Trust-region Based Adaptive Radial Basis Function Algorithm for Global Optimization of Expensive Constrained Black-box Problems

Chengyang Liu[a], Zhiqiang Wan[b], Yijie Liu[c], Xuewu Li[d] and Dianzi Liu[a,d,*]

[a]Engineering Division, Faculty of Science, University of East Anglia, Norwich, UK
[b]School of Aeronautic Science and Engineering, Beihang University, Beijing, China
[c]Department of Engineering Mechanics, Guangzhou University, Guangzhou, China
[d]School of Mechanical Engineering, Xi'an University of Science and Technology, Xi'an, China

ARTICLE INFO

Keywords:
metamodel-based optimization
adaptive radial basis function
balanced trust region strategy
efficient constrained optimization

ABSTRACT

In spite of the continuous advancement in computer science and technology, the real-world engineering optimization process is still facing challenges due to the huge computational cost on simulations. To address this issue, TARBF algorithm (trust-region based adaptive radial basis function interpolation) for solving expensive constrained black-box optimization problems is proposed in this paper. The approach successfully decomposes the original optimization problem into a sequence of sub-problems approximated by radial basis functions in a series of trust regions. Then, the solution of each sub-problem becomes the starting point for the next iteration. According to the values of objective and constraint functions, an effective online normalization technique is further developed to adaptively improve the model accuracy in the trust region, where the surrogate is updated iteratively. Numerical experiments on 21 G-problems (CEC'2006) and 4 engineering problems prove that TARBF is a convergent paradigm, which can find more accurate solutions than other state-of-the-art metamodel-based algorithms within comparable computational budget. Moreover, the sophisticated trust region strategy developed in TARBF has the capability to facilitate an effective balance of exploration and exploitation for solving constrained black-box optimization problems.

1. Introduction

Physics-based simulation and optimization is absolutely imperative for the cost reduction required by smart and modern industry. In spite of continuous advancement in computer science, real-world engineering optimizations are still challeng-

*Corresponding authors
wzq@buaa.edu.cn (Z. Wan); dianzi.liu@uea.ac.uk (D. Liu)

ing due to the huge computational cost on simulations [1]. It is rarely practicable to utilize traditional global optimization methods, such as particle swarm optimization (PSO) [2] and genetic algorithms (GAs) [3], which require an awful lot of evaluations to seek an optimum. In addition, detailed finite element analysis model is usually a black-box, i.e., there is no explicit mathematical expressions of the design variables.

Faced with these limitations, surrogate modeling techniques that manage to imitate the behavior of high-fidelity model have won critical acclaim in recent years. Given a dataset containing vectors of design variables together with their responses, a metamodel can be built and used for further analysis. It is particularly beneficial to optimization because multiple calls for expensive functions can be replaced by calls for comparatively cheap metamodels.

Generally, methods for metamodel building include polynomial regression (PS) [4], radial basis function (RBF) [5], kriging [6], multivariate adaptive regression splines (MARS) [7], artificial neural networks (ANN) [8] and support vector regression (SVR) [9]. Comparative studies have been made over the past years. For example, Jin et al. [10] investigated and compared PS, Kriging, MARS and RBF models. The authors claimed that RBF metamodel outperforms others in most instances, especially in shortage of computational resource (sample size is small). With the increased fitting points, the performance of Kriging and MARS models will gain improvement. In addition, Kriging is sensitive to the noise but PR performs well in this situation. In a comparison with PR, Kriging, MARS and RBF metamodels, Clarke et al. [9] found that SVR has the overall performance in regard to accuracy and robustness with the manually optimized Guassian kernel function. In contrast, Kim et al. [11] compared moving least squares method (MLS), Kriging, RBF and SVR metamodels and concluded that Kriging and MLS are able to build more accurate metamodels than RBF and SVR models. Therefore, it is impractical to draw any decisive conclusions on the superiority of any of the mentioned metamodels. The quality of a metamodel can vary considerably depending on how many design variables are involved, what types of the fitting functions are, and how well the predefined parameters are tuned.

Once the metamodel has been built, optimization methods can then be applied to seek for the optimum, which is therefore referred as metamodel-based or surrogate-assisted design optimization (MBDO) [12, 13]. MBDO has gained continuous development over the past two decades. For example, John et al. [14] proposed efficient global optimization (EGO), which employs Kriging metamodel. The optimization progress is guided by both the prediction and error estimations.

Younis et al. [15] also built Kriging metamodel for objective functions but a region reduction and elimination strategy is used to locate the global optimum. An application of RBF interpolation can be found in [16], which solved sub-problems in successive trust regions. In addition, metamodels have been widely incorporated into evolutionary algorithms [17, 18, 19] and particle swarm optimization [20, 21, 22] for solving expensive black-box problems.

Although the above-mentioned algorithms can obtain good results on black-box problem with boundary constraints, most of them have difficulty dealing with nonlinear constrained optimization problems. As stated by Haftka et al. [23] and Muller et al. [24], the state of the art about constrained optimization is less advanced. The main challenge lies in the definition of a general convergent scheme that seeks a feasible and optimized solution under a reasonable number of function evaluations. The hurdles become even more distinct as the number of constraints increases or when binary and discontinuous responses are present [25]. One common approach is to transform the constrained problem to a unconstrained problem by using a penalty function (penalizing the fitness value of infeasible solutions), as shown in [26, 27, 28]. However, the information of the individual constraint is lost and the additional penalty parameters need to be well tuned for different problems [29]. As a result, this penalty-based technique does not work well in solving complex constrained optimization problems. Recently, there are various novel techniques for tackling expensive constrained black-box optimization problems. For example, Brekelmans [30] applied linear approximation within sequential trust regions and used a filter method to select current iterates. Basudhar et al. [31] developed a constrained EGO, where SVM is applied to approximate the boundary of the feasible region and to support the expected improvement (EI). Regis [32] proposed the ConstrLMSRS that uses RBF surrogate to model objective and constraint functions separately. A feasible starting point is necessary in ConstrLMSRS because further candidate points are generated by perturbing all or a fraction of the coordinates of the current best feasible solution. In addition, Regis [33] developed another RBF-assisted algorithm named as COBRA, where a new iterate is selected according to the distance from previous points. Recently, a self-adjusted version of COBRA was proposed by Bagheri et al. [34] to avoid tuning the parameters manually. Dong [35] presented a kriging-based constrained global optimization algorithm SCGOSR with space reduction strategy. In SCGOSR, new added samples are selected from optimal solutions obtained by the multi-start solver.

However, relatively few algorithms can handle expensive constrained black-

box optimization problem under severely limited budget. Jiao et al. [36] introduced a self-adaptive selection strategy into the evolutionary algorithm which combines feasibility with multi-objective problem techniques. 22 G-problems [37] have been tested and some of them (G05, G06, G08, G11, G12, G18) can be solved very efficiently in less than 1000 evaluations, but others require 5000-100000 evaluations to be solved. COBRA [33], ConstrLMSRS [32] and KCGO [38] can obtain feasible solutions on some G-problems within hundreds or thousands of function evaluations but are not competitive with regard to precision and optimality. Zahara and Kao [39] tested G04, G08, G12, which can be solved within 20000 evaluations. Bagheri's SACOBRA [34] is very efficient that 11 G-problems can be solved within 500 function evaluations under a relaxed threshold.

In this paper, a novel trust-region based adaptive radial basis function algorithm (TARBF) is proposed for solving expensive constrained black-box problems especially under very limited budget. A description of the constrained black-box optimization problem is given in Section 2. Next, an overview of the TARBF framework is presented in Section 3.1. The following subsections will give details such as the novel design of experiments in Section 3.2, the adaptive online normalization strategy in Section 3.3, the metamodel building strategy using radial basis functions in Section 3.4, the moving trust region strategy in Section 3.5, and the early termination strategy in Section 3.6. In Section 4, the experimental and comparison results of TARBF on 25 benchmark problems are given and conclusions are drawn in Section 5.

2. Constrained black-box optimization (CBO) problem

The constrained black-box optimization (CBO) problem addressed in this paper can be formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}\in Q} f(\boldsymbol{x}) \\ &\text{s.t. } g_j(\boldsymbol{x}) \le 0\ (j = 1, \dots, m) \\ &\quad A_i \le x_i \le B_i\ (i = 1, \dots, d) \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ refers to the vector of design variables; Q is the design space bounded by $[\boldsymbol{A}, \boldsymbol{B}]$, A_i and B_i are the given lower and upper bounds on the design variable x_i; d is the total number of the design variables; $f(\boldsymbol{x})$ is the objective function; $g_j(\boldsymbol{x})(j = 1, \dots, m)$ is the constraint function and m is the total number of the constraint functions.

The main characteristic of a CBO problem is that there are no algebraic expressions of both the objective and the constraint functions. In other words, the

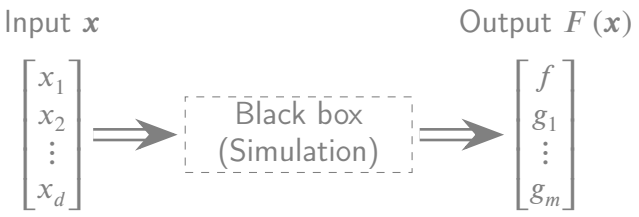

Figure 1: Black-box simulation

functional relationship between the design variables and the response is implicit. It is quite common in engineering when simulation models are used. As illustrated in Fig 1, a set of design variables $\boldsymbol{x} \in \boldsymbol{Q} \subseteq \mathbb{R}^d$ are inputted to the black-box, e.g., a simulation tool, and a certain set of responses $\boldsymbol{F}(\boldsymbol{x}) \subseteq \mathbb{R}^{m+1}$ are the outputs based on the unknown relationship between the variables and responses. In this paper, we assume the CBO problem is computationally expensive and the values $f(\boldsymbol{x})$ and $g_j(\boldsymbol{x})\,(j = 1, ..., m)$ for any input $\boldsymbol{x} \in [\boldsymbol{A}, \boldsymbol{B}]$ can be obtained without any crash on the simulator. Besides, the derivative information of any response is also unavailable or impractical to obtain.

3. Trust-region based adaptive radial basis function interpolation algorithm (TARBF)

3.1. Overview

TARBF is a trust-region based iterative method, which attempts to solve a sequence of constrained optimization sub-problems by using the radial basis function interpolation of the objective and constraint functions in a series of regions of interest (trust region). In this way, TARBF replaces the original optimization problem (Equation 1) by a succession of approximate subproblems as

$$\begin{aligned} \min_{\boldsymbol{x}\in\boldsymbol{Q}^k} \quad & \tilde{f}^k(\boldsymbol{x}) \\ s.t. \quad & \tilde{g}_j^k(\boldsymbol{x})\,(j = 1, ..., m) \\ where \quad & \boldsymbol{Q}^k = [\boldsymbol{A}^k, \boldsymbol{B}^k] \subseteq \mathbb{R}^d, \\ & A_i^k \geq A_i,\ \boldsymbol{B}_i^k \leq \boldsymbol{B}_i\,(i = 1, ..., d) \end{aligned} \tag{2}$$

where $\tilde{f}^k(\boldsymbol{x})$ and $\tilde{g}_j^k(\boldsymbol{x})\,(j = 1, ..., m)$ are approximated objective and constrained functions respectively, $\boldsymbol{Q}^k$ is the subregion in k_{th} iteration that bounded by $\boldsymbol{A}^k$ and $\boldsymbol{B}^k$. The solution of an individual sub-problem becomes the centering point of the trust region in the next iteration and the trust region is resized based on several indicators. This procedure is repeated until certain termination criteria are satisfied. In each iteration, a large portion of the fitting points are generated in the

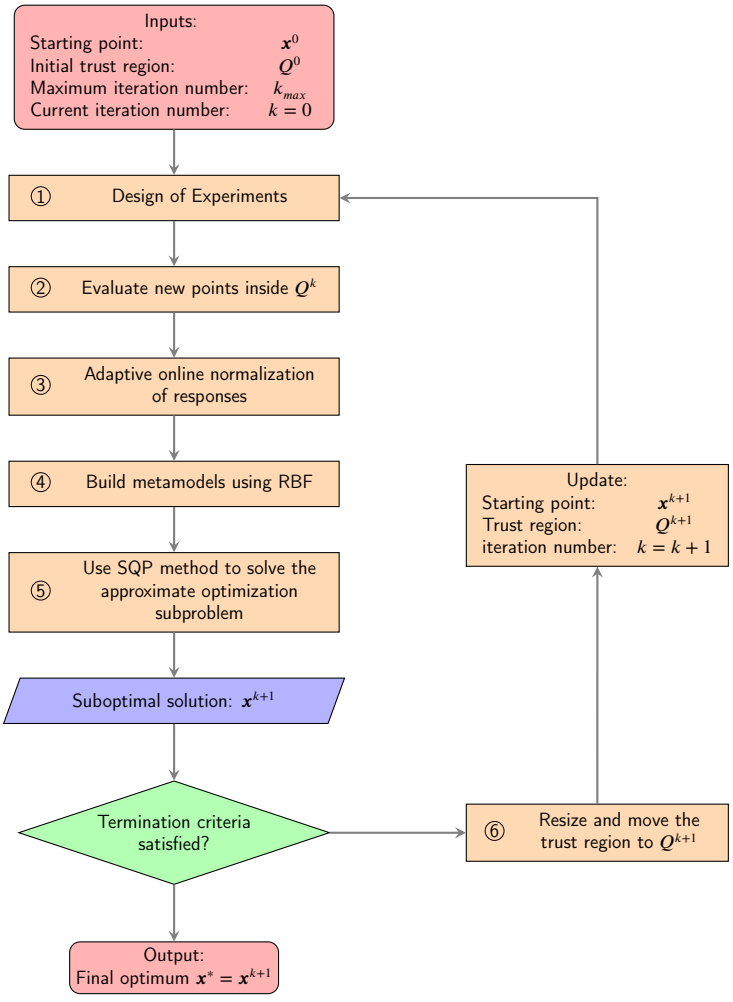

Figure 2: Flow chart of TARBF Method

current trust region, and the rest is taken from the pool of points in the previous iterations.

