A Surrogate-Model-Assisted Evolutionary Algorithm for Computationally Expensive Design Optimization Problems with Inequality Constraints

Bo Liu, Qingfu Zhang, and Georges Gielen

Abstract The surrogate model-aware evolutionary search (SMAS) framework is a newly emerged model management method for surrogate-model-assisted evolutionary algorithms (SAEAs), which shows clear advantages on necessary number of exact evaluations. However, SMAS aims to solve unconstrained or bound constrained computationally expensive optimization problems. In this chapter, an SMAS-based efficient constrained optimization method is presented. Its major components include: (1) an SMAS-based SAEA framework for handling inequality constraints, (2) a ranking and diversity maintenance method for addressing complicated constraints, and (3) an adaptive surrogate model updating (ASU) method to address many constraints, which considerably reduces the computational overhead of surrogate modeling. Empirical studies on complex benchmark problems and a real-world mm-wave integrated circuit design optimization problem are reported in this chapter. The results show that to obtain comparable results, the presented method only needs 1–10 % of the exact function evaluations typically used by the standard evolutionary algorithms with popular constraint handling techniques.

Keywords Surrogate model assisted evolutionary computation • Constrained optimization • Constraint handling • Expensive optimization • Gaussian Process • Surrogate modeling • mm-wave IC synthesis

MSC codes: 00A06, 41A30, 60G15, 62L05, 68T20

B. Liu (✉)
Department of Computing, Glyndwr University, Wrexham, UK
e-mail: b.liu@glyndwr.ac.uk; Bo.Liu@esat.kuleuven.be

Q. Zhang
City University of Hong Kong, Kowloon, Hong Kong SAR
e-mail: qingfu.zhang@cityu.edu.hk

G. Gielen
ESAT-MICAS, Katholieke Universiteit Leuven, Leuven, Belgium
e-mail: Georges.Gielen@esat.kuleuven.be

S. Koziel et al. (eds.), *Simulation-Driven Modeling and Optimization*,
Springer Proceedings in Mathematics & Statistics 153,
DOI 10.1007/978-3-319-27517-8_14

1 Introduction

Many industrial design optimization problems require expensive simulations for evaluating their candidate solutions. Employing surrogate models to replace computationally expensive exact function evaluations is a routine approach to address these problems [22]. Because many real-world problems have constraints, constrained expensive optimization is receiving increasing attention. However, most research works focus on computationally expensive local optimization [1, 12]. This chapter focuses on handling global optimization problems with both expensive simulations (i.e., only very few exact evaluations are allowed) and complex inequality constraints, which can be found in many real-world applications such as mm-wave integrated circuit design (e.g., [19]). Constraints in many industrial design optimization problems come from design specifications (e.g., $power \leq 1.5\,\mathrm{mW}$) which are inequality constraints and equality constraints are often self-contained in the simulator (e.g., Maxwell's equations). For complex constraints, this chapter mainly aims at a number of constraints, active constraints (tight design specifications), disconnected feasible region, and complex (sophisticated landscape) constraint functions.

Surrogate-model-assisted evolutionary algorithms (SAEAs) have been accepted as an effective approach to deal with expensive optimization. SAEAs take advantages of both evolutionary algorithms (EAs) and surrogate modeling techniques. To develop an SAEA for constrained expensive optimization problems, one must consider three highly related issues:

- Which surrogate modeling method should be used to approximate the objective function and the constraints?
- Which SAEA framework should be used?
- How should the constraints be handled?

The Gaussian process (GP) modeling is one of the most popular surrogate modeling methods used in SAEAs. Some principled expensive optimization approaches with the GP model and with prescreening, such as the efficient global optimization (EGO) method [10], have been well investigated and documented. Moreover, very few empirical parameters are necessary in a GP model, making the surrogate modeling more controllable. Due to these, the GP modeling is adopted for approximating the objective function and the constraints.

To deal with the second issue, several SAEA frameworks have been proposed for accommodating surrogate models. Successful examples include the surrogate-model-assisted memetic evolutionary search (SMMS) framework [16, 34], the meta-model-assisted EA (MAEA) framework [5], and the surrogate model-aware evolutionary search (SMAS) framework [20]. These frameworks balance the surrogate model quality and the optimization efficiency in different manners and have been tested mainly on unconstrained optimization problems. The SMAS framework considers EA-driven function optimization and high-quality surrogate model construction at the same time by controlling the locations of the generated

candidate solutions. It has shown clear advantages (up to eight times fewer exact evaluations) over the SMMS and the MAEA frameworks on a set of widely used unconstrained test instances with 20–50 variables in terms of solution quality with a limited number of exact function evaluations [20]. Therefore, SMAS is especially suitable for quite expensive industrial design optimization problems. For this reason, the SMAS framework is selected.

With regard to constraint handling, a number of techniques have been suggested and used in EAs for general (often *inexpensive*) constrained optimization. Besides static penalty function-based methods and the superiority of the feasibility (SF) method [3], some advanced constraint handling methods have been developed for handling complex constraints [21, 25, 31, 32], such as the self-adaptive penalty function-based methods [31], stochastic ranking-based methods [25], and multi-objective ranking-based methods [32]. All these techniques aim at maintaining the population diversity while driving their populations from infeasible region to feasible one by adaptively trading off the objective function optimization and the total constraint violation minimization. In the context of expensive optimization, some general constraint handling methods have successfully been applied to constrained expensive optimization (e.g., [5, 7]). Several prescreening methods have been generalized from unconstrained expensive optimization to constrained expensive optimization [5, 6, 28]. Some modeling methods and model updating methods for constraint function satisfaction and objective function optimization have been developed [2, 14, 26]. In addition, surrogate-model-assisted expensive integer nonlinear programming has been investigated [11].

One focus of this chapter is to handle complex constraints in an efficient manner (i.e., using the SMAS framework). It is not straightforward to combine the above advanced constraint handling methods for inexpensive and expensive constrained optimization with SMAS. The diversity maintenance methods in most advanced constraint handling methods rely on the population updating of a standard EA [21], while the population updating of SMAS is completely different and is critical for its efficiency.

Another focus is to reduce the computational overhead of surrogate modeling. Independent modeling of the constraint functions is needed for constrained expensive optimization problems [5]. When the number of decision variables is large (e.g., 20–50 variables), surrogate model construction itself may cost a few minutes for a single function in some cases (e.g., [20]), and it should be conducted at each iteration for both the objective function and all the constraints. Thus, the computational cost of surrogate modeling can be tremendous, especially for problems with many constraints.

To address these challenges, an improved SMAS framework for efficient constrained expensive optimization, a diversity maintenance method for the SMAS framework to handle complex constraints, and an adaptive surrogate model updating (ASU) method for adaptively saving the computational overhead of surrogate modelling are introduced. Using these three techniques, a Gaussian Process SAEA for computationally expensive inequality constrained optimization problems (GPEEC) is constructed. Empirical studies on 8 benchmark problems that are challenging in

terms of constraint handling, a self-developed 20-dimensional benchmark problem whose objective function is highly multimodal and whose constraint function is very complex, and a real-world mm-wave integrated circuit design optimization problem are used as examples. Results show that comparable solution quality is obtained compared to the state-of-the-art constrained optimization methods (without surrogate models), and that only 1–10 % of the number of exact function evaluations are needed compared to the standard EA with the popular SF method.

The remainder of this chapter is organized as follows. Section 2 introduces the basic techniques. The general framework of GPEEC is then presented in Section 3. Sections 4–6 provide details of the algorithm. Section 7 presents the experimental results of GPEEC. The parameter settings of GPEEC are also discussed. The summary is presented in Section 8.

