be observed at the 4-th, 6-th and 9-th iterations respectively. That phenomenon is caused by an inherent feature of ISES algorithm. When the fitting quality of RSM is good enough, SDS identification algorithm has to enlarge the size of next SDS which may increase the relative error of RSM at the following iteration.

Because the theoretical global optimal design of lifting surface optimization problem is unknown, strictly speaking, the results of Table 11 only prove that ARSM-ISES can converge to a feasible and improved design. To collect more proofs, we have to employ other well-known and widely applied algorithms for comparison, whose numerical convergence properties have been strictly proved. Since MPS, EGO,

Table 14 Comparison results with EARSIM using standard SDS approach

Func.	Global optimum obtained				Number of function evaluations (Nfe)			
	EARSIM		ARSM-ISES		EARSIM		ARSM-ISES	
	Max.	Median	Max.	Median	Max.	Mean	Max.	Mean
SC	2.112	-1.031	-1.030	-1.032	196	103.6	38	31.7
BR	0.398	0.398	0.399	0.398	266	113.4	58	39.8
RS	-1.031	-1.758	-1.395	-2.000	84	57.4	39	31.2
GF	1.064	0.000	0.003	0.000	546	294	74	57
GP	840.726	3.001	30.032	3.004	147	107.8	64	37.1
HN	-1.488	-3.322	-3.193	-3.322	790	774.3	288	188.6

Fig. 10 Illustration of SE function

and MSEG0 are not directly applicable to this problem due to the expensive constraint, sequential quadratic programming (SQP) (*fmincon* function) and GA (*ga* function) provided by MATLAB are used for following comparative studies. The default parameters are used to configure *fmincon* and *ga* functions. Ten runs of each algorithm are performed. In each run of SQP, the initial design is randomly generated inside the design space. The optimization results by using SQP and GA are respectively presented in Tables 12 and 13.

The results in Table 12 reveal that only four runs of SQP converge to a feasible design, while the rest runs abnormally terminate at infeasible results. In our opinion, non-smooth or even discontinuous behavior of the flutter speed constraint results in the six failures of SQP optimization. The fact that 10 runs of ARSM-ISES can converge to feasible optimal designs indicates that the proposed algorithm is more robust and provides favorable capability of solving non-smooth optimizations. The results of the four normally converged SQP optimization are different from each other, which indicates that this problem is multimodal. The 1st run in Table 12 produces a nearly 15 % larger optimal weight than that of the 4-th run of ARSM-ISES since SQP is rather sensitive to the starting point. On the contrary, the largest difference among the ten optimal weights obtained by ARSM-ISES is only 5 %, which demonstrates its better global exploration capability and robustness over SQP. Through comparing the mean values of *Nfe* and CPU time of the four successful SQP runs, we can find that ARSM-ISES can save both of them by more than 60 %.

As can be seen from Table 13, although GA can converge to a feasible optimal design in each run, much more *Nfe* and CPU time is required to perform GA optimization. Compared with ARSM-ISES, GA uses almost 20 times more *Nfe* and CPU time to solve this problem. Since GA always stops when the maximum number of generations is reached, the same *Nfe* for each run is achieved. Furthermore, the flutter speed constraints at optimal designs are more conservative, which do not locate on the boundary of the feasible region. Additionally, to achieve a fair comparison, the population size and maximum

generation number are adjusted to terminate GA when the number of objective evaluations exceeds the maximum *Nfe* of ARSM-ISES (i.e., 110). The last row in Table 12 shows the median value among ten runs, and in the 5-th column the number in the parentheses indicates the constraint evaluations. One can find that the optimal weight from the GA within the limited *Nfe* is larger than that of ARSM-ISES, probably since constraint is still not activated at the final solution.

To sum up, ARSM-ISES performs much better than SQP and GA algorithms when solving this lifting surface optimization in terms of global exploration capability, optimization efficiency, and robustness. Moreover, the results and analysis also manifest that ISES and adaptive penalty method make ARSM-ISES applicable to solving non-smooth, multimodal engineering optimization problems with expensive constraints.