Figure 2 shows the optimization process of TARBF step by step and Figure 3 displays graphically how the trust regions are progressively resized and moved towards the optimum in a 2D optimization problem.

3.2. Design of experiments (DOE) strategies

As discussed by Kitayama [40], the key for obtaining the high-quality approximate global minimum is simultaneously adding new sampling points around (1) the optimum in the subregion and (2) the sparse region in the design space. TARBF achieves the two requirements with the novel design of experiments (DOE) and the moving trust region strategy. As metamodels are constructed within sequential trust regions centered at the successive sub-optimums, the first requirement above

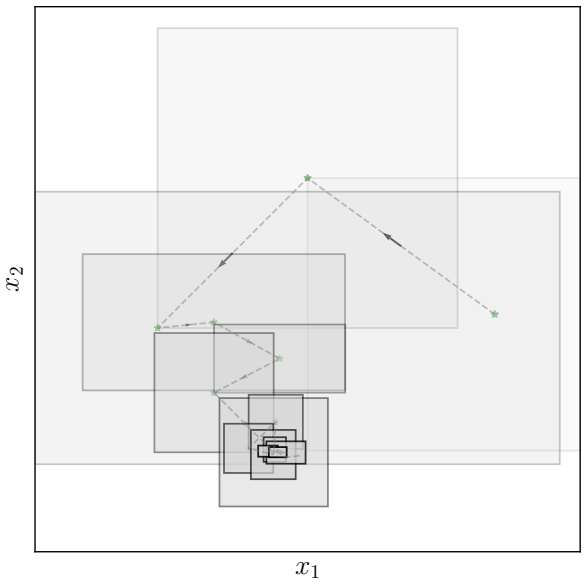

Figure 3: A typical optimization process of TARBF

is automatically achieved in TARBF. More specifically, the design of experiments (DOE) applied in TARBF to build the metamodel comprises three parts of the sampling points as shown in Figure 4. The first part of points (rectangle points in Figure 4) are randomly generated inside the trust region by the maxmin stochastic sampling strategy (MSS) [41, 42]. An additional constraint on the minimal distance between the points is imposed to improve the uniformity of the random plan. Besides, an extended-box selection strategy (EBS) is applied to select previous points located in the neighborhood of the current starting point. The size of the extended box is just 1.4 times larger than that of the current trust region and these points are considered alternatives to new points required in the current iteration. But in order to avoid the abuse of previous information which might cause the deterioration of the quality of the metamodel, only half of the number of points selected by EBS is deemed to be qualified for the alternatives. Last but not the least, for the purpose of making full use of previous information, additional points (triangle points in Figure 4) which are close to the current starting point and positioned out of the extended box are selected into the fitting pool by the global intelligence selection strategy (GIS). Summarily, MSS is responsible for uniformly scattering new points inside the trust region to improve the accuracy of the approximation. EBS is the key to maximally reducing the required number of new sampling points in each iteration without damaging the quality of the metamodel. And GIS reduces the risk

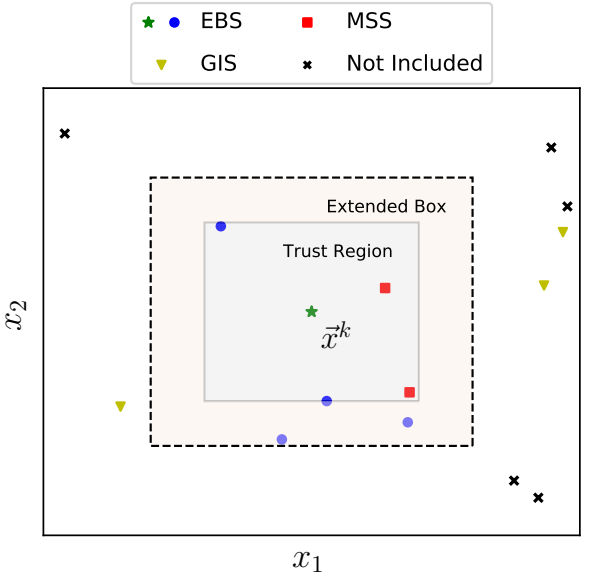

Figure 4: Design of experiments in TARBF

of the optimization process being trapped into local optima as a refined metamodel is built by adding points located outside of the trust region. As a result, the DOE in TARBF takes full advantage of the previous experimental data and contributes to high-accuracy local approximations. But in general, it is the trust region strategy which will be discussed in Section 3.5 that satisfies the second requirement to find the global optimum by attaining the balance between exploration and exploitation of the search space.

3.3. Adaptive normalization of the objective and constraint functions

Relatively few surrogate-based algorithms manage to scale the constraint violations to the same order of magnitude although it is a common sense that scaled constraint and objective functions can smooth the landscape of the metamodel surface and then affect the efficiency and accuracy of the optimization progress. One customary approach is to carry out a pre-process before building the metamodel. This is to say, identify the min-max range of each constraint over the entire design space and then divide each constraint by the determined values [34]. Although this technique does flatten the function by avoiding the extreme value globally, it makes the optimizer even harder to escape from the flat surface because this transformation makes the function where is already somewhat flat surface much flatter. This kind of issue also occurs in the logarithmic transformation proposed by Regis [33].

Based on the aforementioned limitations, a general, easy-to-implement and self-adaptive normalization strategy (SANS) is developed, as shown in Algorithm 1. In the first step of SANS, the maximum of actual responses (both the objective and constraint values) of the fitting points should be determined by

$$\begin{aligned} g_j^{max} &= max(|\boldsymbol{g}_j(\boldsymbol{X}^k)|) \\ f^{max} &= max(|\boldsymbol{f}(\boldsymbol{X}^k)|) \end{aligned} \tag{3}$$

where $\boldsymbol{X}^k$ is the set of fitting points which are used for metamodel building in the k_{th} iteration, $f(\boldsymbol{x})$ is the objective function and $g_j(\boldsymbol{x})$ $(j = 1, \ldots, m)$ are the j_{th} constraint functions. Then, the objective and constraint functions could be normalized respectively according to

$$\begin{aligned} g_j'(\boldsymbol{x}) &= \frac{g_j(\boldsymbol{x})}{g_j^{max}} \cdot \delta_g \in [-\delta_g, \delta_g], \quad if\ g_j^{max} > \delta_g \\ f'(\boldsymbol{x}) &= \frac{f(\boldsymbol{x})}{f^{max}} \cdot \delta_f \in [-\delta_f, \delta_f], \quad if\ f^{max} > \delta_f \end{aligned} \tag{4}$$

where $g_j'(\boldsymbol{x})$ and $f'(\boldsymbol{x})$ are the normalized constraint and objective functions; δ_g and δ_f are two user-specified parameters. Note that usually the inappropriate normalization of objective function (δ_f takes a relatively small value and smaller than δ_g) would obscure the improvement over consecutive iterations, thus, δ_f should be much larger than δ_g. Based on our numerous test results, $\delta_g = 1.0$ and $\delta_f = 10.0$ would be good choices for general optimization tasks.

In this way, the constraint and objective functions can be normalized adaptively during the entire optimization process. When the trust region $\boldsymbol{Q}^k$ is large, the responses of the points located in $\boldsymbol{Q}^k$ usually differ much from each other and then they will be scaled to the same level accordingly. In this situation, SANS is similar with the usual approach. However, when the trust region $\boldsymbol{Q}^k$ is sufficiently small, normally there will be few extreme values in objective and constraint functions in such small design space. In this situation, the conditions $g_j^{max} > \delta_g$ and $f^{max} > \delta_f$ are usually not satisfied and then there is no need to scale the functions. On the other hand, if the landscape of the function in such small region is still not flat, the SANS will be activated as normal. But since the maximum response values g_j^{max} and f^{max} generally should be smaller than the values in the initial stage, only a minor scale is sufficient enough to be applied to the original functions in support of accurate metamodels. Compared with the customary normalization strategy, SANS avoids the risk of normalizing the functions too much and improves the quality of approximations.

Algorithm 1: Self-adaptive normalization procedure (SANS)

Function SANS$(\delta_f, \delta_g, \boldsymbol{f}(X^k), \boldsymbol{g}_j(X^k)\,(j = 1, \ldots, m))$**:**

Input:

- δ_f: Normalizing the objective function value into the range $[-\delta_f, \delta_f]$.
- δ_g: Normalizing the constraint function value into the range $[-\delta_g, \delta_g]$.
- $\boldsymbol{f}(X^k)$: The objective function values of the fitting points.
- $\boldsymbol{g}_j(X^k)$: The j_{th} constraint function values of the fitting points.

Output:

- $f'(\boldsymbol{x})$: The normalized objective function
- $\boldsymbol{g}_j(X^k)\,(j = 1, \ldots, m)$: The normalized constraint functions

$f^{max}, g_j^{max} = Max(|\boldsymbol{f}(X^k)|, |\boldsymbol{g}_j(X^k)|)$

if $f^{max} > \delta_f$ **then** ▷ Objective normalization

  $f'(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{f^{max}} \cdot \delta_f$

else

  $f'(\boldsymbol{x}) = f(\boldsymbol{x})$

if $\boldsymbol{g}_j^{max} > \delta_g$ **then** ▷ Constraint normalization

  $g_j'(\boldsymbol{x}) = \frac{g_j(\boldsymbol{x})}{g_j^{max}} \cdot \delta_g$

else

  $g_j'(\boldsymbol{x}) = g(\boldsymbol{x})$

Return $f'(\boldsymbol{x}), \boldsymbol{g}_j(X^k)\,(j = 1, \ldots, m)$

3.4. Metamodel building using radial basis functions

Generally, any type of metamodel can be implemented in this trust region based framework. But by taking into account the computational time, the model accuracy and the model complexity, we use the simpler radial basis function surrogate in this paper. Given n distinct points $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \mathbb{R}^{\mathbb{d}}$ and their corresponding real function values $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n)$, where $f(\boldsymbol{x})$ could be either the objective function or the constraint function. The metamodel built by radial basis function interpolation can be expressed as

$$s(\boldsymbol{x}) = \sum_{i=1}^{n} \lambda_i \phi\left(\| \boldsymbol{x} - \boldsymbol{x}_i \|\right) + p(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^{\mathbb{d}} \tag{5}$$

where $\| \boldsymbol{x} - \boldsymbol{x}_i \|$ is the Euclidean distance between the studied point $\boldsymbol{x}$ and the fitting point $\boldsymbol{x}_i$. The coefficient λ_i is the unknown parameter that needed to be determined during the process of metamodel building. $P(\boldsymbol{x})$ is a linear polynomial

in d variables with $d+1$ coefficients as in the form:

$$p(x) = c_0 + c_1 \cdot x_1 + c_2 \cdot x_2 + ... + c_d \cdot x_d = \boldsymbol{c}^T \cdot \boldsymbol{x} \tag{6}$$

where ϕ is the radial basis function which can be of any forms but are always radially symmetric. Common types include the cubic ($\phi(r) = r^3$), multiquadric ($\phi(r) = -\sqrt{r^2+\gamma^2}$), Gaussian ($\phi(r) = \exp(-\gamma r^2)$) and thin plate spline ($\phi(r) = r^2 \log r$) forms. Here, γ is a shape parameter related to the smoothness of the function [43]. In this paper, the cubic form of RBF is adopted because it was successfully employed in several surrogate-based algorithms [44, 34, 16] and Wild et al. [45] suggested that the cubic variant is better than other choices especially under limited budget in terms of accuracy. Although the model accuracy can be improved by tuning the shape parameter γ when Gaussian or thin plate spline form is used, the additional computational burden may be unbearable especially for large-scale optimization problems.

