Journal Article

A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Black Box Optimization Problems

Liu, B., Q. Zhang, Q. and Gielen, G.

A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems

Bo Liu, Qingfu Zhang, *Senior Member, IEEE*, and Georges Gielen, *Fellow, IEEE*

***Abstract*—Surrogate model assisted evolutionary algorithms (SAEAs) have recently attracted much attention due to the growing need for computationally expensive optimization in many real-world applications. Most current SAEAs, however, focus on small-scale problems. SAEAs for medium-scale problems (i.e., 20-50 decision variables) have not yet been well studied. In this paper, a Gaussian process surrogate model assisted evolutionary algorithm for medium-scale computationally expensive optimization problems (GPEME) is proposed and investigated. Its major components are a surrogate model-aware search mechanism for expensive optimization problems when a high-quality surrogate model is difficult to build, and dimension reduction techniques for tackling the "curse of dimensionality". A new framework is developed and used in GPEME, which carefully coordinates the surrogate modeling and the evolutionary search, so that the search can focus on a small promising area and is supported by the constructed surrogate model. Sammon mapping is introduced to transform the decision variables from tens of dimensions to a few dimensions, in order to take advantage of Gaussian process surrogate modeling in a low-dimensional space. Empirical studies on benchmark problems with 20, 30 and 50 variables and a real-world power amplifier design automation problem with 17 variables show the high efficiency and effectiveness of GPEME. Compared to three state-of-the-art SAEAs, better or similar solutions can be obtained with 12% to 50% exact function evaluations.**

***Index Terms*—surrogate model assisted evolutionary computation, surrogate models, expensive optimization, Gaussian process, prescreening, space mapping, dimension reduction.**

I. Introduction

Many real-world optimization problems require expensive computer or physical simulation for evaluating their candidate solutions [1], [2]. Often, traditional gradient-based mathematical programming methods cannot be applied directly to these problems since analytic formulations are unavailable. Evolutionary algorithms (EA) cannot directly solve them either since a large number of function evaluations are unaffordable. Surrogate model assisted evolutionary algorithms (SAEAs) are recent promising approaches for dealing with such expensive optimization problems. SAEAs use surrogate models to replace computationally expensive real function evaluations. Since surrogate modeling and prediction/prescreening use much less computational overhead than directly using the computationally expensive exact function evaluator, the computational cost can be reduced significantly.

This paper focuses on expensive optimization problems of 20 to 50 decision variables (medium-scale) and we assume that the computational budget available is in the range of 1,000 real function evaluations. Such problems can be found in many real-world applications such as mm-wave integrated circuit design [3], antenna design [4], mechanical engineering [5] and manufacturing engineering [6]. For example, a single simulation of a high-frequency integrated circuit needs about 10 to 15 minutes, and a typical THz computational electromagnetic simulation may cost 20-30 minutes. Due to the very tight time-to-market requirements, the optimization task should be completed within one to two weeks. Thus, one has to deliver the final solution within 1,000 function evaluations. Many of these computationally very expensive problems have around 20 to 50 design variables [3], [4], [6].

Most current SAEAs focus on small-scale expensive optimization problems [7], [8], [9]. For problems with 20 or 30 variables, some successful attempts include [1], [10] and [11]. [1] uses a weighted sum of the predicted values of different surrogate models to assist an EA, and the weights are adaptively adjusted based on the prediction uncertainty of different surrogate models. Reasonably good results on benchmark problems with 30 decision variables have been obtained, but it cost 8,000 exact function evaluations. [11] uses the Gaussian process (GP) model with probability of improvement prescreening [12] as a global surrogate model and Lamarckian evolution (using the radial basis function) as a local surrogate model to accelerate an EA. Good results on benchmark problems with 20 variables can be obtained with 6,000 exact function evaluations. [10] investigates GP-based local surrogate models and different prescreening methods to assist $(\mu + \lambda)$ evolution strategies. Benchmark problems with 20 variables have been tested, and promising results have been obtained on some problems with only 1,000 exact function evaluations. However, its solution quality (especially for multimodal problems) needs to be improved. We believe that much effort should be made to develop efficient and effective SAEAs for medium-scale computationally expensive optimization problems to meet the increasing industry requirements.

Many surrogate models have been introduced to assist EAs. Among them, the GP modeling, response surface methods, artificial neural networks, support vector machines, and radial

The research of B. Liu was supported by the Humboldt research fellowship, Germany and a scholarship of Katholieke Universiteit Leuven, Belgium.