5 Discussion

5.1 Distinctions from the trust region method and standard SDS approach

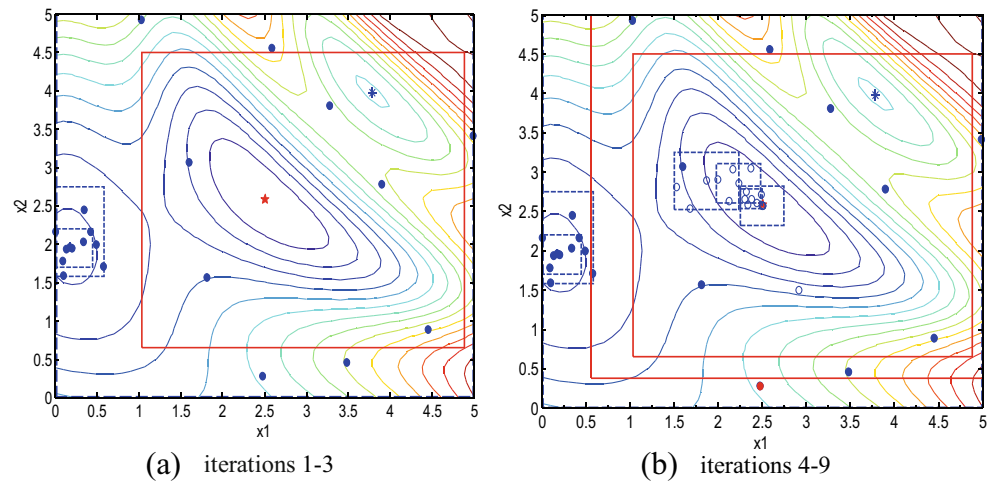
The concept of SDS used in ISES is somewhat similar to the trust region framework. The main distinctions from trust region can be explained as follows.

- (1) Update of SDS is driven by the optimality and metamodel accuracy, where the center is determined in terms of optimality and size is controlled by the fitting

Table 15 Optimal points and function values of SE function

ID	Optimal point	Function value
1	(2.504,2.578)	-1.457
2	(0.176,1.972)	2.866
3	(3.783,3.981)	12.689
4	(4.710,5.000)	33.242

Fig. 11 Illustration of SDS' and sequential sampling in SE optimization. **a** iterations 1–3, **b** iterations 4–9



- quality. Whereas, the size of the trust region is determined in terms of the predicted improvement instead of fitting quality.
- (2) The space contraction and expansion operators are different from those in the trust region method. The space operation operators are believed to be more elaborated in ISES. Moreover, the unexplored SDS concept makes ISES much stronger than the trust region method in seeking global optimal solutions.
 - (3) Besides SDS identification, ISES includes IMS-LHD sampling scheme and tailored termination criteria, which are beyond the basic trust region method.

Although ISES is derived based on the standard SDS concept, most of powerful techniques in ISES including IMS-LHD sampling scheme, new space operation operators, unexplored SDS, and novel termination criteria are newly developed in this work. For further demonstration, EARSIM using the standard SDS approach (Long et al. 2012a) is performed 10 runs on each testing function for comparison. The results summarized in Table 14 show that due to the use of ISES, ARSM-ISES outperforms EARSIM for all the testing problems.

5.2 Illustration of ARSM-ISES optimization process

Carpet plot and contour plot depicted in Fig. 10 show that the SE function has three local optima (indicated by asterisks) and the global optimal is (2.504, 2.578), as indicated by the pentagram. Table 15 details all the optimal points and corresponding objective values of SE.

According to our experience, SE function is hard to be solved. The search is likely trapped into the 2nd local optimal point in Table 15. We also have tried GA (ga function in MATLAB with the default settings) to optimize the SE function. However, among 1000 independent experiments, only 188 runs succeed in finding the true global optimal and the rest

812 runs are all trapped at the point (0.176, 1.972). We use SE to illustrate the ARSM-ISES optimization process for discussion.

Figure 11 intuitively demonstrates detailed information of SDS' and sequential sampling in SE optimization. The filled circles stand for the samples generated at iterations 1–3, and the hollow circles indicate samples produced at iterations 4–9. Regular SDS' are marked by dotted rectangles or squares. Dotted rectangles and squares in Fig. 11a indicate the SDS' identified at iterations 1–3, while the dotted rectangles and squares increased in Fig. 11b are those at iterations 4–9. The solid square stands for the unexplored SDS. From Fig. 11a, it is obvious that sequential samples cluster around the local optima (0.176, 1.972), which leads ARSM-ISES optimization to rapidly achieve the local convergence. Without the assistance of the unexplored SDS operation and the specific termination criteria, it is impossible for ARSM-ISES to reach the global extreme in this run. However, Fig. 11a shows that the unexplored SDS determined by the unexplored samples covers the true global optimal point, which provides ARSM-ISES opportunities to escape from the local optima.