To determine the parameters λ_i and $\boldsymbol{c}$, it results in the following linear system of equations:

$$\begin{bmatrix} \boldsymbol{\Phi}_{n\times n} & \boldsymbol{P}_{n\times(d+1)} \\ \boldsymbol{P}^T & \boldsymbol{0}_{(d+1)\times(d+1)} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\lambda}_{(n)} \\ \boldsymbol{c}_{(d+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{(n)} \\ \boldsymbol{0}_{(d+1)} \end{bmatrix} \tag{7}$$

where $\boldsymbol{\Phi}$ is an $n \times n$ square matrix containing evaluations of the RBF for the distances between all the sampling points, i.e., $\boldsymbol{\Phi}_{ij} = \phi\left(\| x_i - x_j \|\right) (i, j = 1, ..., n)$. $\boldsymbol{P} \in \mathbb{R}^{n\times(d+1)}$ is a matrix, of which the i-th row is $[1, \boldsymbol{x}_i^T]$. $\boldsymbol{0}_{(d+1)\times(d+1)} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a zero matrix, $\boldsymbol{\lambda}_{(n)} = [\lambda_1, ..., \lambda_n]^T \in \mathbb{R}^n$, $\boldsymbol{c}_{(d+1)} = [c_0, c_1, ..., c_d]^T \in \mathbb{R}^{d+1}$, $\boldsymbol{F}_{(n)} = [f(x_1, f(x_2), ..., f(x_n)]^T \in \mathbb{R}^n$ and $\boldsymbol{0}_{(d+1)} \in \mathbb{R}^{d+1}$ is a vector of zeros. The coefficient matrix in Equation 7 is invertible if it has full rank. In other words, there exits a subset of $d+1$ linearly independent points among the fitting sets.

3.5. Modified trust region strategy

Once the RBF surrogates for the objective and constraint functions have been identified, the optimization subproblem (Equation 2) can be solved using any mathematical or metaheuristic solvers. Here, the sequential quadratic programming (SQP) solver in Scipy library [46] is employed as an optimizer to find the optimal solution mathematically. Then, the new search subregion in the next iteration, which is centered at this optimal solution should be determined, i.e., the move limits A_i^k and B_i^k should be updated. Inspired by the move limit strategy described in [47], several indicators as shown in Table 1 have been formulated in support of moving and resizing the trust region.

Table 1
Indicators for trust region strategy

1st indicator	The size of the current trust region Q^k		
	Small	Large	
2nd indicator	The location of $\boldsymbol{x}^{k+1}$ in local bounds $[\boldsymbol{A}^k, \boldsymbol{B}^k]$ and global bounds $[A_i, B_i]$		
	internal	external	boundary
3rd indicator	The movement history of the each design variable x_i^{k+1} $(i = 1, \ldots, d)$		
	forward	backward	
4th indicator	The movement history of $\boldsymbol{x}^{k+1}$ in the design space		
	Forward	Backward	Uncertain

The first indicator is the size of the current search subregion Q^k. If $Q^k \leq \Delta_{min}$, the size of Q^k will be marked as 'Small', or else it will be denoted as 'Large'.

The second indicator is used to identify the location of the each design variable x_i^{k+1} $(i = 1, \ldots, d)$ of the sub-optimal solution $\boldsymbol{x}^{k+1}$ in current bounds $[A_i^k, B_i^k]$ and in global bounds $[A_i, B_i]$. If none of the current move limits ($[A_i^k, B_i^k]$) is active (for example, when one design variable reaches either the upper bound or the lower bound, this move limit is activated), this variable x_i^{k+1} $(i = 1, \ldots, d)$ is considered an 'internal' variable. If any of the current move limits ($[A_i^k, B_i^k]$) is active but none of the global bounds $[A_i, B_i]$ is active, the location of this variable is marked as 'external'. Otherwise the location is denoted as 'boundary'.

The third indicator illustrates the movement history of the each design variable x_i^{k+1} $(i = 1, \ldots, d)$. For simplicity, a measure θ_i^k $(i = 1, \ldots, d)$ is defined as

$$\theta_i^k = (x_i^{k+1} - x_i^k) \cdot (x_i^k - x_i^{k-1}) \ (i = 1, \ldots, d) \tag{8}$$

If $\theta_i^k > 0$, the movement of this variable in this dimension is considered 'forward'. Otherwise, this variable is moving 'backward'.

The forth indicator aims to show the movement history of the sub-optimal solution $\boldsymbol{x}^{k+1}$ in the design space. Therefore, the angle between the last two move vectors is defined as

$$\varphi^k = \frac{\boldsymbol{x}^{k+1} - \boldsymbol{x}^k}{|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k|} \cdot \frac{\boldsymbol{x}^k - \boldsymbol{x}^{k-1}}{|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}|} \tag{9}$$

If $\varphi^k > 0.5$, the optimization process is moving more or less in the same direction, so the sub-optimal solution is moving 'Forward'. If $\varphi^k <= 0$, the convergence

history is marked as 'Backward'. Otherwise, the next search subregion will not be dependent on this indicator because the movement of optimal solutions is 'Uncertain'.

Based on the above indicators, the next search subregion Q^{k+1} can be determined by new move limits A_i^{k+1} and B_i^{k+1} as

$$B_i^{k+1} - A_i^{k+1} = \tau_i^k \cdot (B_i^k - A_i^k) \tag{10}$$

where τ_i^k is the resizing coefficient. In the current implementation, $\tau_i^k = 1.5$ is used for enlargement and $\tau_i^k = 1/1.5$ is used for reduction. To ensure a robust exploration of the design space, in the first k_{RES} (the default value is 5) iterations, the subspace of each design variable will only be shrunk when this variable is located at either the global lower bound or the global upper bound, i.e., this variable is considered 'boundary'. Otherwise, the size of the subspace will remain unchanged to keep the global searching ability. And in order to achieve a moderate exploitation of the promising region where the global optimum might be located, the subspace of each design variable will only be shrunk when this variable is not located on the global bounds and the movement history is labelled as 'Backward'. A schematic description of the trust region strategy is given in Figure 5. The symbols || (logical OR) and & (logical AND) are logical operators. For example, only if the i_{th} variable is 'external' and 'forward', the subspace will be enlarged accordingly.

3.6. Early termination strategy (ETS)

The early termination strategy (ETS) is developed to abort the optimization process appropriately if a really good solution has been found and there seems no big margin for improvement on this solution. In the current implementation, ETS will be activated if three conditions are all satisfied. First, the obtained sub-optimal solution $\boldsymbol{x}^{k+1}$ in k_{th} iteration should be feasible because an infeasible solution is definitely not the global optimum and indicates that the optimization process should continue to run. Second, the variation of the objective function values of the last two sub-optimal solutions defined by $I = f(\boldsymbol{x}^{k+1}) - f(\boldsymbol{x}^k)$ should be positive and less than a small value I_{max} (for example, $1e-8$). If this condition is satisfied, the optimization process possibly converges to a solution so there will be little room for reduction on the objective function value. Third, in order to avoid premature convergence, the relative size of the current search subregion S^k should be smaller than a proposed value of $\Delta_{min,2}$. From numerical tests, $\Delta_{min,2} = 0.01$ is a suitable value for solving general optimization tasks.

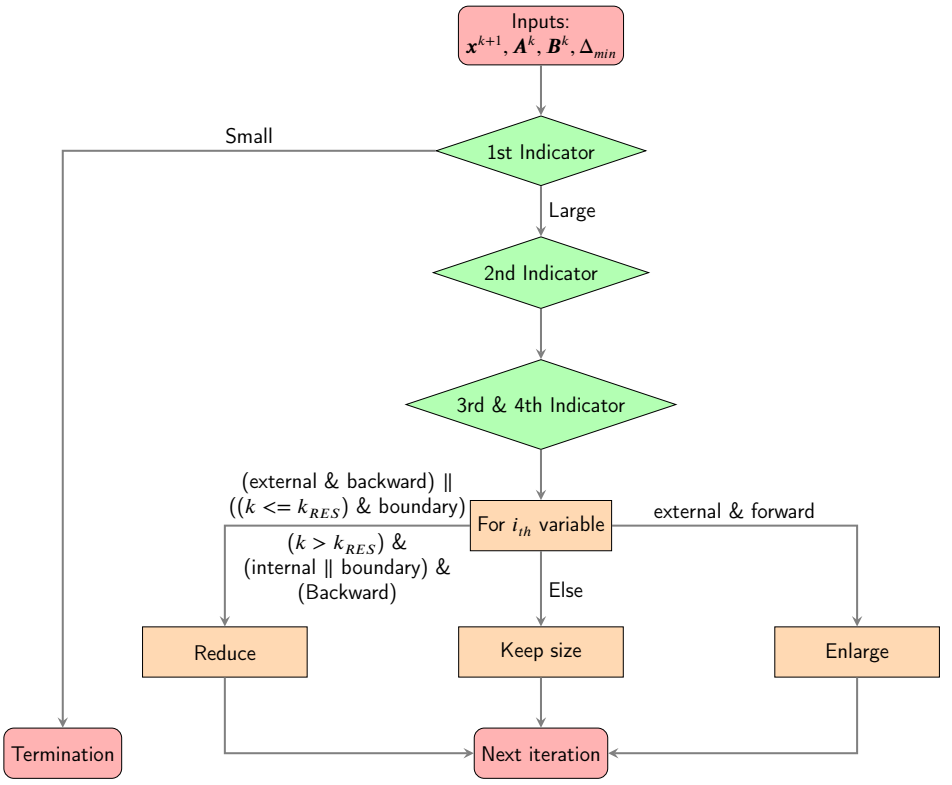

Figure 5: Schematic description of the trust region strategy

4. Experimental results

In this section, computational experiments are executed for the purpose of evaluating how well TARBF performs on seeking the global optima of the black-box constrained optimization problems.

4.1. Benchmark functions

In this work, 21 well-known G-problems of CEC'2006 test suite [37] are used for performance evaluation. Here we do not consider the problems G02, G20 and G22 which are hardly solved by any metamodel-based algorithms. Table 2 shows the main properties of the benchmark functions and please refer to [37] for more details about the functions. Note that in this study, all equality constraints $h(\boldsymbol{x}) = 0$, are converted into inequality constraints of the form $|h(\boldsymbol{x})| - \epsilon \leq 0$ and $\epsilon = 1e-6$ is used for both equality constraints and inequality constraints, which is more stringent than the proposed value $\epsilon = 1e-4$ in CEC'2006. Hence, we use the results given in [48] and [49] as known optima for problems including equality constraints (G03, G05, G11, G13, G14, G15, G17, G21, G23).

Table 2
Summary description of G-problems

Prob.*	Optimum	d[a]	m[b]	Type[c]	$\rho(\%)$[d]
G01	-15.0000	13	9	L+Q	0.0111
G03	-1.0050	10	2	N	0.0000
G04	-30665.5387	5	6	Q+N	52.1230
G05	5126.4967	4	8	L+C+N	0.0000
G06	-6961.8139	2	2	C+N	0.0066
G07	24.3062	10	8	L+Q+N	0.0003
G08	-0.0958	2	2	N	0.8560
G09	680.6300	7	4	N	0.5121
G10	7049.2480	8	6	L+N	0.0010
G11	0.7500	2	2	Q+N	0.0000
G12	-1.0000	3	1	Q+N	4.7713
G13	0.0540	5	6	N	0.0000
G14	-47.7611	10	6	L+N	0.0000
G15	961.7152	3	4	Q+N	0.0204
G16	-1.9052	5	38	L+N	0.0204
G17	8876.9807	6	8	N	0.0000
G18	-0.8660	9	13	Q+N	0.0000
G19	32.6556	15	5	N	33.4761
G21	193.7869	7	11	L+N	0.0000
G23	-400.0000	9	10	L+N	0.0000
G24	-5.5080	2	2	L+N	79.6556

* The problem in bold face means that the optimum satisfies a more stringent constraint tolerance ($1e-6$) than the proposed value ($1e-4$) in [37].
[a] The number of design variables.
[b] The number of inequality constraints.
[c] The problem contains linear (L), quadratic(Q), cubic (Q) or nonlinear (N) functions.
[d] The ratio between the feasible region and the entire search space.