2 Problem Definition and Basic Techniques

2.1 Problem Definition

The following constrained optimization problem is considered in this chapter:

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) \\ &\text{subject to} \quad g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m. \\ &\qquad\qquad \mathbf{x} \in [a, b]^d, \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is the objective function, $g_i(\mathbf{x}) \leq 0$ $(i = 1, \ldots, m)$ are the constraints, and $[a, b]^d$ is the search region. We assume that some constraints $g_i(\mathbf{x}) \leq 0$ can be active. In other words, these constraints become almost equalities at the globally optimal solution. The problem can have disconnected feasible regions, the function of $f(\mathbf{x})$ can be highly multimodal and the function landscape of $g_i(\mathbf{x})$ can be quite complex. We further assume that the calculations of $f(\mathbf{x})$ and the different $g_i(\mathbf{x})$ can be done in a single simulation, which is the case for many real-world expensive optimization problems (e.g., [18]), or can be done in parallel considering the rapid development of parallel computation techniques.

2.2 GP Modeling

To model an unknown function $y = f(\mathbf{x}), \mathbf{x} \in R^d$, the GP modeling assumes that $f(\mathbf{x})$ at any point $\mathbf{x}$ is a Gaussian random variable $N(\mu, \sigma^2)$, where μ and σ are two constants independent of $\mathbf{x}$. For any $\mathbf{x}$, $f(\mathbf{x})$ is a sample of $\mu + \epsilon(\mathbf{x})$, where $\epsilon(\mathbf{x}) \sim N(0, \sigma^2)$. For any $\mathbf{x}, \mathbf{x}' \in R^d$, $c(\mathbf{x}, \mathbf{x}')$, the correlation between $\epsilon(\mathbf{x})$ and $\epsilon(\mathbf{x}')$, depends on $\mathbf{x} - \mathbf{x}'$. More precisely,

$$c(\mathbf{x}, \mathbf{x}') = \exp(-\sum_{i=1}^{d} \theta_i |x_i - x_i'|^{p_i}), \tag{2}$$

where parameter $1 \leq p_i \leq 2$ is related to the smoothness of $f(\mathbf{x})$ with respect to x_i, and parameter $\theta_i > 0$ indicates the importance of x_i on $f(\mathbf{x})$. More details of the GP modeling can be found in [24].

2.2.1 Hyper Parameter Estimation

Given K points $\mathbf{x}^1, \ldots, \mathbf{x}^K \in R^d$ and their f-function values $y^1, \ldots, y^K$, then the hyper parameters μ, σ, $\theta_1, \ldots, \theta_d$, and $p_1, \ldots, p_d$ can be estimated by maximizing the likelihood that $f(\mathbf{x}) = y^i$ at $\mathbf{x} = \mathbf{x}^i$ $(i = 1, \ldots, K)$ [10]:

$$\frac{1}{(2\pi\sigma^2)^{K/2}\sqrt{det(C)}} \exp\left[-\frac{(\mathbf{y} - \mu\mathbf{1})^T C^{-1}(\mathbf{y} - \mu\mathbf{1})}{2\sigma^2}\right] \tag{3}$$

where C is a $K \times K$ matrix whose (i, j)-element is $c(\mathbf{x}^i, \mathbf{x}^j)$, $\mathbf{y} = (y^1, \ldots, y^K)^T$ and $\mathbf{1}$ is a K-dimensional column vector of ones.

To maximize (3), the values of μ and σ^2 must be:

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

and

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^T C^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})}{K}. \tag{5}$$

Substituting (4) and (5) into (3) eliminates the unknown parameters μ and σ from (3). As a result, the likelihood function depends only on θ_i and p_i for $i = 1, \ldots, d$. Equation (3) can then be maximized to obtain estimates of $\hat{\theta}_i$ and $\hat{p}_i$. The estimates $\hat{\mu}$ and $\hat{\sigma}^2$ can then readily be obtained from (4) and (5).

2.2.2 The Best Linear Unbiased Prediction and Predictive Distribution

Given the hyperparameter estimates $\hat{\theta}_i$, $\hat{p}_i$, $\hat{\mu}$, and $\hat{\sigma}^2$, one can predict $y = f(\mathbf{x})$ at any untested point $\mathbf{x}$ based on the f-function values y^i at $\mathbf{x}^i$ for $i = 1, \ldots, K$. The best linear unbiased predictor of $f(x)$ is [10, 27]:

$$\hat{f}(\mathbf{x}) = \hat{\mu} + \mathbf{r}^T C^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{6}$$

and its mean squared error is:

$$s^2(\mathbf{x}) = \hat{\sigma}^2[1 - \mathbf{r}^T C^{-1}\mathbf{r} + \frac{(1 - \mathbf{1}^T C^{-1}\mathbf{r})^2}{\mathbf{1}^T C^{-1}\mathbf{r}}] \tag{7}$$

where $\mathbf{r} = (c(\mathbf{x}, \mathbf{x}^1), \ldots, c(\mathbf{x}, \mathbf{x}^K))^T$. $N(\hat{f}(\mathbf{x}), s^2(\mathbf{x}))$ can be regarded as a predictive distribution for $f(\mathbf{x})$ given the function values y^i at $\mathbf{x}^i$ for $i = 1, \ldots, K$.

More detailed derivations about the hyperparameter estimation and prediction can be found in [9].

2.2.3 Lower Confidence Bound

We consider minimization of $f(\mathbf{x})$ in this chapter. Given the predictive distribution $N(\hat{f}(\mathbf{x}), s^2(\mathbf{x}))$ for $f(\mathbf{x})$, a lower confidence bound (LCB) of $f(\mathbf{x})$ can be defined as [4]:

$$f_{lcb}(\mathbf{x}) = \hat{f}(\mathbf{x}) - \omega s(\mathbf{x}) \tag{8}$$

where ω is a predefined constant. In the GPEEC algorithm, $f_{lcb}(\mathbf{x})$ instead of $\hat{f}(\mathbf{x})$ itself is used to measure the quality of $\mathbf{x}$. The use of LCB can balance the search between promising areas (i.e., with low $\hat{f}(\mathbf{x})$ values) and less explored areas (i.e., with high $s(\mathbf{x})$ values). Following the suggestion in [4, 5], $\omega = 2$ is used for balancing the exploration and exploitation of LCB.

2.3 *Differential Evolution*

The differential evolution (DE) algorithm is used as the search engine in the GPEEC algorithm. DE is an effective and popular global optimization algorithm. It uses a differential operator to create new candidate solutions [23]. There are quite a few different DE variants and DE/best/1 is used here to generate new solutions for prescreening. The DE/best/1 mutation uses the current best solution as the base vector, so as to increase the speed of generating promising candidates.

Suppose that P is a population and the best individual in P is $\mathbf{x}^{best}$. Let $\mathbf{x} = (x_1, \ldots, x_d) \in R^d$ be an individual solution in P. To generate a child solution $\mathbf{u} = (u_1, \ldots, u_d)$ for $\mathbf{x}$, DE/best/1 works as follows.