B. Liu is with Department of Computing, Glyndwr University, Wrexham, U.K. (e-mail: b.liu@glyndwr.ac.uk, liu_bo765@yahoo.com.cn).

Q. Zhang is with the School of Computer Science and Electronic Engineering, University of Essex, CO4 3SQ, U.K. (e-mail: qzhang@essex.ac.uk).

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

basis functions exhibit good performances and are widely used [7], [8], [12], [10], [13], [14], [15], [16]. The GP modeling with the lower confidence bound (LCB) [17] prescreening is used in this paper, mainly based on the following considerations:

- GP modeling is a theoretically sound method for determining a much smaller number of free model parameters than many other surrogate modeling approaches [18], [19].
- GP modeling can provide an estimate of the model uncertainty to each predicted point, which can be interpreted in a very natural way [18], [19].
- Several prescreening methods are available for the GP modeling in optimization. For example, the expected improvement prescreening [7] has demonstrated its ability to escape a locally optimal area [10].

A key issue in an SAEA is how to use a reasonable amount of computational effort to build a good model for locating the most promising candidate solutions. GP modeling can involve heavy computational overheads. The computational complexity of a typical learning algorithm for GP modeling is $O(N_{it}K^3d)$ [10], where N_{it} is the number of iterations, d is the number of variables, and K is the number of training data points. When d is large, a large N_{it} is often required to obtain a good model. Our pilot experiments have shown that when using 150 training data, it can take 240-400 seconds to build a single GP model for an objective function with 50 variables when the Blind DACE toolbox [20] is used on a Xeon 2.66GHz computer with the MATLAB environment. The cost will increase cubically as K increases. An SAEA with a budget of 1,000 function evaluations, as in this paper, may need to do GP modeling for several hundred times. Although starting from the GP model from the previous iteration can reduce the computational overhead to some extent, it is still not realistic to use too many training points in the GP modeling. On the other hand, using too few training data may deteriorate the reliability of the model and thus the solution quality. Besides the number of training data points, another important factor affecting the quality of a surrogate model is the location of the training data points. A standard EA, used as the search engine in most SAEAs (e.g. [1], [8], [10], [11]), generates new points mainly for optimization purpose but not for modeling. When an SAEA with such an EA prescreens a candidate solution, the model could be poor to judge the quality of this solution properly, since it is very likely that not enough training data points used for modeling are close to this solution, particularly in a high dimensional space.

This paper proposes two techniques to address the above issue. One is to employ a dimension reduction technique to map the training data to a lower-dimensional space on which the GP modeling will be conducted. In this way, the quality of the model can be largely improved due to the reduced space and the computational cost of the modeling can be reduced significantly. The other technique is to focus the search on a promising subregion, which is achieved by a new surrogate model-aware search mechanism. A new SAEA method is then proposed that uses these two techniques, called Gaussian process surrogate model assisted evolutionary algorithm for medium-scale computationally expensive optimization problems (GPEME). Experimental results on benchmark problems show that GPEME, with 12% to 50% exact function evaluations, outperforms or performs similarly to some other state-of-the-art SAEAs [1], [10], [11]. We also report the result of GPEME for an mm-wave integrated circuit design optimization problem.

The remainder of this paper is organized as follows. Section II introduces the surrogate modeling technique used in our proposed algorithm. A new SAEA framework and GPEME are then presented in Section III. Section IV presents the experimental results of GPEME on some commonly used benchmark problems and a real-world engineering problem. Comparisons with some state-of-the-art methods are also provided in this section. Concluding remarks are presented in Section V.

II. The Surrogate Modeling in GPEME

A. Gaussian Process Modeling

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

$$\epsilon(x, x') = \exp\left(\sum_{i=1}^{d} \theta_i |x_i - x'_i|^{p_i}\right), \tag{1}$$

where parameter $1 \le p_i \le 2$ is related to the smoothness of $f(x)$ with respect to x_i, and parameter $\theta_i > 0$ indicates the importance of x_i on $f(x)$. More details about GP modeling can be found in [21].

1) Hyper Parameter Estimation: Given K points $x^1, \ldots, x^K \in R^d$ 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 $f(x) = y^i$ at $x = x^i$ $(i = 1, \ldots, K)$ [7]:

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

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

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

$$\hat{\mu} = \frac{\mathbf{1}^T C^{-1} y}{\mathbf{1}^T C^{-1} \mathbf{1}} \tag{3}$$

and

$$\hat{\sigma}^2 = \frac{(y - \mathbf{1}\hat{\mu})^T C^{-1}(y - \mathbf{1}\hat{\mu})}{K}. \tag{4}$$

Substituting (3) and (4) into (2) eliminates the unknown parameters μ and σ from (2). As a result, the likelihood function depends only on θ_i and p_i for $i = 1, \ldots, d$. (2) can then be maximized to obtain estimates of θ_i and p_i. The estimates $\hat{\mu}$ and $\hat{\sigma}^2$ can then readily be obtained from (3) and (4). In our experiments, we use the MATLAB optimization toolbox to optimize the likelihood function.