4.2. Parameter settings

The parameter settings in TARBF are summarized in Table 3. Note that although there is a number of ad hoc parameters in TARBF, they have been properly adjusted and determined through extensive experiments and sensitivity analyses. The default values listed in Table 3 are configured for tackling general optimization tasks and do not need further adjustment. In this paper,TARBF is written in Python and is executed on a desktop machine with an AMD Ryzen CPU 1800X.

Table 3
User parameters of TARBF with their descriptions and default values

Parameter	Description	Value
$\boldsymbol{x}^0$	The initial trial solution	Random
K_{max}	The maximum number of iterations	100
N_{plan}	The number of required sampling points	$d+5$
Δ^0	The relative size of the initial trust region	1.0
Δ_{min}	The minimum relative size of the trust region	$1e-5$
Δext	The relative size of the extended box	1.4
TOL_{con}	The constraint tolerance	$1e-6$
δ_g, δ_f	The normalization coefficients	1.0, 10.0
I_{max}	The tolerated variation in objective function values in ETS	$1e-8$
$\Delta_{min,2}$	The tolerated relative size of the trust region in ETS	0.01

For each problem, TARBF is executed 25 times independently.

4.3. Performance criteria

To illustrate the convergence performance of TARBF, the following performance criteria are used [37].

- $\boldsymbol{x}_{opt}, f(\boldsymbol{x}_{opt})$: $\boldsymbol{x}_{opt}$ is the best point found by TARBF and $f(\boldsymbol{x}_{opt})$ is the corresponding objective value.
- Best, Worst, Average, Median: The best, worst, average and median value from all trials.
- Feasible run and feasible rate (FR): If $\boldsymbol{x}_{opt}$ is feasible, this run is a feasible run and the feasible rate is equal to the number of feasible runs over total runs.
- Successful run and success rate (SR): Let $\boldsymbol{x}^*$ be the known optimum of the problem, a successful run is a run which finds a feasible solution satisfying $f(\boldsymbol{x}_{opt}) - f(\boldsymbol{x}^*) \leq 1e-4$. The success rate is the number of successful runs over total runs.

- NFEs, NTEs, ANFEs, ATEs, ENFEs, EATEs, : NFEs is used to record the number of function evaluations in each run when TARBF terminates and at least one feasible solution is found, while NTEs records the number of function evaluations when TARBF first finds the successful solution satisfying $f(\boldsymbol{x}_{opt}) - f(\boldsymbol{x}^*) \leq 1e-4$. ANFEs or ATEs is the average of the values of NFEs or NTEs in all feasible runs. In addition, both ENFEs and EATEs are defined as ENFEs or EATEs $= {}^{\text{ANFEs or ATEs}}/_{SR \cdot FR}$. Generally, ENFEs reflects the maximum function evaluations required by TARBF to find the optimal solution with high robustness while EATEs shows the minimum function evaluations used. The difference between ENFEs and EATEs actually represents the convergence capability of TARBF. If this value is small, it means TARBF converges to the global optimum quickly, and vice versa.

4.4. General performance of TARBF

Table 7 shows the optimal objective values obtained by TARBF and Table 4 presents the convergence statistics of TARBF on solving these 21 G-problems. As can be seen from the FR values in Table 4, TARBF is able to find a feasible solution of the majority of the G-problems with a one hundred percent. G10, G15 and G21 are three exceptions but the FR values are both over 90%. And according to the SR values, 21 G-problems can be classified into five classes as shown in Table 5. 11 G-problems (G01, G03, G04, G05, G06, G07, G09, G11, G15, G16 and G23) can be robustly solved by TARBF with a success rate over 92%. And another 6 problems (G10, G14, G18, G19, G17 and G24) can still be addressed by TARBF with a high success rate over 52%. Among them, the SR values of the first four cases are over 72%. In addition, TARBF shows instability in solving four problems (G08, G13, G21 and G12), of which the SR values are below 52%. The most difficult problem to TARBF is G12, on which TARBF only has 16% chance of obtaining the global optimum. This is mainly because the feasible region of this problem consists of 9^3 disjointed spheres [37], which makes TARBF difficult to escape from local optima.

Averagely, TARBF has 76.76% chance of finding the global optimum of one of the 21 G-problems within 374.10 function evaluations. And in terms of the ENFEs value, TARBF is believed to robustly and definitely solve one G-problem within 535.69 function evaluations.

4.5. Comparisons between TARBF and other state-of-the-art metamodel-based algorithms

As shown in Table 6, TARBF is compared with various advanced metamodel-based algorithms in order to evaluate its performance. These approaches include COBRA [33], eDIRECT-C [50], SADE-kNN [51] and SACOBRA [34]. COBRA

Table 4
Convergence statistics on G-suite problems.

Prob.	ATEs	ANFEs	FR(%)	SR(%)	EATEs	ENFEs
G01	82	290	100	92	89	315
G02	2126	2126	100	0	-	-
G03	616	1046	100	100	616	1046
G04	37	148	100	100	37	148
G05	36	55	100	96	38	57
G06	14	40	100	100	14	40
G07	396	485	100	100	396	485
G08	71	80	100	28	253	285
G09	521	557	100	92	566	605
G10	494	532	96	72	715	769
G11	27	76	100	100	27	76
G12	78	86	100	16	489	537
G13	324	360	100	28	1158	1287
G14	571	758	100	84	680	902
G15	74	101	92	92	87	120
G16	111	152	100	100	111	152
G17	366	555	100	60	610	924
G18	438	503	100	72	608	698
G19	1295	1449	100	88	1472	1646
G21	327	327	96	40	852	851
G23	106	207	100	100	106	207
G24	37	51	100	52	72	99
Average	286.75	374.10	99.24	76.76	428.34	535.69

Table 5
Classification of G-suite problems by success rate (SR)

Class	Problem(s)	Quantity
$SR \geq 92\%$	G01, G03, G04, G05, G06, G07, G09, G11, G15, G16, G23	11
$72\% \leq SR < 92\%$	G10, G14, G18, G19	4
$52\% < SR < 72\%$	G17, G24	2
$20\% < SR < 52\%$	G08, G13, G21	3
$SR \leq 20\%$	G12	1

[33] makes use of the radial basis function (RBF) to build surrogate models of both the objective and constraint functions and works well in solving the well-known large scale optimization problem MOPTA08 [52]. eDIRECT-C [50] adaptively select metamodel type (Kriging, RBF and polynomial) in the optimization process and then a pure greedy search is conducted to seek the solution. SADE-kNN [51] applied the k-nearest-neighbors (kNN) technique to make predictions of unknown points in a differential evolution framework. In kNN, the approximate response of a unknown solution is a weighted average of the responses of the k nearest solutions from a database. SACOBRA [34] is based on COBRA [33] but capable of efficiently solving constrained problems with no parameter tuning. As the detailed optimization results of COBRA were not reported in [33], here we would use the

Table 6
Comparisons between TARBF and other state-of-the-art metamodel-based algorithms on 21 G-problems

Prob.	Criteria	TARBF	COBRA [50]	eDIRECT-C [50]	SADE–kNN [51]	SACOBRA [34]
G01	Best	-15.0000	-14.9994	-15.0000	-15.0000	-15.0
	ENFEs	315	> 908.4	147	> 3722	100
G03	Best	-1.0005	-0.9397	-1.0005	-0.4515	-1.0
	ENFEs	1046	> 1000	145	N.A.	300
G04	Best	-30665.5394	-30664.0217	-30665.5387	-30665.5386	-30665.539
	ENFEs	148	72.0	65	2598	200
G05	Best	5126.4981	5202.9265	5134.9124	5126.49	5126.498
	ENFEs	57	> 1000	413	> 17810	200
G06	Best	-6961.8139	-6950.5314	-6961.8139	-6961.8138	-6961.81
	ENFEs	40	> 1000	35	1235	100
G07	Best	24.3062	24.3220	24.3062	24.3073	24.306
	ENFEs	485	> 1000	152	N.A.	200
G08	Best	-0.0958	-0.095822	-0.095825	-0.09582	-0.0958
	ENFEs	285	> 980.6	154	292	200
G09	Best	680.6301	696.8378	691.6906	680.638	680.761
	ENFEs	605	> 1000	> 1000	N.A.	300
G10	Best	7049.2480	7370.8840	7049.2480	7049.249	7049.253
	ENFEs	769	> 1000	105	N.A.	300
G11	Best	0.7500	0.7500	0.7499	0.7499	0.75
	ENFEs	76	> 454.8	33	> 2995	100
G12	Best	-1.0000	-1.0000	-1.0000	-1.0000	N.A.
	ENFEs	537	32.0	52	> 386	N.A.
G13	Best	0.0539	0.5018	0.2225	0.05394	N.A.
	ENFEs	1287	> 1000	> 1000	> 43907	N.A.
G14	Best	-47.7611	N.A.	N.A.	-47.764	N.A.
	ENFEs	902	N.A.	N.A.	> 55179	N.A.
G15	Best	961.7152	N.A.	N.A.	961.7150	N.A.
	ENFEs	120	N.A.	N.A.	> 11431	N.A.
G16	Best	-1.9052	N.A.	N.A.	-1.9051	N.A.
	ENFEs	152	N.A.	N.A.	4633	N.A.
G17	Best	8853.5401	N.A.	N.A.	8853.53	N.A.
	ENFEs	698	N.A.	N.A.	> 69887	N.A.
G18	Best	-0.8660	N.A.	N.A.	-0.8654	N.A.
	ENFEs	698	N.A.	N.A.	> 253743	N.A.
G19	Best	32.6556	N.A.	N.A.	32.6632	N.A.
	ENFEs	1646	N.A.	N.A.	N.A.	N.A.
G21	Best	193.7869	N.A.	N.A.	193.7546	N.A.
	ENFEs	851	N.A.	N.A.	N.A.	N.A.
G23	Best	-400.0000	N.A.	N.A.	-400.055	N.A.
	ENFEs	207	N.A.	N.A.	> 68852	N.A.
G24	Best	-5.5080	N.A.	N.A.	-5.5080	N.A.
	ENFEs	99	N.A.	N.A.	765	N.A.

* N.A. means the result is unavailable in the paper.
! The termination criteria used in algorithms are different from each other.

results obtained by Liu et al. in [50]. It is worthwhile noting that this comparison might be unfair for TARBF as other algorithms use different termination criteria. As described in Section 3.6, TARBF will terminate when it converges to a good solution. But for COBRA and eDIRECT-C, Liu et al. applied the relative error ($E = \frac{f(\boldsymbol{x}_{opt})-f(\boldsymbol{x}^*)}{f(\boldsymbol{x}^*)}$) as the stopping criteria. And in SACOBRA [34], a problem

Table 7
Statistical results of the objective values obtained by TARBF

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-14.6769	-14.9794	-15.0000	7.2171e-02
G03	-1.0005	-1.0005	-1.0005	-1.0005	-1.0005	1.8212e-06
G04	-30665.5387	-30665.5393	-30665.5387	-30665.5387	-30665.5387	1.7696e-04
G05	5126.4981	5126.4981	5126.4988	5126.4981	5126.4981	1.2591e-04
G06	-6961.8139	-6961.8140	-6961.8139	-6961.8139	-6961.8139	2.5056e-05
G07	24.3062	24.3062	24.3062	24.3062	24.3062	9.0971e-07
G08	-0.0958	-0.0958	-0.0000	-0.0486	-0.0291	3.4403e-02
G09	680.6301	680.6301	680.6303	680.6301	680.6301	5.4075e-05
G10	7049.2480	7049.2480	7069.6325	7050.8757	7049.2480	5.2572e+00
G11	0.7500	0.7500	0.7500	0.7500	0.7500	2.5026e-07
G12	-1.0000	-1.0000	-0.8102	-0.9681	-0.9864	4.5660e-02
G13	0.0539	0.0539	1.0000	0.3760	0.4389	2.5044e-01
G14	-47.7611	-47.7611	-47.7602	-47.7610	-47.7611	2.3679e-04
G15	961.7152	961.7152	961.7152	961.7152	961.7152	4.3152e-06
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	9.2121e-09
G17	8876.9807	8853.5419	8929.6637	8889.4008	8871.1676	3.1802e+01
G18	-0.8660	-0.8660	-0.5000	-0.7915	-0.8660	1.2794e-01
G19	32.6556	32.6556	32.6942	32.6572	32.6556	7.5597e-03
G21	193.7869	193.7859	196.2486	193.9741	193.7906	5.2647e-01
G23	-400.0000	-400.0001	-400.0000	-400.0000	-400.0000	1.9922e-05
G24	-5.5080	-5.5080	-4.0537	-4.9497	-5.5080	6.3702e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	2.0543e-11
SPD	0.0127	0.0127	0.0133	0.0127	0.0127	1.2386e-04
PVD	5885.3328	5885.3328	5885.3328	5885.3328	5885.3328	3.8984e-07
SRD	2994.4710	2994.4703	2994.4711	2994.4710	2994.4711	1.5679e-04