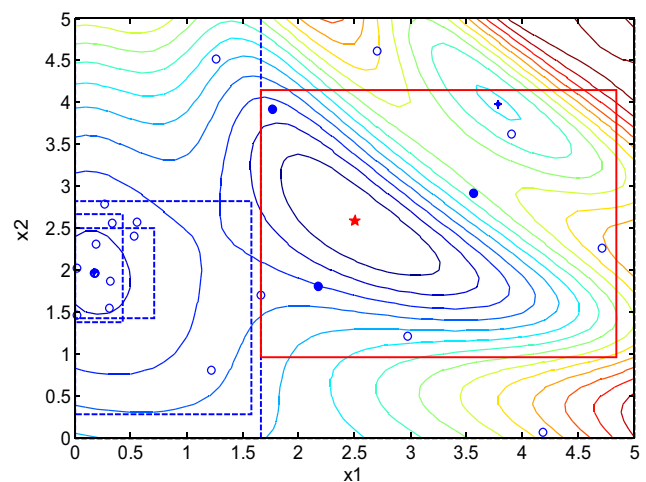


Fig. 12 Failure of finding global optimum in SE optimization

Table 16 Values of α_{ij} , c_i , and p_{ij}

i	$\alpha_{ij}, j=1,2,\dots,6$					c_i	$p_{ij}, j=1,2,\dots,6$						
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Figure 11b illustrates that inside the unexplored SDS some more sequential samples are assigned toward the global optimum. At the 8-th iteration, another unexplored SDS which is even larger than that at the 3rd iteration is constructed. Since the true global optimum has been found, this unexplored SDS has not effect on the optimization process. One can find that the unexplored SDS can cover untapped regions and become larger as iteration increases. However, the current unexplored SDS identification algorithm mainly focuses on the central region; the regions near the boundaries, especially near the corners may be not well covered by the unexplored SDS'.

Except for visualizing the benefit of using unexplored SDS, ARSM-ISES algorithms with and without unexplored SDS operations are respectively performed 1000 runs on SE for comparison. Without unexplored SDS, the success ratio of finding the true global optimum is 74 %, which is less than that (87 %) of using unexplored SDS. Therefore, the use of unexplored SDS is proved to enhance global exploration capability of ARSM-ISES. Moreover, the success ratio of ARSM-ISES is 3.5 times more than that (19 %) of GA.

5.3 Features of ARSM-ISES

In addition to fully inheriting the advantages of standard ARSM, ARSM-ISES shows other appealing merits. The main distinguishing features of ARSM-ISES are summarized as below.

- (1) ARSM-ISES ensures to converge to a local optimum no matter what initial samples are used for starting.
- (2) The novel ISES provides ARSM-ISES with a good global exploration capability to reduce the probability of missing global optima.
- (3) Since no global optimization is required in ISES, which only employs simple logical judgment and efficient local search algorithm, ARSM-ISES shows favorable high efficiency including fewer Nfe and less algorithm execution overhead. Besides, ARSM-ISES also shows promising performance for solving HD optimization problems.
- (4) The proposed adaptive penalty function method extends the application scope of ARSM-ISES for engineering optimization problems involving expensive constraints.
- (5) ARSM-ISES is capable of successfully solving non-smooth and even discontinuous optimization problems.

- (6) All the novel techniques in ISES can be independently used and integrated with other MBDO algorithms.

5.4 Limitations of ARSM-ISES

Although the favorable performance of ARSM-ISES has been demonstrated, we still observed that ARSM-ISES may miss the true global optima for some problems. Here, a visualization of a failed global exploration process on SE function (as shown in Fig. 12) is presented to reveal the cause.

Figure 12 illustrates that though the regular SDS' (indicated by dotted squares and rectangles) lead the sequential samples (indicated by hollow circles) to gather around the local optimum, the unexplored SDS (marked by solid square) still effectively covers the global optimal, more importantly, three samples (indicated by filled cycles) even locate inside the single-modal valley whose bottom is exactly the global optimum of SE. Based on so significant information, ARSM-ISES should have found the global optimal, unfortunately, it failed. The poor approximation capability of RSM metamodel is considered to be the major cause for this failure. Even though some samples locate close to the global optimum, RSM may not correctly capture the trend of the true function if the design space is not small enough.