2) The Best Linear Unbiased Prediction and Predictive Distribution: Given the hyper parameter estimates $\hat{\theta}_i$, $\hat{p}_i$, $\hat{\mu}$ and $\hat{\sigma}^2$, one can predict $y = f(x)$ at any untested point x based on the f-function values y^i at x^i for $i = 1,\ldots,K$. The best linear unbiased predictor of $f(x)$ is [7], [22]:

$$\hat{f}(x) = \hat{\mu} + r^T C^{-1}(y - 1\hat{\mu}) \tag{5}$$

and its mean squared error is:

$$s^2(x) = \hat{\sigma}^2[1 - r^T C^{-1} r + \frac{(1 - 1^T C^{-1} r)^2}{1^T C^{-1} r}] \tag{6}$$

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

3) Lower Confidence Bound: We consider minimization of $f(x)$ in this paper. Given the predictive distribution $N(\hat{f}(x), s^2(x))$ for $f(x)$, the LCB of $f(x)$ can be defined as [17]:

$$f_{lcb}(x) = \hat{f}(x) - \omega s(x) \tag{7}$$

where ω is a constant. In our algorithm, we use $f_{lcb}(x)$ instead of $\hat{f}(x)$ itself to measure the quality of x. The use of LCB can balance the search between promising areas (i.e., with low $\hat{f}(x)$ values) and less explored areas (i.e., with high $s(x)$ values).

B. Dimension Reduction

A key issue in the design of an SAEA can be stated as follows:

- Given K points $x^1,\ldots,x^K \in R^d$ and their f-function values $y^1,\ldots,y^K$. Let $x^{K+1},\ldots,x^{K+M}$ be M untested points in R^d. How can one rank the M untested points without exact function evaluation?

A natural solution to this issue is that one first builds a GP model based on the exact evaluated y^i data, computes the LCB values of each untested point and then ranks them based on their LCB values. This procedure is so called prescreening. In GPEME, we only select the point with the best LCB value for exact function evaluation based on the obtained ranking. It is desirable that the selected point is a high-ranked one among all the untested points. Our pilot experiments have shown that for the test problems with 50 variables used in this paper, it is often very hard to achieve the above goal with 100 or 150 training data. Using a larger K, as argued in Section I, can lead to a large amount of computational overhead in GP modeling.

To deal with problems with around 50 decision variables, we propose an efficient way to do prescreening with dimension reduction. The key idea is to map the training data to a lower-dimensional space and then do GP modeling. As a result, the disadvantages caused by the insufficient number of training data points for medium-scale problems can be overcome. The method works as follows:

Prescreening by GP with dimension reduction (GP+DR)

Input: (1) Training data: $x^1,\ldots,x^K$ and $y^1,\ldots,y^K$. (2) Points to prescreen: $x^{K+1},\ldots,x^{K+M}$.

Output: The estimated best solution among all the untested points $x^{K+1},\ldots,x^{K+M}$.

Step 1: Map $x^1,\ldots,x^{K+M} \in R^d$ into R^l, where $l < d$, and obtain their corresponding images in R^l : $\bar{x}^1,\ldots,\bar{x}^{K+M}$.

Step 2: Build a GP model by using the $\bar{x}^i$ and y^i data ($i = 1,\ldots,K$), and then use the model to compute the LCB value of each $\bar{x}^{K+j}$ ($j = 1,\ldots,M$).

Step 3: Find the point with the smallest LCB value among $\bar{x}^{K+1},\ldots,\bar{x}^{K+M}$. Output its corresponding point in the original decision space R^d as the estimated best solution.

There are many machine learning methods which can transform the original space R^d to the latent space R^l for dimension reduction [23]. We have the following two considerations:

- The neighborhood relation among the data points plays a critical role in the correlation function in GP modeling. Therefore, the neighborhood relationship among the $\bar{x}^i$ points in R^l should be as similar as possible to those among all the x^i points ($i = 1,\ldots,K+M$) in R^d. The pairwise distances among the data points should be preserved as much as possible.
- The mapping technique should not be very costly since it will be used many times in an SAEA.