Table 8
Comparisons between TARBF and other optimization algorithms on engineering problems

Prob.	Criteria	TARBF	KCGO [38]	eDIRECT-C [50]	COBRA [50]	iDEaSm [53]	SCGOSR[35]
WBD	f_{mean}	1.7249	2.3230	N.A.	N.A.	1.7249	> 1.7249
	ATEs	137	115	N.A.	N.A.	4425	101.9
SPD	f_{mean}	0.0127	0.0135	0.0127	0.0131	N.A.	0.0127
	ATEs	211	38	320	> 1000	N.A.	75.7
PVD	f_{mean}	5885.3328	N.A.	7006.8195	7147.0934	5887.1084	> 5885.3653
	ATEs	108	N.A.	412	> 1000	20000	42.9
SRD	f_{mean}	2994.4710	2999.76	2994.4857	2994.7114	N.A.	> 2994.5
	ATEs	78	43	140.2	330.6	N.A.	88.1

is considered to be 'solved' if $f(\boldsymbol{x}_{opt}) - f(\boldsymbol{x}^*) \leq 0.05$ and the SR values are not reported. And for SADE-kNN, the optimization process would stop if a solution satisfying $f(\boldsymbol{x}_{opt}) - f(\boldsymbol{x}^*) \leq 1e-4$. Overall, except TARBF, the results of other algorithms do not show the true convergence statistics reflecting the search ability in solving these problems as the global optima are used as known conditions, which is impractical in solving real-world optimization problems. Generally, if the global optimum of a problem is used as a termination criterion, the required number of function evaluations when this criterion is satisfied is less than the true number of function evaluations when the algorithm actually converges.

Nevertheless, from the comparison results, it can be concluded that TARBF is superior to other algorithms in terms of accuracy and/or efficiency and/or robustness. More specifically, as compared to SADE-kNN which tests all these 21 problems, TARBF requires only a small fraction (about $1/10$ to $1/1000$) of the number of function evaluations. Besides, SADE-kNN fails to solve 5 problems (G03, G07, G10, G14 and G19). And among the rest three algorithms, COBRA seems to have severe difficulty in finding the global optimum as only 3 (G08, G11 and G12) out of the 12 problems are successfully solved. Moreover, eDIRECT-C quickly finds the optimal solutions of the majority of the twelve problems but it fails to solve three problems including G05, G09 and G13. For the other nine problems, although the number of function evaluations reported by eDIRECT-C is less than the result of TARBF, it does not mean that eDIRECT-C is much efficient than TARBF. The main issue lies in how the eDIRECT-C defines a real convergence state. Similarly, SACOBRA terminates before the real convergence as well. It shows good search ability in solving 8 out of the 10 problems (G01 to G11) but does not work well in optimizing G09 and G10 while TARBF is able to tackle these two problems with high accuracy and robustness. Hence, TARBF can be considered the best algorithm among these well-known algorithms.

4.6. Engineering examples

To comprehensively validate the correctness of TARBF, four widely-used real-world engineering problems are tested as well, including the welded beam design (WBD), the tension/compression design (SPD), the pressure vessel design (PVD) and the speed reducer design (SRD). The mathematical formulae of these problems are provided in Appendix A. Table 9 gives the main characteristics of the four engineering problems. The parameter settings and the stopping criteria here are the same as that for solving G-problems, as summarized in Table 3. For the purpose of simplicity, similar to the literatures ([54, 50]), the design variables of integer values are regarded as continuous ones.

Table 7 gives the best-median-worst results of TARBF and Table 10 provides the main convergence statistics (the performance criteria are the same as described in Section 4.3). As Compared with other algorithms including KCGO [38], eDIRECT-C [50], COBRA [50], iDEaSm [53] and SCGOSR[35] in Table 8, the proposed TARBF achieves the highest solution accuracy with the least number of evaluations.

Table 9
Summary description of engineering problems

Prob.	Optimum	d[a]	m[b]	Type[c]	$\rho(\%)$[d]
WBD	1.7249	4	6	L+N	2.7020
TSD	0.0127	3	4	L+N	0.7428
PVD	5885.3328	4	4	L+N	39.8007
SRD	2994.4711	7	11	N	0.0955

[a] The number of design variables.
[b] The number of inequality constraints.
[c] The problem contains linear (L), nonlinear (N) functions.
[d] The ratio between the feasible region and the entire search space.

Table 10
Convergence statistics of engineering problems.

Prob.	ATEs	ANFEs	FR(%)	SR(%)	EATEs	ENFEs
WBD	137	169	100	100	137	169
SPD	211	413	100	96	220	431
PVD	108	157	100	100	108	157
SRD	78	181	100	100	78	181
Average	133.52	230.13	100.00	99.00	135.72	234.44

5. Conclusion

In this paper, a novel trust-region based adaptive radial basis function (TARBF) algorithm is proposed for solving constrained black-box problems within very limited budget. This algorithm includes the economical design of experiments, the adaptive online normalization of the functions and the sophisticated trust region strategy that helps TARBF balance the exploration and exploitation of the search space for the optimal solution with a high level of accuracy and a limited number of function evaluations. Throughout 25 benchmark examples (21 G-problems and 4 engineering problems), TARBF proves to be a convergent and efficient paradigm to find the global optimum. Only about 500 function evaluations are required by TARBF in average to robustly obtain the known optimal solution. By comparison of the published results, TARBF demonstrates the promising performance and superiority over other state-of-the-art algorithms to deal with constrained problems in terms of accuracy, efficiency and robustness. Moreover, TARBF provides a valuable insight into the development of metamodel assisted robust optimization method for solving highly multi-modal and large-scale design problems with less

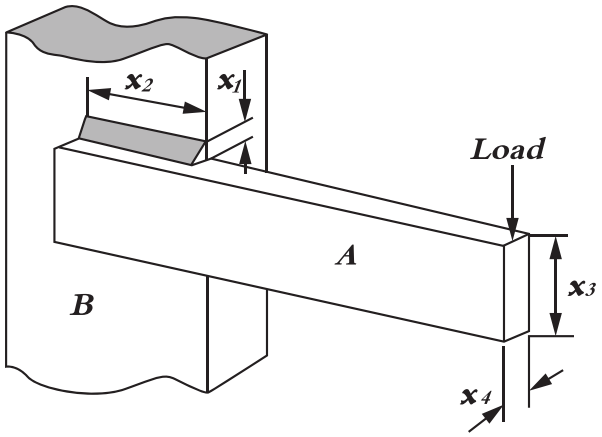

Figure 6: Schematic view of the welded beam structure

computational cost.

Compliance with ethical standards

Conflict of interests The authors declare that they have no conflict of interest.

Replication of results Main Python codes for the results presented in Tables 4 and 7 are available as supplementary material. In case of questions or difficulties, readers are welcome to contact the authors. Additionally, the authors are willing to share raw data of the resulting designs upon the request.

Appendix A

1. Welded Beam Design (WBD)

As shown in Fig. 6, the beam is welded to a rigid support and is designed for the minimum cost, considering constraints on shear stress (τ), bending stress (σ), buckling load $\left(p_c\right)$, and end deflection (δ). The design variables comprise the thickness of the weld $\left(x_1\right)$, the length of the welded joint $\left(x_2\right)$, the width of the beam $\left(x_3\right)$ and the thickness of the beam $\left(x_4\right)$. The problem can be formulated mathematically as Equation 11.

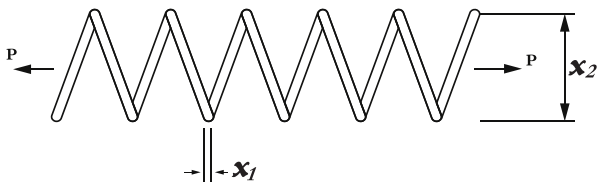

Figure 7: Schematic view of the spring structure

$$
\begin{aligned}
min \quad & f(\boldsymbol{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \\
s.t. \quad & g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{max} \le 0 \\
& g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{max} \le 0 \\
& g_3(\boldsymbol{x}) = x_1 - x_4 \le 0 \\
& g_4(\boldsymbol{x}) = [0.10471x_1^2 + 0.04811x_3x_4(14 + x_2)] - 5 \le 0 \\
& g_5(\boldsymbol{x}) = 0.125 - x_1 \le 0 \\
& g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{max} \le 0 \\
& g_7(\boldsymbol{x}) = p - p_c(\boldsymbol{x}) \le 0 \\
where \quad & P = 6000\,lb,\ L = 14\,in,\ E = 30 \times 10^6\,psi,\ G = 12 \times 10^6\,psi, \\
& \tau_{max} = 13600\,psi,\ \sigma_{max} = 30000\,psi,\ \delta_{max} = 0.25\,in \\
& \tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right) \\
& R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
& \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \\
& J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \\
& \sigma(x) = \frac{6PL}{x_4x_3^2}, \delta(x) = \frac{4PL^3}{Ex_3^3x_4} \\
& p_c(x) = \frac{4.013E\sqrt{\left(\frac{x_3^2x_4^6}{36}\right)}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \\
& 0.1 \le x_1 \le 2,\ 0.1 \le x_2 \le 10,\ 0.1 \le x_3 \le 10,\ 0.1 \le x_4 \le 2
\end{aligned}
\tag{11}
$$

2. Tension/compression spring design (SPD)

As shown in Figure 7, the design variables include the wire diameter $d\left(x_1\right)$, the mean coil diameter $D\left(x_2\right)$, and the number of active coils $N\left(x_3\right)$. The design objective is to minimize the weight of the spring subject to constraints on the minimum deflection g_1, shear stress g_2, surge frequency g_3 and the limits on the outside diameter g_4. The mathematical description of this problem is given as follows:

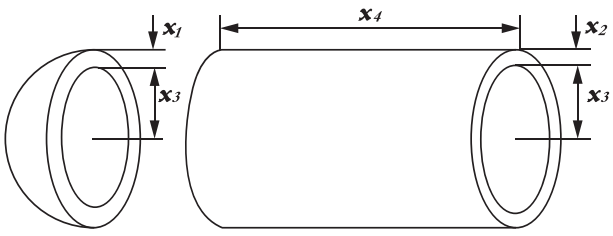

Figure 8: Schematic view of the pressure vessel

$$
\begin{aligned}
min \quad & f(\boldsymbol{x}) = x_1^2 x_2 \left(x_3 + 2\right) \\
s.t. \quad & g_1(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\
& g_2(x) = \frac{4x_2^2 - x_2 x_1}{12566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \\
& g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\
& g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \leq 0 \\
where \quad & 0.05 \leq x_1 \leq 1;\ 0.25 \leq x_2 \leq 1.3;\ 2 \leq x_3 \leq 15.
\end{aligned}
\tag{12}
$$

3. Pressure vessel design (PVD)

Figure 8 shows a cylindrical pressure vessel capped at both ends by hemispherical heads. According to the American society of mechanical engineers (ASME) boiler and pressure vessel code, this vessel is designed for a working pressure of 3000 psi and a minimum volume of 750 ft^3. The objective is to minimize the total cost which involves a welding cost, a material cost and a forming cost. The variables include the thickness of shell (x_1), the thickness of the head (x_2), the inner radius (x_3), and the length of the cylindrical section of the vessel (x_4). Among them, x_3 and x_4 are continuous variables but the thickness x_1 and x_2 can only take integer multiples of 0.0625 $inch$. The mathematical expression of this problem is given in Equation 8.

$$
\begin{aligned}
min \quad & f(\boldsymbol{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_1 x_3^2 + \\
& 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \\
s.t. \quad & g_1(x) - x_1 + 0.0193 x_3 \leq 0 \\
& g_2(x) = -x_2 + 0.00954 x_3 \leq 0 \\
& g_3(x) = -\pi x_3^2 x_4^2 - \frac{4}{3}\pi x_3^3 + 129600 \leq 0 \\
& g_4(x) = x_4 - 240 \leq 0 \\
where \quad & 1 \times 0.0625 \leq x_1, x_2 \leq 99 \times 0.0625 \\
& 10.0 \leq x_3, x_4 \leq 200.0
\end{aligned}
\tag{13}
$$