A donor vector is first produced by mutation:

$$\mathbf{v} = \mathbf{x}^{best} + F \cdot (\mathbf{x}^{r_1} - \mathbf{x}^{r_2}) \tag{9}$$

where $\mathbf{x}^{r_1}$ and $\mathbf{x}^{r_2}$ are two different solutions randomly selected from P and also different from $\mathbf{x}^{best}$. $F \in (0, 2]$ is a control parameter, often called the scaling factor [23]. Then the following crossover operator is applied to produce $\mathbf{u}$:

1. Randomly select a variable index $j_{rand} \in \{1, \ldots, d\}$,
2. For each $j = 1$ to d, generate a uniformly distributed random number *rand* from $(0, 1)$ and set:

$$u_j = \begin{cases} v_j, \text{ if } (rand \leq CR) | j = j_{rand} \\ x_j, \text{ otherwise} \end{cases} \tag{10}$$

where $CR \in [0, 1]$ is a predefined constant called the crossover rate.

3 Algorithm Framework

As described above, GPEEC adopts and improves the SMAS framework originally proposed for unconstrained expensive optimization [20]. GPEEC maintains a database and iteratively updates surrogate models for the objective function and the constraints until a stopping criterion is met.

- The database is composed of all the evaluated solutions and their exact function values. At the first step, α solutions from $[a, b]^d$ are sampled by an experimental design method and are evaluated (through exact function evaluations) to form the initial database.
- Surrogate models for the objective function and each constraint are constructed at the first step and are then updated at the consecutive iterations.

In each iteration, GPEEC works as follows:

Step 1: Selecting working population: Select the λ best solutions from the current database to form a population P.

Step 2: Diversity maintenance: Check the diversity of P. When necessary, conduct diversity enhancement operations on P.

Step 3: Generating child population: Apply evolutionary operators on P to generate λ child solutions.

Step 4: Prescreening of child solutions: Adaptively update the surrogate models for the objective function and for the constraint functions using information extracted from the database and the available surrogate models. Estimate the quality of the λ child solutions generated in Step 3 based on the updated surrogate models and prescreening methods.

Step 5: Function evaluation: Perform exact function evaluation on the estimated best candidate solution $\mathbf{x}^b$ from Step 4 and then add $\mathbf{x}^b$ and its exact function values to the database.

Since the working population P consists of the best solutions in the current database, the search concentrates on the current promising subregion, which is moving in the search space for exploration. This is necessary because the computational budget for exact function evaluations is very limited. In surrogate modeling, training data points that are close to the child solutions can be obtained so as to construct high-quality surrogate models. This will further be illustrated in Section 5.

4 Constraint Handling and Diversity Maintenance

This section explains and discusses the implementation of Step 1 and Step 2 of GPEEC for handling constraints.

4.1 Basic Constraint Handling Method

It is natural to use the information of constraint satisfaction to rank the candidate solutions. For simplicity and efficiency, a revised SF method proposed in [3] is adopted in Step 1 for selecting the λ best candidate solutions from the database. The ranking rules based on SF are as follows:

1. Feasible solutions rank higher than infeasible ones.
2. Feasible solutions are ranked solely based on their objective function values in ascending order.
3. Infeasible solutions are ranked solely based on their total constraint violation values ($\sum_{i=1}^{m} max\{0, g_i(\mathbf{x})\}$) in ascending order.

4.2 The Diversity Maintenance Method

Many constrained optimization procedures have the following phases [21]: (1) The population moves towards the feasible region and the main driving force is the minimization of the total amount of constraint violations. (2) A part of the population is in the feasible region and the other part is in the infeasible region, and the main driving forces are both the minimization of the total constraint violations and the optimization of the objective function. (3) Most candidate solutions of the population are in the feasible region and the main driving force is optimization of the objective function. An early stage and a late stage are used, which are separated by T, the number of feasible solutions generated so far, which should be set to several multiples of λ (the reason is explained in Section 7). This indicates that at the end of the early stage, most candidates are feasible while a substantial effort is used for objective function optimization; the late stage, on the other hand, mainly focuses on optimizing the objective function.

There are d decision variables x_i ($i = 1, \ldots, d$) in Problem (1). Let $\mathbf{x}^*$ be its globally optimal solution and let P^i contain the x_i values of all the solutions in the current population P. Ideally, each P^i will converge to the x_i value in $\mathbf{x}^*$. However, due to complex constraints and other reasons, some P^i may get trapped at some wrong position and thus lose its diversity at some search stages. If P^i is trapped at a value, x_i is called a trapped variable. Figure 1 provides an example with two variables, illustrating why the trapping of some P^i may happen. In this example, Ω_1 and Ω_2 are two parts of the feasible region. For each value of x_2, the feasible range

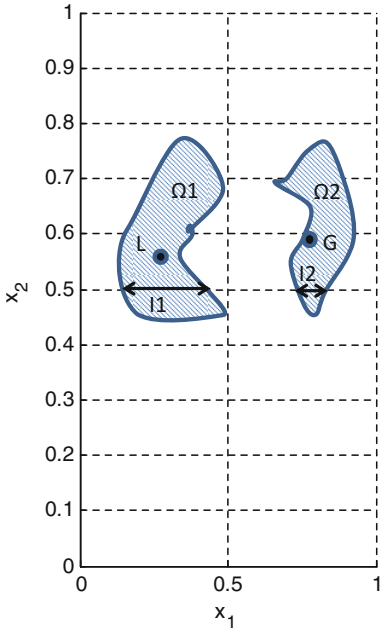

Fig. 1 An illustrative figure of the trapping of variables

of x_1 is different. In Figure 1, the feasible range of x_1 for each fixed x_2 value consists of two disconnected intervals since the feasible region is disconnected. When the early stage begins, the major driving force is the minimization of the total amount of constraint violations. When there are many constraints in Problem (1), it is very likely that the total amount of constraint violations drops significantly when some elements in P^1 enter the interval I_1 instead of I_2. Therefore, once some elements in P^1 fall in I_1, most elements in P^1 will enter I_1 very soon. Since I_2 is not connected with I_1, it is very difficult for x_1 to get out of I_1 just by reproduction operators such as crossovers and small mutations, and P^1 may lose its diversity and then get stuck at a value in I_1 because of the objective function optimization near the end of the early stage. To deal with this issue, the following method is used to improve the diversity.

The variables $x_1, \ldots, x_d$ in $\mathbf{x}$ are treated separately in the DM procedure.[1] For each x_k, its diversity in P, $D_k(P)$, is:

$$D_k(P) = max_{\tilde{\mathbf{x}}=(\tilde{x}_1,\ldots,\tilde{x}_d)\in TP}\{|\tilde{x}_k - x_k^{best}|\}$$

where TP contains the top ν solutions in P based on the SF ranking, $\mathbf{x}^{best} = (x_1^{best}, \ldots, x_d^{best})$ is the best solution in P, and ν is a control parameter.

[1] We assume that all the decision variables are at least related to one of the constraints; otherwise, they can be easily eliminated from the DM method.

The DM method (Step 2 in GPEEC in Section 3) works as follows:

If the total number of feasible solutions in the current database $\leq T$
FOR $k = 1, \ldots, d$
IF $D_k(P) \leq \varepsilon$.
For every solution $\widetilde{x} = (\widetilde{x}_1, \ldots, \widetilde{x}_d) \in P$, reset $\widetilde{x}_k$ to a uniformly distributed random number in $[a, b]$.
END
END

Several remarks can be made on the above method:

- The DM method is applied when the total number of feasible solutions generated so far is less than a predefined number T. In other words, this method will not be used at the late search stage. A major consideration is that the algorithm should focus on optimizing the objective function starting from a diversity maintained (if necessary) population at the late stage.
- When $D_k(P) \leq \varepsilon$, the x_k values of the top ν solutions in P are very close. It indicates that the current population P does not have a good diversity in x_k. Note that TP is the best subset of P and thus the current database. To prevent the search from being trapped in a locally optimal area, the value of x_k is randomly reset for each solution in P.
- At the early search stage, some $x_i(s)$ (often trapped variables) converge much faster than others (see definition and explanation of the trapped variables). Therefore, in the for loop of the DM method, not all the variables will be re-sampled from $[a, b]$. The newly generated solutions still inherit variables with good diversity in the current population. The re-sampling of trapped variables after the detection, in contrast to the random sampling in the beginning of the early stage, is effective for jumping out of the premature convergence. As said above, substantial effort must be spent on objective function optimization when the trapped variable can be detected (i.e., it converges to a very narrow range ε). Hence, many feasible solutions should have been generated and P should be around the feasible region, i.e., most constraints are satisfied. This implies that after re-sampling, each interval of a trapped variable has nearly equal chance to be selected considering the feasibility and the value of the objective function.
- The DM method can easily be used in the SMAS framework, since it improves the population diversity by only using information extracted from the decision space, rather than trading off the objective function value and the constraint satisfaction like most advanced constrained optimization methods.