In fact, the cooperation of several techniques including the expansion operation of SDS, construction of unexplored SDS, IMS-LHD sampling method, and the tailored termination criteria makes ARSM-ISES have a favorable global convergence capability. However, it is worth noting that ARSM-ISES still may be trapped into local optima for particular problems.

Although the proposed algorithm shows better performance compared with several existing methods, the Nfe needed to perform HD optimizations still dramatically increases. Besides the well-known "curse of dimensionality", there are two facts causing the efficiency decay. First, the minimum number of samples required to rebuild RSM tremendously increases as the dimensionality increases. Second, it is found that the total number of iterations for HD problems is obviously larger than those for LD problems. For instance, for LD problems ARSM-ISES usually spends about 10 iterations before

convergence, whereas the number increases to more than 50 for HD problems. Therefore, how to further improve the optimization efficiency of the proposed algorithm for HD problems is expected in future work.

6 Conclusions and future work

In this article, a promising intelligent space exploration strategy consisting of IMS-LHD sampling scheme, significant design space (SDS) identification algorithm, and tailored termination criteria is developed. Independent from global optimization subroutines, ISES only invokes computationally inexpensive processes, such as logical judgments and local optimization for space reduction and metamodel management. Based on ISES, an efficient adaptive response surface method (ARSM), ARSM-ISES, is proposed for expensive engineering optimizations. Since no global optimization procedure is required as in the original ARSM, algorithm execution overhead of ARSM-ISES is dramatically reduced. The use of unexplored SDS and specific termination criteria improves the global exploration capability of ARSM-ISES. A number of multimodal numerical benchmark problems are used to test and validate the performance of our proposed algorithm. The comparison results demonstrate that in terms of global exploration capability, optimization efficiency, and robustness, ARSM-ISES outperforms several existing MBDO algorithms including ARSM, IARSM, MPS, EGO, MSEG, and TR-MPS. It is also found that the execution time of MBDO algorithms significantly affects the overall efficiency especially for high dimensional problems. Moreover, an adaptive penalty function method is proposed to enable ARSM-ISES to be applicable to engineering optimizations involving expensive constraints. The capability of handling expensive constraints has been demonstrated through comparative studies with CiMPS, SQP, and GA. Eventually, visual illustrations of ARSM-ISES optimization processes are presented to intuitively discuss its unique features and limitations.

Future work includes further improvement of the overall performance of ARSM-ISES. First, other interpolation metamodels (e.g., RBF and Kriging) will be tested to approximate the expensive function inside an unexplored SDS. Second, we will develop a distributed computing framework to enable the simultaneous computation of ARSM-ISES. Last, ISES can be generalized as a common framework of managing metamodels for developing other MBDO algorithms.

Acknowledgment This work is sponsored by the National Natural Science Foundation of P. R. China (Grant No. 51105040 and Grant No.

11372036), Aeronautical Science Foundation of China (Grant No. 2011ZA72003), Excellent Young Scholars Research Fund of Beijing Institute of Technology (Grant No. 2011CX0402), Fundamental Research Fund of Beijing Institute of Technology (Grant No. 20130142008), and Natural Science and Engineering Research Council (NSERC) of Canada. The authors would like to appreciate Dr. Viana's online SURROGATES toolbox.

Appendix

Sasena function (SE)

$$f_{SE} = 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7\sin(0.5x_1)\sin(0.7x_1x_2) \quad (9)$$

Peaks function (PK)

$$f_{PK} = 3(1 - x_1)^2 e^{-x_1^2 - (x_2 + 1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) e^{-x_1^2 - x_2^2} - \frac{1}{3} e^{-(x_1 + 1)^2 - x_2^2} \quad (10)$$

Six-hump Camelback function (SC)

$$f_{SC} = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \quad (11)$$

Branin function (BR)

$$f_{BR} = \left(x_2 - 5.1(x_1/2\pi)^2 + 5x_1/\pi - 6\right)^2 + 10(1 - 1/(8\pi)\cos(x_1) + 10) \quad (12)$$

Rastrigin function (RS)

$$f_{RS} = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2) \quad (13)$$

Generalized polynomial function (GF)

$$f_{GF} = u_1^2 + u_2^2 + u_3^2 \\ u_i = c_i - x_1(1 - x_2^i), \quad i = 1, 2, 3. \\ c_1 = 1.5, \quad c_2 = 2.25, \quad c_3 = 2.625 \quad (14)$$