4. Speed reducer design (SRD)

A speed reducer as shown in Figure 9 is part of the gear box of mechanical system. The total weight of the speed reducer is to be minimized subject to the nine constraints which include the limits on the bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts. The design variables are the face width (x_1), the module of the teeth (x_2),

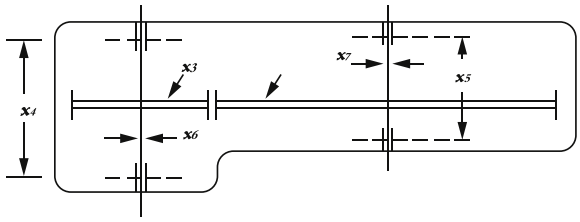

Figure 9: Schematic view of the speed reducer

the number of teeth on pinion (x_3), the length of the first shaft between bearings (x_4), the length of the second shaft between bearings (x_5), the diameter of the first and the second shaft (x_6 and x_7). Among them, x_3 is integer and the rest are continuous. The mathematical formulation can be summarized in Equation 14.

$$
\begin{aligned}
\textit{min}\quad & f(\boldsymbol{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
& - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\
& + 0.7854(x_4x_6^2 + x_5x_7^2) \\
\textit{s.t.}\quad & g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \le 0 \\
& g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0 \\
& g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0 \\
& g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0 \\
& g_5(x) = \frac{1.0}{110x_6^3}\sqrt{(\frac{745.0x_4}{x_2x_3})^2 + 16.9 \times 10^6} - 1 \le 0 \\
& g_6(x) = \frac{1.0}{110x_6^3}\sqrt{(\frac{745.0x_4}{x_2x_3})^2 + 16.9 \times 10^6} - 1 \le 0 \\
& g_7(x) = \frac{x_2x_3}{40} - 1 \le 0 \\
& g_8(x) = \frac{5x_2}{x_1} - 1 \le 0 \\
& g_9(x) = \frac{x_1}{12x_2} - 1 \le 0 \\
& g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0 \\
& g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0 \\
\textit{where}\quad & 2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,\ 17 \le x_3 \le 28 \\
& 7.3 \le x_4 \le 8.3,\ 7.8 \le x_5 \le 8.3,\ 2.9 \le x_6 \le 3.9 \\
& 5.0 \le x_7 \le 5.5
\end{aligned}
\tag{14}
$$

5. G01

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\
s.t.\ \ & g_1(\boldsymbol{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0 \\
& g_2(\boldsymbol{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0 \\
& g_3(\boldsymbol{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0 \\
& g_4(\boldsymbol{x}) = -8x_1 + x_{10} \le 0 \\
& g_5(\boldsymbol{x}) = -8x_2 + x_{11} \le 0 \\
& g_6(\boldsymbol{x}) = -8x_3 + x_{12} \le 0 \\
& g_7(\boldsymbol{x}) = -2x_4 - x_5 + x_{10} \le 0 \\
& g_8(\boldsymbol{x}) = -2x_6 - x_7 + x_{11} \le 0 \\
& g_9(\boldsymbol{x}) = -2x_8 - x_9 + x_{12} \le 0 \\
where\ \ & 0 \le x_i \le 1 (i = 1, \dots, 9), \\
& 0 \le x_i \le 100 (i = 10, 11, 12), \\
& 0 \le x_{13} \le 1
\end{aligned}
\tag{15}
$$

6. G03

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i \\
s.t.\ \ & h_1(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \\
where\ \ & 0 \le x_i \le 1 (i = 1, \dots, n), n = 10
\end{aligned}
\tag{16}
$$

7. G04

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + \\
& 37.293239x_1 - 40792.141 \\
s.t.\ \ & g_1(\boldsymbol{x}) = 85.334407 + 0.0056858x_2x_5 + \\
& 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0 \\
& g_2(\boldsymbol{x}) = -85.334407 - 0.0056858x_2x_5 - \\
& 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0 \\
& g_3(\boldsymbol{x}) = 80.51249 + 0.0071317x_2x_3 + \\
& 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0 \\
& g_4(\boldsymbol{x}) = -80.51249 - 0.0071317x_2x_5 - \\
& 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0 \\
& g_5(\boldsymbol{x}) = 9.300961 + 0.0047026x_3x_5 + \\
& 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0 \\
& g_6(\boldsymbol{x}) = -9.300961 - 0.0047026x_3x_5 - \\
& 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0 \\
where\ \ & 78 \le x_1 \le 102, 33 \le x_2 \le 45, \\
& 27 \le x_i \le 45 (i = 3, 4, 5)
\end{aligned}
\tag{17}
$$

8. G05

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3 \\
s.t.\ \ & g_1(\boldsymbol{x}) = -x_4 + x_3 - 0.55 \le 0 \\
& g_2(\boldsymbol{x}) = -x_3 + x_4 - 0.55 \le 0 \\
& h_3(\boldsymbol{x}) = 1000 \sin\left(-x_3 - 0.25\right) + \\
& 1000 \sin\left(-x_4 - 0.25\right) + 894.8 - x_1 = 0 \\
& h_4(\boldsymbol{x}) = 1000 \sin\left(x_3 - 0.25\right) + \\
& 1000 \sin\left(x_3 - x_4 - 0.25\right) + 894.8 - x_2 = 0 \\
& h_5(\boldsymbol{x}) = 1000 \sin\left(x_4 - 0.25\right) + \\
& 1000 \sin\left(x_4 - x_3 - 0.25\right) + 1294.8 = 0 \\
where\ \ & 0 \le x_1 \le 1200, 0 \le x_2 \le 1200, -0.55 \le x_3 \le 0.55, \\
& -0.55 \le x_4 \le 0.55
\end{aligned}
\tag{18}
$$

9. G06

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = \left(x_1 - 10\right)^3 + \left(x_2 - 20\right)^3 \\
s.t.\ \ & g_1(\boldsymbol{x}) = -\left(x_1 - 5\right)^2 - \left(x_2 - 5\right)^2 + 100 \le 0 \\
& g_2(\boldsymbol{x}) = \left(x_1 - 6\right)^2 + \left(x_2 - 5\right)^2 - 82.81 \le 0 \\
where\ \ & 13 \le x_1 \le 100, 0 \le x_2 \le 100
\end{aligned}
\tag{19}
$$

10. G07

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + \\
& \left(x_3 - 10\right)^2 + 4\left(x_4 - 5\right)^2 + \left(x_5 - 3\right)^2 + \\
& 2\left(x_6 - 1\right)^2 + 5x_7^2 + 7\left(x_8 - 11\right)^2 + \\
& 2\left(x_9 - 10\right)^2 + \left(x_{10} - 7\right)^2 + 45 \\
s.t.\ \ & g_1(\boldsymbol{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0 \\
& g_2(\boldsymbol{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0 \\
& g_3(\boldsymbol{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0 \\
& g_4(\boldsymbol{x}) = 3\left(x_1 - 2\right)^2 + 4\left(x_2 - 3\right)^2 + 2x_3^2 - 7x_4 - \\
& 120 \le 0 \\
& g_5(\boldsymbol{x}) = 5x_1^2 + 8x_2 + \left(x_3 - 6\right)^2 - 2x_4 - 40 \le 0 \\
& g_6(\boldsymbol{x}) = x_1^2 + 2\left(x_2 - 2\right)^2 - 2x_1 x_2 + 14x_5 - 6x_6 \le 0 \\
& g_7(\boldsymbol{x}) = 0.5\left(x_1 - 8\right)^2 + 2\left(x_2 - 4\right)^2 + 3x_5^2 - x_6 - \\
& 30 \le 0 \\
& g_8(\boldsymbol{x}) = -3x_1 + 6x_2 + 12\left(x_9 - 8\right)^2 - 7x_{10} \le 0 \\
where\ \ & -10 \le x_i \le 10 (i = 1, \ldots, 10)
\end{aligned}
\tag{20}
$$

11. G08

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = -\frac{\sin^3\left(2\pi x_1\right)\sin\left(2\pi x_2\right)}{x_1^3\left(x_1 + x_2\right)} \\
s.t.\ \ & g_1(\boldsymbol{x}) = x_1^2 - x_2 + 1 \le 0 \\
& g_2(\boldsymbol{x}) = 1 - x_1 + \left(x_2 - 4\right)^2 \le 0 \\
where\ \ & 0 \le x_1 \le 10, 0 \le x_2 \le 10
\end{aligned}
\tag{21}
$$

12. G09

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = \left(x_1 - 10\right)^2 + 5\left(x_2 - 12\right)^2 + x_3^4 + 3\left(x_4 - 11\right)^2 \\
& + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6 x_7 - 10x_6 - 8x_7 \\
s.t.\ \ & g_1(\boldsymbol{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0 \\
& g_2(\boldsymbol{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0 \\
& g_3(\boldsymbol{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0 \\
& g_4(\boldsymbol{x}) = 4x_1^2 + x_2^2 - 3x_1 x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0 \\
where\ \ & -10 \le x_i \le 10 (i = 1, \ldots, 7)
\end{aligned}
\tag{22}
$$

13. G10

$$\begin{aligned} min\ \ & f(\boldsymbol{x}) = x_1 + x_2 + x_3 \\ s.t.\ \ & g_1(\boldsymbol{x}) = -1 + 0.0025\left(x_4 + x_6\right) \le 0 \\ & g_2(\boldsymbol{x}) = -1 + 0.0025\left(x_5 + x_7 - x_4\right) \le 0 \\ & g_3(\boldsymbol{x}) = -1 + 0.01\left(x_8 - x_5\right) \le 0 \\ & g_4(\boldsymbol{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - \\ & 83333.333 \le 0 \\ & g_5(\boldsymbol{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \le 0 \\ & g_6(\boldsymbol{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \le 0 \\ where\ \ & 100 \le x_1 \le 10000, 1000 \le x_i \le 10000(i = 2, 3), \\ & 10 \le x_i \le 1000(i = 4, \dots, 8) \end{aligned} \tag{23}$$

14. G11

$$\begin{aligned} min\ \ & f(\boldsymbol{x}) = x_1^2 + \left(x_2 - 1\right)^2 \\ s.t.\ \ & g(\boldsymbol{x}) = x_2 - x_1^2 = 0 \\ where\ \ & -1 \le x_1 \le 1, -1 \le x_2 \le 1 \end{aligned} \tag{24}$$

15. G12

$$\begin{aligned} min\ \ & f(\boldsymbol{x}) = -\frac{\left(100 - \left(x_1 - 5\right)^2 - \left(x_2 - 5\right)^2 - \left(x_3 - 5\right)^2\right)}{100} \\ s.t.\ \ & g(\boldsymbol{x}) = \left(x_1 - p\right)^2 + \left(x_2 - q\right)^2 + \left(x_3 - r\right)^2 - \\ & 0.0625 \le 0 \\ where\ \ & 0 \le x_i \le 10(i = 1, 2, 3), \\ & p, q, r = 1, 2, \dots, 9 \end{aligned} \tag{25}$$

16. G13

$$\begin{aligned} min\ \ & f(\boldsymbol{x}) = e^{x_1x_2x_3x_4x_5} \\ s.t.\ \ & h_1(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ & h_2(\boldsymbol{x}) = x_2x_3 - 5x_4x_5 = 0 \\ & h_3(\boldsymbol{x}) = x_1^3 + x_2^3 + 1 = 0 \\ where\ \ & -2.3 \le x_i \le 2.3(i = 1, 2), -3.2 \le x_i \le 3.2(i = 3, 4, 5) \end{aligned} \tag{26}$$

17. G14

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right) \\
s.t.\ \ & h_1(\boldsymbol{x}) = x_1 + 2x_2 + +2x_3 + x_6 + x_{10} - 2 = 0 \\
& h_2(\boldsymbol{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0 \\
& h_3(\boldsymbol{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0 \\
where\ \ & 0 < x_i \leq 10 (i = 1, \dots, 10), \\
& c_1 = -6.089, c_2 = -17.164, c_3 = -34.054, \\
& c_4 = -5.914, c_5 = -24.721, c_6 = -14.986, \\
& c_7 = -24.1, c_8 = -10.708, c_9 = -26.662, \\
& c_{10} = -22.179
\end{aligned}
\tag{27}
$$

18. G15

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1 x_2 - x_1 x_3 \\
s.t.\ \ & h_1(\boldsymbol{x}) = x_1^2 + x_2^2 + +x_3^2 - 25 = 0 \\
& h_2(\boldsymbol{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0 \\
where\ \ & 0 \leq x_i \leq 10 (i = 1, 2, 3)
\end{aligned}
\tag{28}
$$