The parameters in the DM method are set as follows: $T = 5 \times \lambda$, $\nu = 10$ and $\varepsilon = 0.1$ assuming a $[-10, 10]$ search region (we can make this assumption come true by scaling). More details are in Section 7.

5 Surrogate Modeling

This section discusses the implementation of Step 4 of GPEEC for the surrogate modeling and prescreening.

5.1 Surrogate Modeling Based on Improved SMAS

The idea of SMAS [20] is used in GPEEC. Different from other SAEA frameworks, SMAS concentrates both its search and its surrogate modeling on the current promising region and gradually moves the promising region for exploration. It uses the λ best candidate solutions to form its working population, and only evaluates the exact function value of the estimated best candidate solution.

Training data points with a higher quality can be generated using the SMAS framework than using an SAEA framework which relies on a standard EA. The solutions in P (Step 1) are not necessarily far away from one another since they are the current best candidate solutions. Also, because at most a single new solution enters P in each iteration, the selected estimated best solutions (which are generated from P in step 3 and which will serve as training data points) in several consecutive iterations will not be far away from one another. As a consequence, most training data points are also in or near the current promising region. Therefore, a high-quality surrogate model for this region can be constructed for prescreening newly generated solutions. That is why SMAS is a surrogate model-aware search mechanism. In contrast, new solutions often spread in different regions in standard EAs and thus often no sufficient number of training data points are around candidate solutions to be prescreened, affecting the surrogate model quality negatively.

To select training data points, the median of the λ new solutions for each decision variable is computed to construct a vector $\mathbf{m}_v$. The τ available training data points that are nearest to $\mathbf{m}_v$ are selected to construct the surrogate model. τ should be set between $5 \times d$ and $7 \times d$. $\tau = 6 \times d$ is used in all the experiments.

Note that the LCB prescreening is used only for the objective function, while for constraints the predicted value is used (i.e., $\omega = 0$). The reason is to prevent that many near-feasible solutions are selected, since in SMAS only a single candidate solution is selected and evaluated in each iteration.

5.2 The Adaptive Surrogate Model Updating Method

The ASU method in GPEEC adaptively decides whether the surrogate model will be updated or not, with the goal of reducing the computational overhead of surrogate modeling. This is important for GPEEC, because the computational overhead of surrogate model construction for constrained expensive optimization problems can be tremendous (see Section 1).

Since the goal is to minimize $f(\mathbf{x})$, the quality of the surrogate model for $f(\mathbf{x})$ does matter. For this reason, the surrogate model for $f(\mathbf{x})$ is updated at every iteration. As to the surrogate model for each $g_i(\mathbf{x})$, there are different considerations for the different search stages. In the early stage, the purpose of most surrogate models for $g_i(\mathbf{x})$ is to estimate $max\{0, g_i(\mathbf{x})\}$. Therefore, one needs high-quality models at this stage in order to estimate the total amount of constraint violations reliably. In contrast, when the search has almost entered the feasible region (the late stage), the surrogate models for $g_i(\mathbf{x})$ serve the purpose as long as the models can distinguish feasible solutions from infeasible ones. Therefore, it is not necessary to update the models of $g_i(\mathbf{x})$ at every iteration in the late stage. For those iterations where the surrogate model updating is not conducted, previous models are used for estimating the constraint function values. Therefore, the ASU method is applied when the total number of feasible solutions in the current database is larger than the parameter T which was introduced in Section 4.

At each iteration t at the late stage ($t \geq T$), the surrogate model is updated for $g_i(\mathbf{x})$ if

- $remaining(t, t_c) = 0$; or
- one of the η most recently evaluated points does not satisfy $g_i(\mathbf{x}) \leq 0$

where t_c and η are control parameters.

Several remarks can be made:

- The update of the surrogate models for constraint functions is conducted at every iteration in the early stage and after every t_c iterations in the late stage to reinforce the reliability of the surrogate models.
- At the late stage, since many candidate solutions in P are deep inside the feasible region, it is unlikely that all of the λ new solutions are infeasible. Thus, there is a high probability that some infeasible solution (considering $g_i(\mathbf{x})$) among the λ new solutions should be predicted as feasible and ranked as the best. In this case, it is possible that the search region is near the boundary of $g_i(\mathbf{x}) = 0$ or that the previous GP model cannot work. Hence, the surrogate model is updated for $g_i(\mathbf{x})$ in this case. For $j \in \{1, \ldots, m\}$ and $j \neq i$, the surrogate model of $g_j(\mathbf{x})$ is then not updated.

$t_c = 10$ and $\eta = 5$ are used in all the experiments. Their settings are discussed in Section 7.

6 Implementation Details

Some implementation details included in Step 1, Step 3 and Step 5 are as follows. Note that various alternative methods can be investigated and applied in the general GPEEC framework:

- In Step 1, the selected experimental design method is Latin hypercube sampling (LHS) [29]. LHS is widely used for the initial database generation in SAEA research. α (the number of initial samples) is often relatively small and the empirical rule for setting α for the SMAS framework is in [20], which is also applicable to GPEEC.
- The selected EA operators in Step 3 are the DE/best/1 mutation operator and the DE crossover operator. Section 2 has provided more details.
- In Step 5, sometimes a modified form of the estimated best candidate solution ($\mathbf{x}^b$) is used. The purpose is to avoid the repeated expensive evaluation of the same candidate solution and to perform local search. A Gaussian distributed random number with zero mean and with 5 % of the search range of each decision variable as variance is added to modify $\mathbf{x}^b$ if $\mathbf{x}^b$ has been evaluated before.

7 Experimental Studies

7.1 Test Problems and Parameter Settings

The GPEEC algorithm is tested with ten problems, which are shown in Table 1. First, 8 hard benchmark test problems for constrained optimization are used (G1 to G10 from the CEC 2006 special session on constrained real-parameter optimization [15], except G3 and G5 which use equality constraints: GN1–GN8 correspond to G1, G2, G4, G6, G7, G8, G9, G10 in [15], respectively), involving many constraints, disconnected feasible regions and active constraints. In addition, to test the ability of GPEEC on handling complex objective and constraint functions, a 20-dimensional test function is constructed with the Ackley and Griewank test functions [30] (please see the Appendix). Finally, a real-world problem from the mm-wave integrated circuit (IC) design field is used to demonstrate the capabilities of GPEEC.