Goldstein and Price function (GP)

$$f_{GP} = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \\ \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right] \quad (15)$$

Griewank function (GN)

$$f_{GN} = \sum_{i=1}^2 \frac{x_i^2}{200} + \prod_{i=1}^2 \cos\left(x_i/\sqrt{i}\right) + 1 \quad (16)$$

Hartman function (HN)

$$f_{HN} = -\sum_{i=1}^4 c_i \exp \left[-\sum_{j=1}^6 \alpha_{ij} (x_i - p_{ij})^2 \right], \quad i = 1, 2, \dots, 6 \quad (17)$$

where, α_{ij} , c_i , and p_{ij} are listed in Table 16

10D SUR-T1-14 function (HD1)

$$f_{HD1} = (x_1 - 1)^2 + (x_{10} - 1)^2 + 10 \sum_{i=1}^9 (10 - i) (x_i^2 - x_{i+1})^2, \quad i = 1, 2, \dots, 10 \quad (18)$$

10D Rosenbrock function (R10)

$$f_{R10} = \sum_{i=1}^9 \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right) \quad (19)$$

16 Variable Function (F16)

$$f_{F16} = \sum_{i=1}^{16} \sum_{j=1}^{16} a_{ij} (x_i^2 + x_i + 1) (x_j^2 + x_j + 1) \quad (20)$$

Where, the values of a_{ij} can be found in reference (Wang et al. 2004)

Spring design problem (SD)

$$\begin{aligned} \min \quad & f_{SD}(x) = (x_3 + 2)x_2x_1^2 \\ \text{s.t.} \quad & g_1(x) = 1.0 - \frac{x_2^3x_3}{71875x_1^4} \leq 0; \quad g_2(x) = \frac{x_2(4x_2 - x_1)}{12566x_1^3(x_2 - x_1)} + \frac{2.46}{12566x_1^2} - 1.0 \leq 0 \\ & g_3(x) = 1.0 - \frac{140.54x_1}{x_2^3x_3} \leq 0; \quad g_4(x) = \frac{x_2 + x_1}{1.5} - 1.0 \leq 0 \\ & 0.05 \leq x_1 \leq 0.20; \quad 0.25 \leq x_2 \leq 1.30; \quad 2.00 \leq x_3 \leq 15.00 \end{aligned} \quad (21)$$

Pressure vessel design problem (PVD)

$$\begin{aligned} \min \quad & f_{PVD}(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\ \text{s.t.} \quad & g_1(x) = -x_1 + 0.0193x_3 \leq 0; \quad g_2(x) = -x_2 + 0.00954x_3 \leq 0 \\ & g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0; \quad g_4(x) = x_4 - 240 \leq 0 \\ & g_5(x) = 1.1 - x_1 \leq 0; \quad g_6(x) = 0.6 - x_2 \leq 0 \\ & 1.0 \leq x_1 \leq 1.375; \quad 0.625 \leq x_2 \leq 1.0; \quad 25 \leq x_3 \leq 150; \quad 25 \leq x_4 \leq 240 \end{aligned} \quad (22)$$