19. G17

$$
\begin{aligned}
&min \quad f(\boldsymbol{x}) = f(x_1) + f(x_2) \\
&s.t. \quad h_1(\boldsymbol{x}) = -x_1 + 300 - \frac{x_3 x_4}{131.078}\cos(1.48477 - x_6) + \\
&\frac{0.90798 x_3^2}{131.078}\cos(1.47588) \\
&h_2(\boldsymbol{x}) = -x_2 - \frac{x_3 x_4}{131.078}\cos((1.48477 + x_6) + \\
&\frac{0.90798 x_4^2}{131.078}\cos(1.47588)) \\
&h_3(\boldsymbol{x}) = -x_5 - \frac{x_3 x_4}{131.078}\sin((1.48477 + x_6) + \\
&\frac{0.90798 x_4^2}{131.078}\sin(1.47588)) \\
&h_4(\boldsymbol{x}) = 200 - \frac{x_3 x_4}{131.078}\sin((1.48477 - x_6) + \\
&\frac{0.90798 x_3^2}{131.078}\sin(1.47588)) \\
&f_1(x_1) = \begin{cases} 30x_1 & 0 \le x_1 < 300 \\ 31x_1 & 300 \le x_1 < 400 \end{cases}. \\
&where \quad f_2(x_2) = \begin{cases} 28x_2 & 0 \le x_2 < 100 \\ 29x_2 & 100 \le x_2 < 200 \\ 30x_2 & 200 \le x_2 < 1000 \end{cases}. \\
&0 \le x_1 \le 400, 0 \le x_2 \le 1000, 340 \le x_3 \le 420, \\
&340 \le x_4 \le 420, -1000 \le x_5 \le 1000, 0 \le x_6 \le 0.5236
\end{aligned}
\tag{29}
$$

20. G18

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - \\
& x_5x_9 + x_5x_8 - x_6x_7) \\
s.t.\ \ & g_1(\boldsymbol{x}) = x_3^2 + x_4^2 - 1 \le 0 \\
& g_2(\boldsymbol{x}) = x_9^2 - 1 \le 0 \\
& g_3(\boldsymbol{x}) = x_5^2 + x_6^2 - 1 \le 0 \\
& g_4(\boldsymbol{x}) = x_1^2 + \left(x_2 - x_9\right)^2 - 1 \le 0 \\
& g_5(\boldsymbol{x}) = \left(x_1 - x_5\right)^2 + \left(x_2 - x_6\right)^2 - 1 \le 0 \\
& g_6(\boldsymbol{x}) = \left(x_1 - x_7\right)^2 + \left(x_2 - x_8\right)^2 - 1 \le 0 \\
& g_7(\boldsymbol{x}) = \left(x_3 - x_5\right)^2 + \left(x_4 - x_6\right)^2 - 1 \le 0 \\
& g_8(\boldsymbol{x}) = \left(x_3 - x_7\right)^2 + \left(x_4 - x_8\right)^2 - 1 \le 0 \\
& g_9(\boldsymbol{x}) = x_7^2 + \left(x_8 - x_9\right)^2 - 1 \le 0 \\
& g_{10}(\boldsymbol{x}) = x_2x_3 - x_1x_4 \le 0 \\
& g_{11}(\boldsymbol{x}) = -x_3x_9 \le 0 \\
& g_{12}(\boldsymbol{x}) = x_5x_9 \le 0 \\
& g_{13}(\boldsymbol{x}) = x_6x_7 - x_5x_8 \le 0 \\
where\ \ & -10 \le x_i \le 10(i = 1, \dots, 8), 0 \le x_9 \le 20
\end{aligned}
\tag{30}
$$

21. G19

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = \sum_{j=1}^{5}\sum_{i=1}^{5} c_{ij}x_{(10+i)}x_{(10+j)} + 2\sum_{j=1}^{5} d_j x_{(10+j)}^3 \\
& - \sum_{i=1}^{10} b_i x_i \\
s.t.\ \ & g_j(\boldsymbol{x}) = -2\sum_{i=1}^{5} c_{ij}x_{(10+i)} - 3d_j x_{(10+j)}^2 - \\
& e_j + \sum_{i=1}^{10} a_{ij}x_i \le 0 \quad j = 1, \dots, 5 \\
where\ \ & \boldsymbol{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1], \\
& \text{See Table 11} \\
& 0 \le x_i \le 10(i = 1, \dots, 15)
\end{aligned}
\tag{31}
$$

Table 11
Data set for test problem G19

j	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
$c_{1,j}$	30	-20	-10	32	-10
$c_{2,j}$	-20	39	-6	-31	32
$c_{3,j}$	-10	-6	10	-6	-10
$c_{4,j}$	32	-31	-6	39	-20
$c_{5,j}$	-10	32	-10	-20	30
d_j	4	8	10	6	2
$a_{1,j}$	-16	2	0	1	0
$a_{2,j}$	0	-2	0	0.4	2
$a_{3,j}$	-3.5	0	2	0	0
$a_{4,j}$	0	-2	0	-4	-1
$a_{5,j}$	0	-9	-2	1	-2.8
$a_{6,j}$	2	0	-4	0	0
$a_{7,j}$	-1	-1	-1	-1	-1
$a_{8,j}$	-1	-2	-3	-2	-1
$a_{9,j}$	1	2	3	4	5
$a_{10,j}$	1	1	1	1	1

22. G21

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = x_1 \\
s.t.\ \ & g_1(\boldsymbol{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \le 0 \\
& h_1(\boldsymbol{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + \\
& 25x_4x_6 + x_3x_4 = 0 \\
& h_2(\boldsymbol{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - \\
& 25x_4x_7 - 15536.5 = 0 \\
& h_3(\boldsymbol{x}) = -x_5 + \ln\left(-x_4 + 900\right) = 0 \\
& h_4(\boldsymbol{x}) = -x_6 + \ln\left(x_4 + 300\right) = 0 \\
& h_5(\boldsymbol{x}) = -x_7 + \ln\left(-2x_4 + 700\right) = 0 \\
where\ \ & 0 \le x_1 \le 1000, 0 \le x_2, x_3 \le 40, 100 \le x_4 \le 300, \\
& 6.3 \le x_5 \le 6.7, 5.9 \le x_6 \le 6.4, 4.5 \le x_7 \le 6.25
\end{aligned}
\tag{32}
$$

23. G23

$$\begin{aligned} min\ \ & f(\boldsymbol{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10\left(x_6 + x_7\right) \\ s.t.\ \ & g_1(\boldsymbol{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \le 0 \\ & g_2(\boldsymbol{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \le 0 \\ & h_1(\boldsymbol{x}) = x_1 + x_2 - x_3 - x_4 = 0 \\ & h_2(\boldsymbol{x}) = 0.03x_1 + 0.01x_2 - x_9\left(x_3 + x_4\right) = 0 \\ & h_3(\boldsymbol{x}) = x_3 + x_6 - x_5 = 0 \\ & h_4(\boldsymbol{x}) = x_4 + x_7 - x_8 = 0 \\ where\ \ & 0 \le x_1, x_2, x_6 \le 300, 0 \le x_3, x_5, x_7 \le 100, \\ & 0 \le x_4, x_8 \le 200, 0.01 \le x_9 \le 0.03 \end{aligned} \tag{33}$$

24. G24

$$\begin{aligned} min\ \ & f(\boldsymbol{x}) = -x_1 - x_2 \\ s.t.\ \ & g_1(\boldsymbol{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \le 0 \\ & g_2(\boldsymbol{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \le 0 \\ where\ \ & 0 \le x_1 \le 3, 0 \le x_2 \le 4 \end{aligned} \tag{34}$$

25. G16

$$
\begin{aligned}
min\ \ & f(\boldsymbol{x}) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + \\
& 0.0321y_{12} + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17} \\
s.t.\ \ & g_1(\boldsymbol{x}) = \tfrac{0.28}{0.72}y_5 - y_4 \le 0 \qquad g_2(\boldsymbol{x}) = x_3 - 1.5x_2 \le 0 \\
& g_3(\boldsymbol{x}) = 3496\tfrac{y_2}{c_{12}} - 21 \le 0 \qquad g_4(\boldsymbol{x}) = 110.6 + y_1 - \tfrac{62212}{c_{17}} \le 0 \\
& g_5(\boldsymbol{x}) = 213.1 - y_1 \le 0 \qquad g_6(\boldsymbol{x}) = y_1 - 405.23 \le 0 \\
& g_7(\boldsymbol{x}) = 17.505 - y_2 \le 0 \qquad g_8(\boldsymbol{x}) = y_2 - 1053.6667 \le 0 \\
& g_9(\boldsymbol{x}) = 11.275 - y_3 \le 0 \qquad g_{10}(\boldsymbol{x}) = y_3 - 35.03 \le 0 \\
& g_{11}(\boldsymbol{x}) = 214.228 - y_4 \le 0 \qquad g_{12}(\boldsymbol{x}) = y_4 - 665.585 \le 0 \\
& g_{13}(\boldsymbol{x}) = 7.458 - y_5 \le 0 \qquad g_{14}(\boldsymbol{x}) = y_5 - 584.463 \le 0 \\
& g_{15}(\boldsymbol{x}) = 0.961 - y_6 \le 0 \qquad g_{16}(\boldsymbol{x}) = y_6 - 265.916 \le 0 \\
& g_{17}(\boldsymbol{x}) = 1.612 - y_7 \le 0 \qquad g_{18}(\boldsymbol{x}) = y_7 - 7.046 \le 0 \\
& g_{19}(\boldsymbol{x}) = 0.146 - y_8 \le 0 \qquad g_{20}(\boldsymbol{x}) = y_8 - 0.222 \le 0 \\
& g_{21}(\boldsymbol{x}) = 107.99 - y_9 \le 0 \qquad g_{22}(\boldsymbol{x}) = y_9 - 273.366 \le 0 \\
& g_{23}(\boldsymbol{x}) = 922.693 - y_{10} \le 0 \qquad g_{24}(\boldsymbol{x}) = y_{10} - 1286.105 \le 0 \\
& g_{25}(\boldsymbol{x}) = 926.832 - y_{11} \le 0 \qquad g_{26}(\boldsymbol{x}) = y_{11} - 1444.046 \le 0 \\
& g_{27}(\boldsymbol{x}) = 18.766 - y_{12} \le 0 \qquad g_{28}(\boldsymbol{x}) = y_{12} - 537.141 \le 0 \\
& g_{29}(\boldsymbol{x}) = 1072.163 - y_{13} \le 0 \qquad g_{30}(\boldsymbol{x}) = y_{13} - 3247.039 \le 0 \\
& g_{31}(\boldsymbol{x}) = 8961.448 - y_{14} \le 0 \qquad g_{32}(\boldsymbol{x}) = y_{14} - 26844.086 \le 0 \\
& g_{33}(\boldsymbol{x}) = 0.063 - y_{15} \le 0 \qquad g_{34}(\boldsymbol{x}) = y_{15} - 0.386 \le 0 \\
& g_{35}(\boldsymbol{x}) = 71084.33 - y_{16} \le 0 \qquad g_{36}(\boldsymbol{x}) = -140000 + y_{16} \le 0 \\
& g_{37}(\boldsymbol{x}) = 2802713 - y_{17} \le 0 \qquad g_{38}(\boldsymbol{x}) = y_{17} - 12146108 \le 0 \\
& y_1 = x_2 + x_3 + 41.6 \qquad y_2 = \tfrac{12.5}{c_1} + 12 \\
& y_3 = \tfrac{c_2}{c_3} \qquad y_4 = 19y_3 \\
& y_6 = x_1 - y_5 - y_4 - y_3 \qquad y_5 = c_6c_7 \\
& y_7 = \tfrac{c_8}{y_1} \qquad y_8 = \tfrac{c_8}{3798} \\
& y_9 = \tfrac{96.82}{c_9} + 0.321y_1 \\
& y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6 \\
& y_{11} = 1.71x_1 - 0.452y_4 + 0.580y_3 \\
& y_{12} = c_{10}x_1 + \tfrac{c_{11}}{c_{12}}y_{13} = c_{12} - 1.75y_2 \\
& y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \tfrac{146312}{y_9 + x_5} \quad y_{15} = \tfrac{y_{13}}{c_{13}} \\
& y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13} \\
& y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \tfrac{c_{14}}{c_{12}} \\
where\ \ & c_1 = 0.024x_4 - 4.62 \\
& c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1 \\
& c_3 = 0.052x_1 + 78 + 0.002377y_2x_1 \\
& c_4 = 0.04782\left(x_1 - y_3\right) + \tfrac{0.1956\left(x_1 - y_3\right)^2}{x_2} + 0.6376y_4 + 1.594y_3 \\
& c_5 = 100x_2 \qquad c_6 = x_1 - y_3 - y_4 \\
& c_7 = 0.950 - \tfrac{c_4}{c_5} \qquad c_8 = \left(y_5 + y_4\right)0.995 \\
& c_9 = y_7 - \tfrac{0.0663y_7}{y_8} - 0.3153 \qquad c_{10} = \tfrac{12.3}{752.3} \\
& c_{11} = \left(1.75y_2\right)\left(0.995x_1\right) \qquad c_{12} = 0.995y_{10} + 1998 \\
& c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095 \\
& c_{14} = 2324y_{10} - 28740000y_2 \qquad c_{15} = \tfrac{y_{13}}{y_{15}} - \tfrac{y_{13}}{0.52} \\
& c_{16} = 1.104 - 0.72y_{15} \qquad c_{17} = y_9 + x_5 \\
& 704.4148 \le x_1 \le 906.3855 \qquad 68.6 \le x_2 \le 288.88 \\
& 0 \le x_3 \le 134.75 \qquad 193 \le x_4 \le 287.0966 \\
& 25 \le x_5 \le 84.1988
\end{aligned}
\tag{35}
$$

References

[1] J. Gu, G. Li, N. Gan, Hybrid metamodel-based design space management method for expensive problems, Engineering Optimizationdoi:10.1080/0305215X.2016.1261126.