Table 1 Test problems used in the experimental studies

Problem	Opt.	d	ρ (%)	a	m	λ	N_{eval}
GN1 (G1 in [15])	−15	13	0.011	6	9	40	1000
GN2 (G2 in [15])	−0.8036	20	99.99	1	2	40	2000
GN3 (G4 in [15])	−30665.54	5	52.12	2	6	30	800
GN4 (G6 in [15])	−6961.81	2	0.006	2	2	30	1000
GN5 (G7 in [15])	24.31	10	0.00	6	8	40	1000
GN6 (G8 in [15])	−0.0958	2	0.86	0	2	30	800
GN7 (G9 in [15])	680.63	7	0.52	2	4	30	800
GN8 (G10 in [15])	7049.33	8	0.00	3	6	40	1000
GN9	0	20	0.00	0	2	40	1500

Opt. is the globally optimal objective function value of each problem

In Table 1, ρ is an estimate of the ratio of the feasible space to the entire search space. a is the number of active constraints and m is the total number of constraints. All the parameter setting rules of GPEEC have been described above. For GN1–GN9, $\alpha = 50$ is used for problems with 10–20 variables, and $\alpha = 40$ for problems with less than ten variables. The population size λ and the number of exact function evaluations N_{eval} are shown in Table 1. Note that we assume that the calculations of $f(\mathbf{x})$ and the different $g_i(\mathbf{x})$ can be done in a single simulation or can be done in parallel, which is common to many expensive optimization problems nowadays. In all the experiments, DE/best/1 is used and the crossover rate CR is set to 0.8 according to the suggestions in [23]. Note that a relatively large scaling factor F (e.g., $F \in [0.75, 0.95]$) is necessary to promote exploration for the SMAS framework, in order to avoid getting stuck in local optima. F is set to 0.8 in the experiments. ω in LCB prescreening is set to 2 according to [5].

The experiments are carried out on a 2.66 GHz computer with 7.8 Gb RAM in the MATLAB environment on the Linux system. The ooDACE toolbox [8] is used for GP modeling.

7.2 The GPEEC Performance and Analysis

7.2.1 Reference Results

To evaluate the solution quality of GPEEC, a state-of-the-art constrained optimization method (without surrogate modeling) is used to provide a reference result. The method is the self-adaptive penalty function (SAPF) method from [31]. In [31], the real-coded genetic algorithm is used as the search engine. For fair comparison, the same DE as in GPEEC is used with $\lambda \times N_{eval}$ evaluations (the average is about 40,000 evaluations). Twenty runs are performed for each case. The results are shown in Table 2.

Table 2 Statistics of the best function values obtained by the first reference method SAPF for GN1–GN9 over 20 runs ($\lambda \times N_{eval}$ function evaluations)

Problem	Best	Worst	Mean	Median	Std	R_{inf}
GN1	−15	−12.01	−14.51	−15	1.06	0
GN2	−0.75	−0.53	−0.61	−0.60	0.06	0
GN3	−30665.53	−30664.04	−30665.07	−30665.03	0.16	0
GN4	−6961.22	−6898.73	−6935.50	−6937.11	18.56	0
GN5	25.37	29.57	26.60	26.02	1.49	0
GN6	−0.0958	−0.0958	−0.0958	−0.0958	1.9e−17	0
GN7	680.64	680.65	680.64	680.64	0.0034	0
GN8	8075.23	15419.02	9545.32	8681.38	2.30e+3	0
GN9	0.61	0.83	0.74	0.75	0.0761	0

R_{inf} refers to the percentage of runs providing an infeasible final result

Table 3 Statistics of the best function values obtained by the second reference method SF for GN1–GN9 over 20 runs ($\lambda \times N_{eval}$ function evaluations)

Problem	Best	Worst	Mean	Median	Std	R_{inf}
GN1	−15	−11	−13.44	−13	1.34	5 %
GN2	−0.77	−0.42	−0.62	−0.62	0.11	0
GN3	−30665.53	−30664.30	−30664.05	−30665.03	0.41	0
GN4	−6961.81	−6944.68	−6959.42	−6961.81	6.48	30 %
GN5	24.31	512.25	73.65	24.31	154.12	0
GN6	−0.0958	−0.0958	−0.0958	−0.0958	1.03e−17	0
GN7	680.63	680.63	680.63	680.63	5.73e−4	0
GN8	7050.92	20101.53	9880.37	7222.31	4295.13	55 %
GN9	0	0.020	0.011	0.015	0.0095	0

R_{inf} refers to the percentage of runs providing an infeasible final result.

Compared to [31], the results of SAPF for GN1, GN2, GN4, and GN8 are worse than the published results and the results of GN3, GN5, GN6, GN7 are better than the published results. This is because different search engines with different parameter settings are used inside SAPF.

Then, as a second reference, the SF method [3] is used with DE. Also, $\lambda \times N_{eval}$ evaluations are used. The results are shown in Table 3. Note that the results of some runs are infeasible, so the statistics only considers the runs that provide feasible final results.

It can be seen that some SF results for GN1, GN4, and GN8 are infeasible. For GN1 and GN5, it can be observed that premature convergence largely harms the performance in some runs. The SAPF method shows clear advantages on these problems. On the other hand, for GN7 and GN9, the SF method performs better than the SAPF method. This indicates that in some cases when the additional diversity enhancement does not help much, the SF method shows its advantage of fast convergence (being more efficient). Tables 2 and 3 will be used for comparison with GPEEC.

7.2.2 The Effect of the DM Method

The method to simulate the SMAS framework is shown in [20]. The same method is used here for the GPEEC framework. To simulate it, the GP modeling and prescreening are removed. Instead, exact function evaluations are conducted on all the λ child solutions in each iteration, and randomly select one from the top β solutions. $\beta = 5$ is used. By this simulation, the search (optimization) ability of the GPEEC framework can be analyzed and then it can be seen whether the surrogate modeling and prescreening/prediction work as expected or not. We first simulate the GPEEC framework without the DM method (without Step 2 in Section 3). Similar results are obtained compared to Table 3, the SF method. Then, the DM method is added to simulate the GPEEC framework. These results are in Table 4.

Table 4 Statistics of the best function values obtained by the simulated GPEEC framework for GN1–GN9 over 20 runs

Problem	Best	Worst	Mean	Median	Std	R_{inf}
GN1	−15	−13	−14.51	−14.92	0.71	0
GN2	−0.73	−0.50	−0.61	−0.63	0.09	0
GN3	−30665.53	−30664.17	−30664.81	−30665.04	0.42	0
GN4	−6961.81	−6928.37	−6957.16	−6961.74	10.41	0
GN5	24.47	25.08	24.73	24.62	0.25	0
GN6	−0.0958	−0.0958	−0.0958	−0.0958	1.02e−17	0
GN7	680.63	680.66	680.64	680.64	0.0095	0
GN8	7273.61	8331.37	7457.52	7335.19	318.99	0
GN9	0	0.096	0.026	0.017	0.028	0

R_{inf} refers to the percentage of runs providing an infeasible final result

Compared to Table 2 (the SAPF method), it can be seen that, except for GN1, GN2, GN3, and GN6, where the results are comparable to that of the SAPF method, the results are better than the SAPF method for the other five test problems. For GN5, GN8, and GN9 considerably better results are observed. Compared to Table 3 (the SF method), it can be seen that no infeasible solution has been provided by GPEEC. For GN1, GN4, GN5, and GN8, which are difficult to solve by the SF method, the DM method is seen to be a very effective solution method. Therefore, it can be concluded that the simulated GPEEC framework incorporates both a high constraint handling ability and a fast convergence, which is effective for problems with complex constraints. Moreover, experiments show that the DM method can be triggered more than once and to more than one variable for problems with complex constraints, while it often is not triggered for problems that can be solved well by the SF method.