Based on these considerations, we use the Sammon mapping [24] which minimizes the differences between corresponding inter-point distances in the two spaces. Moreover, it is not expensive when $K+M$ and l are not large. Sammon mapping minimizes the following error function:

$$E = \sum_{1\le i<j\le K+M} \frac{1}{dis(x^i,x^j)} \times \sum_{1\le i<j\le K+M} \frac{[dis(x^i,x^j) - dis(\bar{x}^i,\bar{x}^j)]^2}{dis(x^i,x^j)} \tag{8}$$

where $dis(*,*)$ is the Euclidean distance.

Minimization of the error function in (8) is a large-scale quadratic optimization problem. In our experiments, we use the gradient descent method in [25] to solve it and we set $l = 4$ (more details can be found in Section IV). We use the principle component analysis (PCA) technique to generate an initial point for the optimization process. When $K = 100$, $M = 50$ and $d = 50$, as used in our experiments in this paper, the CPU time consumed by Sammon mapping is a few seconds on a 2.66GHz computer.

C. GP Modeling with Dimension Reduction using Sammon Mapping vs Direct GP Modeling

In the following, we discuss the major advantages and disadvantages of these two approaches.

GP modeling with dimension reduction (GP + DR):

- Advantages:
  - Sammon mapping transforms the training data points from the original space R^d to a lower-dimensional space R^l. Therefore, the number of training data points required for a good GP model can be reduced.
  - Since the modeling is conducted in a lower-dimensional space, the computational overhead can also be reduced significantly.

- Disadvantage:
  - Although Sammon mapping minimizes the error function E between the two spaces, the error cannot be zero. This means that some neighborhood information of the training data points in the original space will be lost in the latent space.

Direct GP modeling in the original space R^d:

- Advantage:
  - Errors caused by dimension reduction can be avoided.
- Disadvantages:
  - In the case of large d (such as d 50), it is often very difficult to select a sufficient number of proper training data points from the available data set to build a good GP model for a particular search area of interest.
  - When d is large, direct modeling will be costly as pointed out in Section I.

For these reasons, we decide to use the direct GP modeling in our proposed algorithm when d 20 and 30, and GP+DR in the case of d 50. One thing that needs to be mentioned is that for problems with 20 or 30 variables, both methods can obtain reasonably good results (better than the available methods). This is related to our new SAEA framework, which is another main contributor to GPEME and which will be described in the next section.

III. The Proposed GPEME Algorithm

A. Differential Evolution (DE)

The DE algorithm is used as the search engine in our proposed GPEME algorithm. DE is an effective and popular global optimization algorithm. It uses a differential operator to create new candidate solutions [26]. There are quite a few different DE variants. In this paper, we use DE/best/1 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 x^{best}. Let x $(x_1, \ldots, x_d) \in R^d$ be an individual solution in P. To generate a child solution u $(u_1, \ldots, u_d)$ for x, DE/best/1 works as follows.

A donor vector is first produced by mutation:

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

where x^{r_1} and x^{r_2} are two different solutions randomly selected from P and are also different from x^{best}. $F \in (0, 2]$ is a control parameter, often called the scaling factor [26]. Then the following crossover operator is applied to produce 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:

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

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

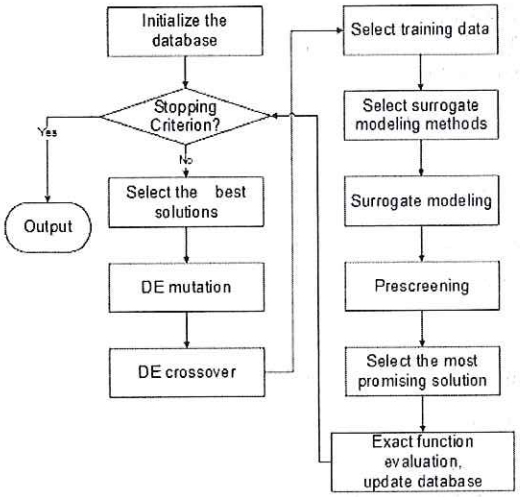

Fig. 1. The flow diagram of GPEME

B. The GPEME Framework

Like most other SAEAs, GPEME records all the evaluated solutions and their function values in a database. Once an exact function evaluation has been conducted for a new solution x, this solution and its real function value y will be added to the database. To initialize the database, a Design of Experiments method, Latin Hypercube sampling (LHS) [27], is used to sample a set of initial points from the search space. The LHS sampling method samples the design space more uniformly, and hence, can use fewer samples to achieve a more effective sampling. It has been widely used for initialization in some other SAEAs [7], [15]. Let the search space be $[a, b]^d$. The GPEME algorithm works as follows (see Fig. 1):

Step 1: Use LHS to sample a solutions from $[a, b]^d$, evaluate the real function values of all these solutions and let them form the initial database.