References

- Alexandrov NM, Dennis JE, Lewis RM, Torczon V (1998) A trust-region framework for managing the use of approximation models in optimization. *Struct Optim* 15:16–23
- Bichon BJ, Eldred MS, Mahadevan S, McFarland JM (2013) Efficient global surrogate modeling for reliability-based design optimization. *J Mech Des* 135(1):011009. doi:10.1115/1.4022999
- Cheng G, Wang GG (2012) Trust region based MPS method for global optimization of high dimensional design problems. Paper presented at the 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference, Honolulu, Hawaii, 23–26 April 2012. doi:10.2514/6.2012-1590
- Duan X, Wang GG, Kang X, Niu Q, Naterer G, Peng Q (2009) Performance study of mode-pursuing sampling method. *Eng Optim* 41(1):1–21. doi:10.1080/03052150802345995
- Forrester AIJ, Keane AJ (2009) Recent advances in surrogate-based optimization. *Prog Aerosp Sci* 45(1–3):50–79. doi:10.1016/j.paerosci.2008.11.001
- Gano SE, Renaud JE, Martin JD, Simpson TW (2006) Update strategies for kriging models used in variable fidelity optimization. *Struct Multidiscip Optim* 32(4):287–298. doi:10.1007/s00158-006-0025-y
- Jin R, Chen W, Simpson TW (2001) Comparative studies of metamodeling techniques under multiple modeling criteria. *Struct Multidisc Optim* 23:14
- Jin R, Chen W, Sudjianto A (2005) An efficient algorithm for constructing optimal design of computer experiments. *J Stat Plan Infer* 134(1):268–287. doi:10.1016/j.jspi.2004.02.014
- Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. *J Glob Optim* 13(4):455–492. doi:10.1023/a:1008306431147
- Kazemi M, Wang GG, Rahnamayan S, Gupta K (2011) Metamodel-based optimization for problems with expensive objective and constraint functions. *J Mech Des* 133(1):014505. doi:10.1115/1.4003035
- Lewis RM (1996) A trust region framework for managing approximation models in engineering. Paper presented at the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, 4–6 September 1996. doi:10.2514/6.1996-4101
- Li YL, Liu L, Long T, Dong WL (2013) Metamodel-based global optimization using fuzzy clustering for design space reduction. *Chin J Mech Eng* 26(5):928–939. doi:10.3901/cjme.2013.05.928
- Long T (2009) Research on methods of multidisciplinary design optimization and integrated design environment for aircrafts. Ph.D Dissertation, Beijing Institute of Technology
- Long T, Liu L, Peng L (2012a) Global optimization method with enhanced adaptive response surface method for computation-intensive design problems. *Adv Sci Lett* 5(2):881–887. doi:10.1166/asl.2012.1847
- Long T, Liu L, Peng L, Li Y (2012b) Aero-structure coupled optimization of high aspect ratio wing using enhanced adaptive response surface method. Paper presented at the 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, Indiana, 17–19 September 2012. doi:10.2514/6.2012-5456
- McNamara JJ, Friedmann PP, Powell KG, Thuruthimattam BJ, Bartels RE (2008) Aeroelastic and aerothermoelastic behavior in hypersonic flow. *AIAA J* 46(10):2591–2610. doi:10.2514/1.36711
- Panayi AP, Diaz AR, Schock HJ (2009) On the optimization of piston skirt profiles using a pseudo-adaptive response surface method.