7.2.3 Integrating the GP Modeling

GPEEC with surrogate modeling and prescreening but without the ASU method is then tested. N_{eval} evaluations are used (see Table 1). The results are in Table 5.

It can be seen from Table 5 that: (1) All the final solutions are feasible. (2) Except for GN2, the results are comparable to the results provided by the SAPF method. For GN2 with 20 variables, experiments have shown that GP modeling is not very fit for the properties of this function. For GN2 with fewer variables, on the other hand, the performance of GP modeling is good. It is no surprise that a single surrogate modeling method has difficulty to work well on all kinds of problems [13, 16], and the GPEEC method is compatible with hybrid surrogate models. (3) Thanks to the improved SMAS framework, the surrogate modeling works as expected. There is a little degradation compared to the simulation results in Table 4 for some problems, but GN5, GN7, and GN8 show better results than the simulation results. The advantages of using surrogate models in terms of solution quality have been

Table 5 Statistics of the best function values obtained by the GPEEC framework without the ASU method for GN1–GN9 over 20 runs (N_{eval} function evaluations)

Problem	Best	Worst	Mean	Median	Std	R_{inf}
GN1	−15.00	−12.93	−14.29	−14.85	0.94	0
GN2	−0.77	−0.38	−0.53	−0.53	0.12	0
GN3	−30665.53	−30664.01	−30664.33	−30664.21	0.43	0
GN4	−6956.70	−6769.34	−6902.68	−6915.19	58.81	0
GN5	24.31	25.01	24.39	24.32	0.22	0
GN6	−0.0958	−0.0932	−0.0954	−0.0958	8.66e−4	0
GN7	680.63	680.63	680.63	680.63	4.56e−4	0
GN8	7056.87	7583.43	7290.59	7247.08	192.75	0
GN9	1.42e−10	0.014	0.006	0.007	0.006	0

R_{inf} refers to the percentage of runs providing an infeasible final result

Table 6 Statistics of the best function values obtained by GPEEC for GN1–GN9 over 20 runs (N_{eval} function evaluations)

Problem	Best	Worst	Mean	Median	Std	R_{inf}	M/S
GN1	−14.99	−12.88	−14.36	−14.96	0.98	0	1610/17 %
GN2	−0.75	−0.40	−0.51	−0.48	0.12	0	1623/27 %
GN3	−30665.42	−30661.14	−30663.79	−30664.21	1.21	0	1869/36 %
GN4	−6961.54	−6764.85	−6886.90	−6898.43	75.31	0	0/0
GN5	24.31	25.03	24.60	24.32	0.36	0	4816/56 %
GN6	−0.0958	−0.0918	−0.0952	−0.0957	0.0012	0	666/29 %
GN7	680.63	680.63	680.63	680.63	9.27e−4	0	2216/58 %
GN8	7050.20	7310.41	7119.88	7097.64	89.20	0	3041/46 %
GN9	1.25e−9	0.72	0.08	2.39e−9	0.24	0	2038/47 %

R_{inf} refers to the percentage of runs providing an infeasible final result. M/S indicates the number of surrogate model constructions using the ASU method, and the percentage compared to updating every surrogate model in each iteration

discussed in [5, 16]. (4) Note that a fixed number of N_{eval} function evaluations are used and the convergence is earlier than that for most problems, which will be illustrated later on.

7.2.4 The GPEEC Performance and the Effect of the ASU Method

At last, when also integrating the ASU method, the full GPEEC algorithm is tested. N_{eval} function evaluations are used. The results are in Table 6.

In terms of optimality, it can be observed that the results are comparable to the results from Table 5, as well as with the SAPF results from Table 2. In terms of the necessary number of surrogate modeling, less than half or about half the surrogate modeling runs are used in most cases compared to not using ASU. For GN1 (13-dimensional, 9 constraints), GN2 (20-dimensional), and GN9 (20-dimensional),

when not using ASU, the surrogate modeling time often costs more than 15 h. When using the ASU method, the surrogate modeling only costs a few hours. The ASU method is especially useful when the exact evaluation is not very expensive but with many constraints, or when the number of decision variables is quite large.

7.2.5 Comparisons

The SF method is widely used in constrained optimization; its main advantages are its simplicity and efficiency [3]. To observe the speed enhancement of GPEEC, GPEEC (Table 6) is compared with SF (Table 3). Note that only the objective function values of the feasible runs of the SF method are used for comparison. In the experiments, N_{eval} evaluations are used for GPEEC, but most convergence happens earlier than that. To mark the convergence, a threshold, δ, is used, which means that after N_{ec} evaluations, the current best solution is feasible and the improvement to the objective function is less than δ after that. Thus, it can be considered that GPEEC converges at N_{ec} evaluations. Because the objective function values for different test problems are in different scales (e.g., GN6 between -0.1 and 0, GN8 about 10^4), the selected δ are shown in Table 7.

To make the comparisons, the following information for GPEEC and SF are reported.

- $G_{N_{ec}}$: the median of the best function values obtained using N_{ec} exact function evaluations by GPEEC;
- $S_{N_{ec}}$: the median of the best function values obtained using N_{ec} exact function evaluations by SF;
- $H_{N_{ec}}$: the number of exact function evaluations needed for SF to achieve $G_{N_{ec}}$. If the final results of the SF method after $\lambda \times N_{eval}$ evaluations are worse than the GPEEC result with N_{ec} exact function evaluations, we denote this as N.A.

The comparison results are shown in Table 8. It can be seen that 6 out of 9 of the problems, tens to even more than a hundred times less exact function evaluations are needed by GPEEC.

Table 7 δ for GN1–GN9

Problem	GN1	GN2	GN3	GN4	GN5
δ	0.1	0.01	1	1	0.1
Problem	GN6	GN7	GN8	GN9	
δ	0.001	0.1	1	0.1	

Table 8 Comparisons between GPEEC and SF for GN1–GN9 over 20 runs

Problem	N_{ec}	$G_{N_{ec}}$	$S_{N_{ec}}$	$H_{N_{ec}}$	Speedup
GN1	268	−14.9	−7.0	N.A.	>150
GN2	1959	−0.47	−0.33	4880	2.5
GN3	209	−30663.7	−29961.1	840	4
GN4	134	−6898.4	−4520.9	1280	10
GN5	920	24.3	1008.4	24640	27
GN6	535	−0.0957	−0.0916	840	1.6
GN7	523	680.4	1738.5	8040	15
GN8	992	7097.6	14615.3	N.A.	>40
GN9	724	0.094	148.29	12800	18

7.2.6 Discussions on Parameter Setting

The parameters of GPEEC can be classified into three categories: (1) parameters for the improved SMAS framework, (2) parameters for the DM method, and (3) parameters for the ASU method.

The parameter setting for the SMAS framework has followed the practice in [17, 20] where the setting has been discussed in detail.

As to parameter setting in the DM method, ε is the threshold for $D_k(P)$ to measure the diversity of the current population on x_k. A small number is needed depending on the range of each decision variable. $\varepsilon = 0.1$ is used assuming a $[-10, 10]$ search region (we can make this assumption come true by scaling). T is used to define the division of the two search stages. In the late stage, the search is conducted mainly in the feasible region and a substantial effort is made to optimize the objective function. Although every solution in the population P will be feasible after λ feasible solutions have been generated due to the ranking rules used in Section 4.1, experiments show that many child solutions produced from P are still infeasible until after $3 \times \lambda$ to $4 \times \lambda$ feasible solutions have been generated. Based on this observation, T is set to be $5 \times \lambda$. ν is used to determine whether a variable is trapped or not. A big ν value may not be necessary. It is suggested to be from 5 to 10. In the following, GN5 and GN7 are used as examples to test the impact of T and ν. It can be seen from Table 9 that if ν and T are selected in the suggested ranges, the DM method performs well and is robust.