Step 2: If a preset stopping criterion is met, output the best solution in the database; otherwise go to step 3.

Step 3: Select the λ best solutions (i.e., with the lowest function values) from the database to form a population P.

Step 4: Apply the DE operators on P to generate λ child solutions.

Step 5: Take the τ newest solutions in the database and their function values as the training data to prescreen the λ child solutions generated in Step 4 by using GP in the original space or GP+DR with the LCB prescreening.

Step 6: Evaluate the real function value of the estimated best child solution from Step 5. Add this evaluated solution and its function value to the database. Go back to Step 2.

The flow diagram of the GPEME framework is given in Fig. 1.

We can make the following remarks on the GPEME framework:

- The population P generated in Step 3 consists of the λ best solutions in the database. Most of these solutions may not be far away from each other, particularly after several iterations. Therefore, most child solutions generated in Step 4 are in a relatively small promising subregion.
- In each iteration, at most one new solution enters P in Step 3. Hence, in several consecutive iterations, the solutions to be prescreened in Step 5, which are generated from P in step 4, should not be far away from each other, particularly, when most solutions in P are distributed in a small region. For this reason, the training data points (obtained from Step 6) could not be far from a solution to be perscreened, which is desirable for GPEME.
- The τ solutions in Step 5 are the most recently generated best candidate solutions, and it is reasonable to assume that these solutions are also not far away from the current promising subregion. Thus the model built by using these solutions should be reasonably accurate for ranking the λ child solutions generated in Step 4.
- It is not necessary to build a very accurate GP model in Step 5 for prescreening. We only require that the GP model can help to select a reasonably good solution.

Taking the Ackley function with 20 variables as an example, in GPEME (the experimental setting is given in the next section), the probabilities on 20 runs that the best ranked solution by prescreening in Step 5 is among the top 20% and 10% of the λ child solutions in terms of exact function values are 86.1% and 75.8%, respectively. Clearly, the GP modeling works very well in GPEME.

To get a rough idea of the search ability of the GPEME framework, we remove the GP modeling and prescreening. Instead, we conduct exact function evaluations to all the λ child solutions in each iteration, and randomly select one from the top β solutions. In such a way, we simulate the GP modeling and prescreening.

We set α 100 and λ 50. We have tested five different β values: 1, 3, 5, 7 and 9. The function values of the best solution found so far versus the number of iterations is plotted in Fig. 2. These results are based on 20 runs. It is clear from Fig. 2 that the convergence speed decreases as the β value increases. This indicates that the quality of the GP model does have an impact on the convergence speed of GPEME. When $1 \leq \beta \leq 7$, the best objective function value obtained with 1,000 iterations is very close to 0, the global optimal value. This implies that the proposed GPEME framework works well if the best solution based on prescreening is among the top-ranked candidates in the newly generated λ candidate solutions in terms of exact function values.

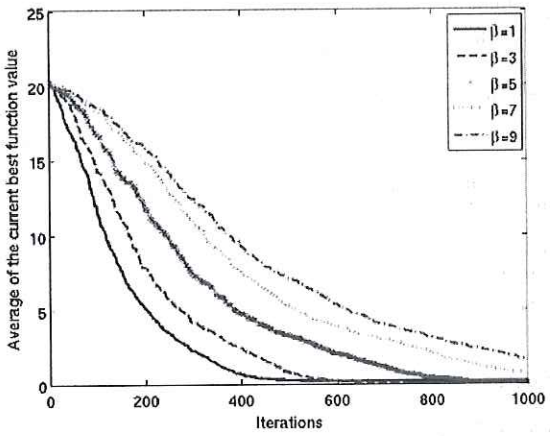

Fig. 2. Convergence Curve in *simulated* GPEME

IV. Experimental Studies

A. Parameter Settings

There are several control parameters in GPEME. To set some parameters, we have conducted pilot experiments on benchmark problems. Their settings and our considerations are given as follows:

- The scaling factor F and the crossover rate CR in the DE operators: The setting of the DE parameters has been well investigated. Following [26], we set F to 0.8, and CR to 0.8.
- The number of training data points τ in the GP modeling: The tradeoff between the model quality and the computational cost is considered when setting τ. Our pilot experiments have shown that for medium-scale problems, when using less than 80 training data to construct the surrogate model, the selected best candidate based on the GP model and prescreening is not a truly high-ranked candidate in many occasions. On the other hand, although the model quality can improve when using more training data, the computational cost of modeling also increases significantly. Here are some training times to construct a GP model (using the Blind DACE toolbox [20]) for the problems tested in this paper (see Appendix) on a Xeon 2.66GHz computer in the MATLAB environment. If 100 training data are used, for the problems with 20, 30 and 50 variables, the CPU time to construct a GP model is about 10-20 seconds, 15-35 seconds, 120 to 250 seconds, respectively. When 150 training data are used, the CPU time increases to 30-50 seconds, 80-110 seconds, and 240-400 seconds, respectively. We have found that $80 \leq \tau \leq 120$ can produce a reasonably good surrogate model for GPEME with an acceptable amount of computational time. In our experiments using benchmark problems, τ 100 is used.
- The dimensionality l of the low-dimensional space in Sammon mapping: Our pilot experiments have indicated that $l > 6$ can make the search quite slow and l 2 can lead to poor results. In our experiments, we set l 4.
- ω used in LCB: Following [10], [17], ω 2 is used.

- The number of initial samples a in Step 1: [10] suggests that at least $2d$ samples should be used to construct a reasonably good local GP model. However, the initial samples are randomly generated spreading all over the search space, so more samples are necessary. We use 100 initial samples in the experiments using benchmark problems.
- λ in Step 3: Our pilot experiments on λ 20, 30, . . . , 80 have shown that $30 \leq \lambda \leq 60$ works well. A large λ value causes slow convergence and a small value can easily lead to premature convergence. We set λ 50 in the experiments using benchmark problems.

B. Test Problems

To compare with three state-of-the-art methods reported in [1], [10], [11], we take the problems used in their works as the test instances. The test problems in [10], [11] have 20 variables. [10] uses the Sphere problem (simple unimodal), the Ellipsoid problem (unimodal), the Step problem (discontinuous) and the Ackley problem (multimodal). [11] uses the Sphere problem, the Rosenbrock problem (unimodal but with a narrow valley near the global optimum), the Ackley problem, the Griewank problem (multimodal) and the Rastrigin problem (multimodal). [1] uses test problems with 30 variables, including the Ackley problem, the Griewank problem, the Rosenbrock problem, two shifted rotated problems and five hybrid composition functions from [28]. Besides problems with 20 or 30 variables, we also test some problems with 50 variables, which have not been tested by the other methods. The test problems used in our experiments are listed in Table I and more details can be found in the Appendix.

We also test our method on a practical design optimization problem of an mm-wave power amplifier with 17 design variables.

TABLE I
TEST PROBLEMS USED IN THE EXPERIMENTAL STUDIES

Problem	Objective function	No. of Variables	Global optimum	Property	Comparison Method
F1	Ellipsoid	20	0	unimodal	[10]
F2	Ellipsoid	30	0	unimodal	N.A.
F3	Ellipsoid	50	0	unimodal	N.A.
F4	Rosenbrock	20	0	unimodal with narrow valley	[11]
F5	Rosenbrock	30	0	unimodal with narrow valley	[1]
F6	Rosenbrock	50	0	unimodal with narrow valley	N.A.
F7	Ackley	20	0	multimodal	[10], [11]
F8	Ackley	30	0	multimodal	[1]
F9	Ackley	50	0	multimodal	N.A.
F10	Griewank	20	0	multimodal	[11]
F11	Griewank	30	0	multimodal	[1]
F12	Griewank	50	0	multimodal	N.A.
F13	Shifted Rotated Rastrigin	30	-330	very complicated multimodal	[1]
F14	Rotated Hybrid Composition Function (F19 in [1])	30	10	very complicated multimodal	[1]

N.A. means that none of the three methods [1], [10], [11] has been tested on these instances.

C. The GPEME Performance and Comparisons with State-of-the-art SAEAs

The statistics of the best function values obtained by GPEME with 1,000 function evaluations on 20 independent runs for F1-F14 are reported in Table II.

From Table II, for most problems with 20 and 30 variables, GPEME can find very good solutions with 1,000 exact function evaluations. The exceptions are F4 and F5 (Rosenbrock problem with 20 and 30 variables), F13 and F14. Although the Rosenbrock problem is unimodal, the narrow valley of the global optimum makes it difficult to optimize. F13 and F14 are artificially designed complex problems. The standard DE with 30,000 exact function evaluations cannot produce good results for F5, F13 and F14 as shown in the next subsection. For the problems with 50 variables, although the obtained solutions are not very close to the global optimal solutions, their qualities are reasonably good.