- Struct Multidiscip Optim 38(3):317–330. doi:[10.1007/s00158-008-0295-7](https://doi.org/10.1007/s00158-008-0295-7)
- Parr JM, Keane AJ, Forrester AIJ, Holden CME (2012) Infill sampling criteria for surrogate-based optimization with constraint handling. Eng Optim 44(10):1147–1166. doi:[10.1080/0305215x.2011.637556](https://doi.org/10.1080/0305215x.2011.637556)
- Pérez VM, Renaud JE, Watson LT (2002) Adaptive experimental design for construction of response surface approximations. AIAA J 40(12):2495–2503. doi:[10.2514/2.1593](https://doi.org/10.2514/2.1593)
- Queipo NV, Haftka RT, Shyy W, Goel T, Vaidyanathan R, Kevin Tucker P (2005) Surrogate-based analysis and optimization. Prog Aerosp Sci 41(1):1–28. doi:[10.1016/j.paerosci.2005.02.001](https://doi.org/10.1016/j.paerosci.2005.02.001)
- Regis RG (2011) Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Comput Oper Res 38(5):837–853
- Regis RG (2014) Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Eng Optim 46(2):218–243
- Roux WJ, Stander N, Haftka RT (1998) Response surface approximations for structural optimization. Int J Numer Methods Eng 42(3):517–534
- Sasena MJ (2002) Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations. Ph.D Dissertation, Univ. of Michigan
- Sasena MJ, Parkinson M, Reed MP, Papalambros PY, Goovaerts P (2005) Improving an ergonomics testing procedure via approximation-based adaptive experimental design. J Mech Des 127(5):1006–1013. doi:[10.1115/1.1906247](https://doi.org/10.1115/1.1906247)
- Shan SQ, Wang GG (2010) Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Struct Multidiscip Optim 41(2):219–241. doi:[10.1007/s00158-009-0420-2](https://doi.org/10.1007/s00158-009-0420-2)
- Sharif B, Wang GG, ElMekkawy TY (2008) Mode pursuing sampling method for discrete variable optimization on expensive black-box functions. J Mech Des 130(2):021402. doi:[10.1115/1.2803251](https://doi.org/10.1115/1.2803251)
- Simpson TW, Mauery TM, Korte JJ, Mistree F (1998) Comparison of response surface and kriging models for multidisciplinary design optimization. Paper presented at the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, 2–4 September 1998. doi:[10.2514/6.1998-4755](https://doi.org/10.2514/6.1998-4755)
- Simpson TW, Booker AJ, Ghosh D, Giunta AA, Koch PN, Yang RJ (2004) Approximation methods in multidisciplinary analysis and optimization: a panel discussion. Struct Multidiscip Optim 27(5):302–313. doi:[10.1007/s00158-004-0389-9](https://doi.org/10.1007/s00158-004-0389-9)
- Simpson TW, Toropov V, Balabanov V, Viana FAC (2008) Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come—or not. Paper presented at the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia Canada, 10–12 September 2008. doi:[10.2514/6.2008-5802](https://doi.org/10.2514/6.2008-5802)
- Viana FAC, Venter G, Balabanov V (2010a) An algorithm for fast optimal Latin hypercube design of experiments. Int J Numer Methods Eng 82(2):135–156. doi:[10.1002/nme.2750](https://doi.org/10.1002/nme.2750)
- Viana FAC, Haftka R, Watson L (2010b) Why not run the efficient global optimization algorithm with multiple surrogates? Paper presented at the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Orlando, Florida, 12–15 April 2010. doi:[10.2514/6.2010-3090](https://doi.org/10.2514/6.2010-3090)
- Viana FAC, Haftka R, Watson L (2013) Efficient global optimization algorithm assisted by multiple surrogate techniques. J Glob Optim 56(2):669–689. doi:[10.1007/s10898-012-9892-5](https://doi.org/10.1007/s10898-012-9892-5), Available at <https://sites.google.com/site/felipeacviana/surrogatetoolbox>
- Wang GG (2003) Adaptive response surface method using inherited Latin hypercube design points. J Mech Des 125(2):210–220. doi:[10.1115/1.1561044](https://doi.org/10.1115/1.1561044)
- Wang GG, Shan S (2007) Review of metamodeling techniques in support of engineering design optimization. J Mech Des 129(4):370–380. doi:[10.1115/1.2429697](https://doi.org/10.1115/1.2429697)
- Wang GG, Simpson TW (2004) Fuzzy clustering based hierarchical metamodeling for design space reduction and optimization. Eng Optim 36(3):313–335
- Wang GG, Dong ZM, Aitchison P (2001) Adaptive response surface method—a global optimization scheme for approximation-based design problems. Eng Optim 33(6):707–733
- Wang LQ, Shan SQ, Wang GG (2004) Mode-pursuing sampling method for global optimization on expensive black-box functions. Eng Optim 36(4):419–438. doi:[10.1080/03052150410001686486](https://doi.org/10.1080/03052150410001686486), Available at <http://www.sfu.ca/~gwa5/software.html>
- Wang H, Li GY, Zhong ZH (2008) Optimization of sheet metal forming processes by adaptive response surface based on intelligent sampling method. J Mater Process Technol 197(1–3):77–88. doi:[10.1016/j.jmatprotec.2007.06.018](https://doi.org/10.1016/j.jmatprotec.2007.06.018)
- Ye KQ, Li W, Sudjianto A (2000) Algorithmic construction of optimal symmetric Latin hypercube designs. J Stat Plan Infer 90(1):145–159. doi:[10.1016/s0378-3758\(00\)00105-1](https://doi.org/10.1016/s0378-3758(00)00105-1)
- Zhu HG, Liu L, Long T, Peng L (2012a) A novel algorithm of maximin Latin hypercube design using successive local enumeration. Eng Optim 44(5):551–564. doi:[10.1080/0305215x.2011.591790](https://doi.org/10.1080/0305215x.2011.591790)
- Zhu HG, Liu L, Long T, Zhao JF (2012b) Global optimization method using SLE and adaptive RBF based on fuzzy clustering. Chin J Mech Eng 25(4):768–775. doi:[10.3901/cjme.2012.04.768](https://doi.org/10.3901/cjme.2012.04.768)
- Zhu HG, Liu L, Zhou SD, Li YL (2012c) Integrated aerodynamic thermal structure design optimization method of lifting surfaces. J Aircr 49(5):1521–1526. doi:[10.2514/1.c031464](https://doi.org/10.2514/1.c031464)