As to parameter setting in the ASU method, note that this method is only used when the number of generated feasible solutions is larger than T and the surrogate models for constraint functions are used to differentiate feasible solutions and infeasible ones. η is used to judge if the search is near the boundary of the feasible region. A small η value may lead to a wrong judgement, while a large η value may cause unnecessary model updating and thus may waste computational efforts. It is suggested to be set around 5. t_c defines how often the surrogate model for each constraint function must be updated regularly. Considering the help of η and the

Table 9 Statistics of the best function values obtained by the GPEEC framework without the ASU method for GN5 and GN7 over 20 runs (N_{eval} function evaluations) with different DM parameters

Problem	Best	Worst	Mean	Median	Std	R_{inf}
GN5, $\nu = 5, T = 5 \times \lambda$	24.31	25.21	24.61	24.32	0.35	0
GN5, $\nu = 10, T = 4 \times \lambda$	24.31	25.05	24.57	24.34	0.34	0
GN5, $\nu = 10, T = 6 \times \lambda$	24.31	25.01	24.47	24.32	0.30	0
GN7, $\nu = 5, T = 5 \times \lambda$	680.63	680.63	680.63	680.63	0.0014	0
GN7, $\nu = 10, T = 4 \times \lambda$	680.63	680.63	680.63	680.63	2.7e−4	0
GN7, $\nu = 10, T = 6 \times \lambda$	680.63	680.63	680.63	680.63	4.4e−4	0

R_{inf} refers to the percentage of runs providing an infeasible final result

Table 10 Statistics of the best function values obtained by GPEEC for GN5 and GN7 over 20 runs (N_{eval} function evaluations) with different ASU parameters

Problem	Best	Worst	Mean	Median	Std	R_{inf}
GN5, $t_c = 20, W = 5$	24.31	25.07	24.45	24.32	0.28	0
GN5, $t_c = 10, W = 3$	24.31	25.01	24.59	24.36	0.35	0
GN5, $t_c = 10, W = 7$	24.31	25.01	24.47	24.33	0.27	0
GN7, $t_c = 20, W = 5$	680.63	680.63	680.63	680.63	5.3e−4	0
GN7, $t_c = 10, W = 3$	680.63	680.63	680.63	680.63	8.8e−4	0
GN7, $t_c = 10, W = 7$	680.63	680.64	680.63	680.63	0.0027	0

R_{inf} refers to the percentage of runs providing an infeasible final result

main goal of the late stage, the regular updating does not need to be very frequent. t_c is suggested to be set between 10 and 20. In the following, GN5 and GN7 are used as examples to test if the suggested setting is robust. It can be observed from Table 10 that GPEEC performs well when η and t_c are not very far from the suggested values. When using η in the suggested range, t_c is not sensitive.

7.3 mm-Wave IC Design Optimization Example

This section provides a real-world engineering application of GPEEC: the design optimization of a 60 GHz power amplifier in a 65 nm CMOS technology. At mm-wave frequencies, the simple equivalent circuit models typically used for passive components at low frequencies are no longer accurate, and the way left to the designers is "trial and error." Therefore, the global optimization of mm-wave ICs is very important. However, electromagnetic (EM) simulation is needed in the evaluation of candidate designs, which is computationally expensive. In power amplifier design, the 1 dB compression point ($P_{1\,\mathrm{dB}}$), the power added efficiency ($PAE@P_{1\,\mathrm{dB}}$) and the power gain (G_p) are key performances. In practical design, the goal is often to maximize $P_{1\,\mathrm{dB}}$ or $PAE@P_{1\,\mathrm{dB}}$, with constraints on the other two specifications.

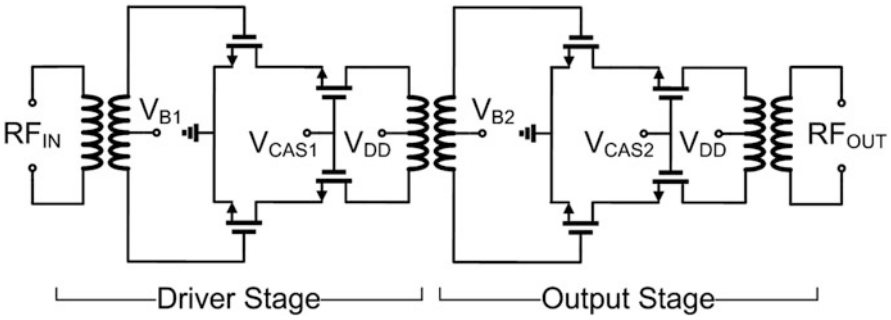

Fig. 2 Schematic of the 60 GHz power amplifier [33]

The problem is defined in (11). The circuit configuration is shown in Figure 2. The design parameters (in total 18) include the inner diameters and metal width of the primary and secondary inductors of every transformer (in total three transformers; each transformer has two inductors), 5 biasing voltages and the number of fingers of the driver stage. The inner diameter has a range from 20 to 100 μm. The metal width has a range from 3 to 10μm. The 5 biasing voltages have ranges from 0.5 to 2 V. The number of fingers can be 2/3/4. This is a simulation-based (black-box) optimization problem, so no explicit analytical expression is available. A Xeon 2.66 GHz computer is used for the synthesis (design optimization). GPEEC is programmed in MATLAB and the simulation is carried out in Cadence and ADS-Momentum (IC and electromagnetic simulation software). All the programs are run on the Linux system. The evaluation of a candidate design of this power amplifier needs 10–13 min using the simulation software to obtain the values of $P_{1\,\mathrm{dB}}$, $PAE@P_{1\,\mathrm{dB}}$, and G_p. The total computational time is restricted to about 2 days to reach the practical requirement of a design automation software tool acceptable in industry.

$$\begin{aligned} \text{maximize} \quad & PAE@P_{1\,\mathrm{dB}} \\ \text{subject to} \quad & P_{1\,\mathrm{dB}} \geq 13\,\mathrm{dBm} \\ & G_p \geq 10\,\mathrm{dB} \end{aligned} \tag{11}$$

The initial number of samples α is set to 70. All the other settings are the same as those used in the benchmark problem tests. After 200 exact function evaluations, GPEEC gets the optimized result: $P_{1\,\mathrm{dB}}$ is 14.34 dBm, $PAE@P_{1\,\mathrm{dB}}$ is 9.52 % and G_p is 10.47 dB. The time cost is 41.6 h (wall clock time).

The high quality of this optimized result by GPEEC can be verified by comparing to a manual design [33] using the same circuit structure in the 65 nm technology. The reference result is as follows: $P_{1\,\mathrm{dB}}$ is 10.8 dBm, $PAE@P_{1\,\mathrm{dB}}$ is 4.5 % and G_p is 10.2 dB. It can be seen that the result of GPEEC fully dominates the experience-based manual design result on all the performances. To verify the efficiency of GPEEC, the SF method with the same DE optimizer is used and the optimization

time is set to 10 days. The result is $P_{1\,\mathrm{dB}} = 9.44\,\mathrm{dBm}$, $PAE@P_{1\,\mathrm{dB}} = 7.95\,\%$, and $G_p = 12.60\,\mathrm{dB}$. It can be seen that the $P_{1\,\mathrm{dB}}$ constraint is not satisfied and the *PAE* value is much worse than that obtained by GPEEC.