We have compared GPEME with GS-SOMA [1], SAGA-GLS [11] and MAES [10]. As shown in Table I, GS-SOMA has been tested on F5, F8, F11, F13 and F14 in [1], SAGA-GLS on F4, F7 and F10 in [11], and MAES on F1 and F7 in [10]. Table III to Table V report the following values:

TABLE II
STATISTICS OF THE BEST FUNCTION VALUES OBTAINED BY GPEME IN 1,000 EXACT FUNCTION EVALUATIONS FOR F1-F14

Problem	best	worst	average	std
F1	1.27e-6	7.2e-5	1.3e-5	2.18e-5
F2	0.0155	0.1647	0.0762	0.0401
F3	134.0681	372.5567	221.0774	81.6123
F4	15.1491	75.8806	22.4287	18.7946
F5	26.2624	88.2325	46.1773	25.5199
F6	172.3547	401.4187	258.2787	80.1877
F7	0.0037	1.8403	0.1990	0.5771
F8	1.9491	4.9640	3.0105	0.9250
F9	9.2524	14.9343	13.2327	1.5846
F10	0.0002	0.2234	0.0307	0.0682
F11	0.7368	1.0761	0.9969	0.1080
F12	22.5456	64.9767	36.6459	13.1755
F13	-57.0678	18.0327	-21.8610	36.4492
F14	933.1601	992.8618	958.5939	25.6946

TABLE III
COMPARISON OF GPEME WITH GS-SOMA [1]

Problem	GPEME ($Best_{1,000}$)	GS-SOMA ($Best_{1,000}$)	GS-SOMA ($H_{1,000}$)
F5	-46.1773	2981	>8000
F8	3.0105	20	>8000
F11	0.9969	365	2500
F13	-21.86	>50	2100
F14	958.5939	>1118.8	5200

TABLE IV
COMPARISON OF GPEME WITH SAGA-GLS [11]

Problem	GPEME ($Best_{1,000}$)	SAGA-GLS ($Best_{1,000}$)	SAGA-GLS ($H_{1,000}$)
F4	22.4287	64	>6000
F7	0.1990	5.5	6000
F10	0.0307	2.1170	>6000

- $Best_{1,000}$: the mean of the best function values obtained using 1,000 exact function evaluations by GPEME;
- $H_{1,000}$: the number of exact function evaluations needed by an algorithm in [1], [10], [11] to achieve the same best function value obtained by GPEME. In total, GS-SOMA, SAGA-GLS and MAES used 8,000, 6,000, and 1,000 exact function evaluations, respectively. If their final results are worse than the GPEME result with 1,000 exact function evaluations, we denote this as $> 8,000$, $> 6,000$, and $> 1,000$, respectively.

Some of the above values have not been provided in [1], [10], [11] in numerical form, so we extract them from the figures in these papers. [10] tried four different prescreening methods and obtained four different results; we choose the best one of them in our comparison.

From Table III to Table V, it is clear that GPEME outperforms the three state-of-the-art methods on these test instances. More specifically:

- As shown in Table III, with 1,000 exact function evaluations for F5, F8, F11 and F13, GPEME can produce much better results than GS-SOMA. GS-SOMA is not able to produce solutions with the same quality with 8,000 function evaluations on F5 and F8. For F11 and F13, GPEME achieves about two times speed enhancement compared with GS-SOMA. For F14, the result obtained by GPEME is also better than GS-SOMA when both algorithms use 1,000 function evolutions, and GS-SOMA needs about 5,200 function evaluations to obtain similar results.
- The average best function values obtained by GPEME with 1,000 exact function evaluations over 20 runs are better than or similar to those obtained by SAGA-GLS with 6,000 function evaluations as shown in Table IV. The best function values of SAGA-GLS with 1,000 function evaluations are much worse than those of GPEME.

TABLE V
COMPARISON OF GPEME WITH MAES [10]

Problem	GPEME ($Best_{1000}$)	MAES ($Best_{1000}$)	MAES (H_{1000})
F1	1.3e-5	0.075	>1000
F7	0.1990	3	>1000

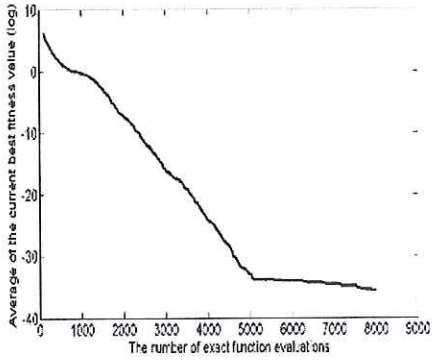

Fig. 3. Convergence curve of the objective function for F11 with 8,000 exact function evaluations by GPEME