[2] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

[3] J. H. Holland, Adaptation in Natural and Artificial Systems: An introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1975. doi:10.1137/1018105.

[4] R. H. Myers, D. T. Montgomery, G. G. Vining, C. M. Borror, S. M. Kowalski, Response Surface Methodology: A Retrospective and Literature Survey, Journal of Quality Technology 36 (1) (2004) 53–78. doi:10.1080/00224065.2004.11980252.

[5] N. Dyn, D. Levin, S. Rippa, Numerical procedures for surface fitting of scattered data by radial functions, SIAM Journal on Scientific and Statistical Computing 7 (2) (1986) 639–659. doi:10.1137/0907043.

[6] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and Analysis of Computer Experiments, Statistical Science 4 (4) (1989) 409–423. doi:10.1214/ss/1177012413.

[7] J. H. Friedman, Multivariate Adaptive Regression Splines, The Annals of Statistics 19 (1) (1991) 1–67.

[8] D. E. Rumelhart, B. Widrow, M. A. Lehr, The Basic Ideas in Neural Networks, Communications of the ACM 37 (3) (1994) 87–92. doi:10.1145/175247.175256.

[9] S. M. Clarke, J. H. Griebsch, T. W. Simpson, Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses, Transactions of ASME, Journal of Mechanical Design 127 (6) (2005) 1077–1087. doi:10.1115/1.1897403.

[10] R. Jin, W. Chen, T. W. Simpson, Comparative studies of metamodelling techniques under multiple modelling criteria, Structural and Multidisciplinary Optimization 23 (1) (2001) 1–13. doi:10.1007/s00158-001-0160-4.

[11] B. S. Kim, Y. B. Lee, D. H. Choi, Comparison study on the accuracy of metamodeling technique for non-convex functions, Journal of Mechanical Science and Technology 23 (4) (2009) 1175–1181. doi:10.1007/s12206-008-1201-3.

[12] T. W. Simpson, J. Peplinsk, P. N. Koch, J. K. Allen, Metamodels for Computer-Based Engineering Design: Survey and Recommendations, Engineering with Computers 17 (2) (2001) 129–150. arXiv:arXiv:1011.1669v3, doi:10.1017/CBO9781107415324.004.

[13] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. Kevin Tucker, Surrogate-based analysis and optimization, Progress in Aerospace Sciences 41 (1) (2005) 1–28. doi:10.1016/j.paerosci.2005.02.001.

[14] D. R. Jones, M. Schonlau, W. J. Welch, Efficient Global Optimization of Expensive Black-Box Functions, Journal of Global Optimization 13 (1998) 455–492.

[15] A. Younis, Z. Dong, Metamodelling and search using space exploration and unimodal region elimination for design optimization, Engineering Optimization 42 (6) (2010) 517–533. doi:10.1080/03052150903325540.

[16] S. M. Wild, R. G. Regis, C. A. Shoemaker, ORBIT: Optimization by Radial Basis Function Interpolation in Trust-Regions, SIAM J. SCI. COMPUT. 30 (6) (2008) 3197–3219.

[17] C. Setzkorn, A. F. Taktak, B. E. Damato, On the use of multi-objective evolutionary algorithms for survival analysis, BioSystems 87 (1) (2007) 31–48. doi:10.1016/j.biosystems.2006.03.002.

[18] K. S. Won, T. Ray, A framework for design optimization using surrogates, Engineering Optimization 37 (7) (2005) 685–703. doi:10.1080/03052150500211911.

[19] A. Shahrokhi, A. Jahangirian, A surrogate assisted evolutionary optimization method with application to the transonic airfoil design, Engineering Optimization 42 (6) (2010) 497–515. doi:10.1080/03052150903305468.

[20] R. G. Regis, Particle swarm with radial basis function surrogates for expensive black-box optimization, Journal of Computational Science 5 (1) (2014) 12–23. doi:10.1016/j.jocs.2013.07.004.

[21] C. Sun, Y. Jin, J. Zeng, Y. Yu, A two-layer surrogate-assisted particle swarm optimization algorithm, Soft Computing 19 (6) (2015) 1461–1475. doi:10.1007/s00500-014-1283-z.

[22] X. Cai, H. Qiu, L. Gao, C. Jiang, X. Shao, An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems, Knowledge-Based Systems 184 (2019) 104901. doi:10.1016/j.knosys.2019.104901.

[23] R. T. Haftka, D. Villanueva, A. Chaudhuri, Parallel surrogate-assisted global optimization with expensive functions – a survey, Structural and Multidisciplinary Optimization 54 (1) (2016) 3–13. doi:10.1007/s00158-016-1432-3.

[24] J. Müller, J. D. Woodbury, GOSAC: global optimization with surrogate approximation of constraints, Journal of Global Optimization 69 (1) (2017) 117–136. doi:10.1007/s10898-017-0496-y.

[25] A. I. Forrester, A. Sobester, A. Keane, Engineering design via surrogate modelling: a practical guide, Wiley, 2008.

[26] T. P. Runarsson, Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models, in: X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Ti\vno, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII, Springer, Berlin, Heidelberg, Berlin, Heidelberg, 2004, pp. 401–410. doi:10.1007/978-3-540-30217-9_41.

[27] L. Shi, K. Rasheed, ASAGA: An adaptive surrogate-assisted genetic algorithm, GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008 (2008) 1049–1056.

[28] Y. Tang, J. Chen, J. Wei, A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions, Engineering Optimization 45 (5) (2013) 557–576. doi:10.1080/0305215X.2012.690759.

[29] M. J. Powell, A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation, Springer, Dordrecht, Dordrecht, 1994. doi:10.1007/978-94-015-8330-5_4.

[30] R. Brekelmans, L. Driessen, H. Hamers, D. Den Hertog, Constrained optimization involving expensive function evaluations: A sequential approach, European Journal of Operational Research 160 (1) (2005) 121–138. doi:10.1016/j.ejor.2003.10.009.

[31] A. Basudhar, C. Dribusch, S. Lacaze, S. Missoum, Constrained efficient global optimization with support vector machines, Structural and Multidisciplinary Optimization 46 (2) (2012) 201–221. doi:10.1007/s00158-011-0745-5.

[32] R. G. Regis, Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions, Computers and Operation Research 38 (5) (2011) 837–853. doi:10.1016/j.cor.2010.09.013.

[33] R. G. Regis, Constrained Optimization by Radial Basis Function Interpolation for High-Dimensional Expensive Black-Box Problems with Infeasible Initial Points, Engineering Optimization 46 (2) (2014) 218–243. doi:10.1080/0305215X.2013.765000.

[34] S. Bagheri, W. Konen, M. Emmerich, T. Bäck, Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets, Applied Soft Computing Journal 61 (2017) 377–393. doi:10.1016/j.asoc.2017.07.060.

[35] H. Dong, B. Song, Z. Dong, P. Wang, SCGOSR : Surrogate-based constrained global optimization using space reduction, Applied Soft Computing Journal 65 (2018) 462–477. doi:10.1016/j.asoc.2018.01.041.

[36] L. Jiao, L. Li, R. Shang, F. Liu, R. Stolkin, A novel selection evolutionary strategy for constrained optimization, Information Sciences 239 (2013) 122–141. doi:10.1016/j.ins.2013.03.002.

[37] J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. A. C. Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, Nangyang Technological University, Singapore, Tech. Rep 41.

[38] Y. Li, Y. Wu, J. Zhao, L. Chen, A Kriging-based constrained global optimization algorithm for expensive black-box functions with infeasible initial points, Journal of Global Optimization 67 (1-2) (2017) 343–366. doi:10.1007/s10898-016-0455-z.

[39] E. Zahara, Y. T. Kao, Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems, Expert Systems with Applications 36 (2009) 3880–3886. doi:10.1016/j.eswa.2008.02.039.

[40] S. Kitayama, M. Arakawa, K. Yamazaki, Sequential Approximate Optimization using Radial Basis Function network for engineering optimization, Optimization and Engineering 12 (4) (2011) 535–557. doi:10.1007/s11081-010-9118-y.

[41] A. Polynkin, V. V. Toropov, Mid-range metamodel assembly building based on linear regression for large scale optimization problems, Structural and Multidisciplinary Optimization 45 (4) (2012) 515–527. doi:10.1007/s00158-011-0692-1.

[42] D. Liu, V. V. Toropov, Implementation of Discrete Capability into the Enhanced Multipoint Approximation Method for Solving Mixed Integer-Continuous Optimization Problems, International Journal of Computational Methods in Engineering Science and Mechanics 17 (1). doi:10.1080/15502287.2016.1139013.

[43] A.-b. Ryberg, R. D. Bäckryd, N. Larsgunnar, Metamodel-Based Multidisciplinary Design Optimization for Automotive Applications, no. September, 2012. doi:LIU-IEI-R-12/003.

[44] R. G. Regis, S. M. Wild, CONORBIT: constrained optimization by radial basis function interpolation in trust regions, Optimization Methods and Software 32 (3) (2017) 552–580. doi:10.1080/10556788.2016.1226305.

[45] S. M. Wild, C. Shoemaker, Global convergence of radial basis function trust region derivative-free algorithms, SIAM Journal on Optimization 21 (3) (2011) 761–781. doi:10.1137/09074927X.

[46] J. Eric, O. Travis, P. Peterson, SciPy: Open source scientific tools for Python (2001).

[47] F. van Keulen, V. V. Toropov, V. Markine, Recent refinements in the multi-point approximation method in conjunction with adaptive mesh refinement, Design Engineering Technical Conference (1996) 1–12.

[48] A. H. Aguirre, A. E. M. Zavala, E. V. Diharce, S. B. Rionda, COPSO : Constrained Optimization via PSO algorithm, Tech. rep., Center for Research in Mathematics. CIMAT (2007).

[49] T. Okamoto, H. Hirata, Constrained optimization using the quasi-chaotic optimization method with the exact penalty function and the sequential quadratic programming, in: Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, IEEE, 2011, pp. 1765–1770. doi:10.1109/ICSMC.2011.6083927.

[50] H. Liu, S. Xu, X. Chen, X. Wang, Q. Ma, Constrained global optimization via a DIRECT-type constraint-handling technique and an adaptive metamodeling strategy, Structural and Multidisciplinary Optimization 55 (1) (2016) 155–177. doi:10.1007/s00158-016-1482-6.

[51] R. d. P. Garcia, B. S. L. P. de Lima, A. C. d. C. Lemonge, A Surrogate Assisted Differential Evolution to Solve Constrained Optimization Problems, in: 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), IEEE, Arequipa, Peru, 2017, pp. 1–6.

[52] D. R. Jones, Large-Scale Multi-Disciplinary Mass Optimization in the Auto Industry, Tech. rep. (2008).

[53] N. H. Awad, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization, Information Sciences 451-452 (2018) 326–347. doi:10.1016/j.ins.2018.04.024.

[54] M. Kazemi, G. G. Wang, Metamodel-Based Optimization for Problems With Expensive Objective and Constraint Funtions, Journal of Mechanical Design 133. doi:10.1115/1.4003035.