8 Conclusions

This chapter has presented the GPEEC algorithm for dealing efficiently with computationally expensive inequality constrained optimization problems, which is of great importance for the industry. GPEEC has the ability to handle complex constraints in an efficient manner. Thanks to the improved SMAS framework and the ranking method, an efficient SAEA for constrained expensive optimization has been constructed. Thanks to the DM method, complex constraints can be handled effectively under the improved SMAS framework. The ASU method saves more than half the computational effort on surrogate modeling for most test problems compared to updating the surrogate models in each iteration, which is especially useful for problems with several tens of variables or/and with many constraints. In addition, although the ideas behind the key components of GPEEC are not easy, their implementation is straightforward, showing GPEEC potential usage in industrial applications. Experimental studies on a set of widely used test problems have shown that comparable results in terms of optimality can be obtained when compared to a SAPF method (without surrogate model), and that several tens to more than one hundred times less exact function evaluations are needed compared to an efficient SF method. GPEEC is also applied to a mm-wave IC design optimization problem and have obtained a high-performance result with an affordable amount of computational effort.

Appendix

Benchmark test problems:

GN9

$$\begin{aligned}
&\text{minimize } f(\mathbf{x}) = 1 + \textstyle\sum_{i=1}^{d} \frac{(100*x_i)^2}{4000} - \prod_{i=1}^{d} cos(\frac{(100*x_i)}{\sqrt{i}}) \\
&\text{subject to } g_1(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)} \\
&\qquad -5 \le 0 \\
&\qquad g_2(\mathbf{x}) = -\textstyle\sum_{i=1}^{20} x_i - 10 \le 0 \\
&\qquad x_i \in [-6, 6], i = 1, \ldots, 20 \\
&\qquad \mathit{minimum} : f(\mathbf{x}^*) = 0
\end{aligned} \tag{12}$$

References

1. Alexandrov, N.M., Lewis, R.M., Gumbert, C.R., Green, L.L., Newman, P.A.: Approximation and model management in aerodynamic optimization with variable-fidelity models. J. Aircr. **38**(6), 1093–1101 (2001)
2. Basudhar, A., Dribusch, C., Lacaze, S., Missoum, S.: Constrained efficient global optimization with support vector machines. Struct. Multidiscip. Optim. **46**, 1–21 (2012)
3. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2), 311–338 (2000)
4. Dennis, J., Torczon, V.: Managing approximation models in optimization. In: Multidisciplinary Design Optimization: State-of-the-Art, SIAM pp. 330–347 (1997)
5. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. IEEE Trans. Evol. Comput. **10**(4), 421–439 (2006)
6. Forrester, A., Sóbester, A., Keane, A.: Engineering Design via Surrogate Modelling: A Practical Guide. Wiley, New York (2008)
7. Goh, C., Lim, D., Ma, L., Ong, Y., Dutta, P.: A surrogate-assisted memetic co-evolutionary algorithm for expensive constrained optimization problems. In: 2011 IEEE Congress on Evolutionary Computation (CEC), pp. 744–749. IEEE, New York (2011)
8. Gorissen, D., Couckuyt, I., Demeester, P., Dhaene, T., Crombecq, K.: A surrogate modeling and adaptive sampling toolbox for computer based design. J. Mach. Learn. Res. **11**, 2051–2055 (2010)
9. Jones, D.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Optim. **21**(4), 345–383 (2001)
10. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)
11. Kleijnen, J.P., Beers, W.V., Nieuwenhuyse, I.V.: Constrained optimization in expensive simulation: Novel approach. Eur. J. Oper. Res. **202**(1), 164–174 (2010)
12. Koziel, S., Leifsson, L.: Surrogate-Based Modeling and Optimization. Applications in Engineering, Springer (2013)
13. Le, M.N., Ong, Y.S., Menzel, S., Jin, Y., Sendhoff, B.: Evolution by adapting surrogates. Evol. Comput. **21**(2), 313–340 (2013)
14. Lee, C., Ahn, J., Bae, H., Kwon, J.: Efficient global optimization incorporating feasibility criterion for the design of a supersonic inlet. Proc. Inst. Mech. Eng. G J. Aerosp. Eng. **226**(11), 1362–1372 (2011)
15. Liang, J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C.A.C., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization, IEEE (2006)
16. Lim, D., Jin, Y., Ong, Y., Sendhoff, B.: Generalizing surrogate-assisted evolutionary computation. IEEE Trans. Evol. Comput. **14**(3), 329–355 (2010)
17. Liu, B., Aliakbarian, H., Ma, Z., Vandenbosch, G., Gielen, G., Excell, P.: An efficient method for antenna design optimization based on evolutionary computation and machine learning techniques. IEEE Trans. Antennas Propag. **62**(1), 7–18 (2014)
18. Liu, B., Deferm, N., Zhao, D., Reynaert, P., Gielen, G.: An efficient high-frequency linear RF amplifier synthesis method based on evolutionary computation and machine learning techniques. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **31**(7), 981–993 (2012)
19. Liu, B., Gielen, G., Fernández, F.: Automated design of analog and high-frequency circuits. Stud. Comput. Intell. **501** (2014)
20. Liu, B., Zhang, Q., Gielen, G.: A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Trans. Evol. Comput. **18**(2), 180–192 (2014)
21. Mallipeddi, R., Suganthan, P.: Ensemble of constraint handling techniques. IEEE Trans. Evol. Comput. **14**(4), 561–579 (2010)

22. Ong, Y., Lum, K.Y., Nair, P.: Hybrid evolutionary algorithm with hermite radial basis function interpolants for computationally expensive adjoint solvers. Comput. Optim. Appl. **39**(1), 97–119 (2008)
23. Price, K., Storn, R., Lampinen, J.: Differential evolution: a practical approach to global optimization. Springer, New York (2005)
24. Rasmussen, C.: Gaussian Processes in Machine Learning. Advanced Lectures on Machine Learning, Springer pp. 63–71 (2004)
25. Runarsson, T., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Comput. **4**(3), 284–294 (2000)
26. Runarsson, T.P.: Constrained evolutionary optimization by approximate ranking and surrogate models. In: Parallel Problem Solving from Nature-PPSN VIII, pp. 401–410. Springer, Berlin (2004)
27. Sacks, J., Welch, W., Mitchell, T., Wynn, H.: Design and analysis of computer experiments. Stat. Sci. **4**(4), 409–423 (1989)
28. Sasena, M.J., Papalambros, P., Goovaerts, P.: Exploration of metamodeling sampling criteria for constrained global optimization. Eng. Optim. **34**(3), 263–278 (2002)
29. Stein, M.: Large sample properties of simulations using Latin hypercube sampling. Technometrics **29**, 143–151 (1987)
30. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technological University, Singapore, Technical Report **2005005** (2005)
31. Tessema, B., Yen, G.: A self adaptive penalty function based algorithm for constrained optimization. In: IEEE Congress on Evolutionary Computation (CEC 2006), pp. 246–253. IEEE, New York (2006)
32. Venkatraman, S., Yen, G.: A generic framework for constrained optimization using genetic algorithms. IEEE Trans. Evol. Comput. **9**(4), 424–435 (2005)
33. Zhao, D., He, Y., Li, L., Joos, D., Philibert, W., Reynaert, P.: A 60 GHz 14 dBm power amplifier with a transformer-based power combiner in 65 nm CMOS. Int. J. Microwave Wireless Technolog. **3**(02), 99–105 (2011)
34. Zhou, Z., Ong, Y., Nair, P., Keane, A., Lum, K.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Trans. Syst. Man Cybern. C Appl. Rev. **37**(1), 66–76 (2007)