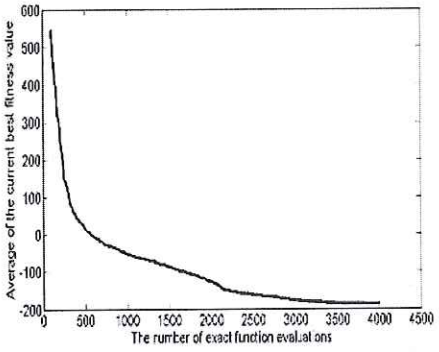

Fig. 4. Convergence curve of the objective function for F13 with 4,000 exact function evaluations by GPEME

- MAES also uses 1,000 function evaluation on F1 and F7. Table V shows that the results obtained by GPEME are better by one or two orders of magnitude than the results of MAES.

Note that for F11 and F13, GS-SOMA can achieve the result of GPEME in 2,000-3,000 exact function evaluations. The convergence curves in [1] show that the fitness value decreases very rapidly in GS-SOMA from 2,000 to 3,000 exact function evaluations. To get an insight into the behavior of GPEME with more than 1,000 function evaluations, we conduct experiments on GPEME for F11 with 8,000 exact function evaluations and for F13 with 4,000 exact function evaluations. The convergence curves of the average best function values found so far over 20 runs are given in Fig. 3 and Fig. 4. The average best function values of GS-SOMA with 8,000 exact function evaluations are $2.2e-3$ and -126 for F11 and F13, respectively [1], while the GPEME result for F11 with 8,000 exact function evaluations is $3.33e-16$ and for F13 with 4,000 exact evaluations is -187.04. From Fig. 3 and Fig. 4, it can be seen that the final result of GS-SOMA can be achieved by GPEME with about 2,000 exact function evaluations.

TABLE VI
THE EXPERIMENTAL RESULTS BY THE STANDARD DE FOR F1-F14

Problem	1,000 FEV	T	30,000 FEV
F1	203.22	12400	9.47e-17
F2	788.05	10750	4.28e-10
F3	3281.3	5750	4.65e-4
F4	534.85	4550	1.24
F5	1621.1	5650	14.53
F6	4147.9	5800	43.70
F7	16.94	9000	4.18e-9
F8	18.35	8050	0.13
F9	19.24	4100	1.18
F10	81.06	11400	0.013
F11	183.18	9500	0.0025
F12	467.77	5400	0.0069
F13	184.09	3150	-138.15
F14	995.58	1600	918.84

The second and last columns are the average best function values obtained by DE with 1,000 and 30,000 function evaluations over 20 runs, respectively. T is the number of function evaluations required by the standard DE to achieve similar results by GPEME with 1,000 function evaluations.

TABLE VII
COMPARISON BETWEEN GP+DR AND DIRECT GP FROM FIG. 5 TO FIG. 18

No. of variables	GP+DR wins	GP wins
20	1/4	3/4
30	1/6	5/6
50	4/4	0/4

The second column shows the number of problems which can provide a better result using GP+DR over the total number of test problems. The third column shows the same statistics but for the case that better results can be obtained by direct GP.

D. Comparison with Standard DE

To understand the benefits of the key techniques in GPEME, we compare GPEME with the standard DE [26] without surrogate model on F1-F14. When applying standard DE, the population size, the scaling factor F and the crossover rate CR are the same as those used in GPEME. DE/best/1 mutation is also used. The number of exact function evaluations is set to be 30,000. The experimental results are presented in Table VI.

It is evident from Table VI that GPEME performs much better than DE if the computational budget is 1,000 function evaluations. The efficiency of GPEME comes from the surrogate model-aware search mechanism and the corresponding surrogate modeling method.

E. GP+DR vs Direct GP

In the above experiments, direct GP modeling is used for the test problems with 20 and 30 variables, and GP+DR for test problems with 50 variables. In the following, we compare the performances of GP+DR and direct GP modeling in GPEME. Both modeling approaches are tested in GPEME for F1 to F14. Fig. 5 to Fig. 18 compare how the average best function value found so far deceases against the number of exact function evaluations in GPEME with direct GP and GP+DR. The results are based on 20 runs and are summarized in Table VII.

One can observe that:

- For the four problems with 20 variables, GPEME with direct GP clearly performs better than GP+DR on F1, F7 and F10, and GP+DR is a little bit better on F4.
- For the six problems with 30 variables, GPEME with direct GP is better than GP+DR on F2, F8, F11 and F14. On F5, GPEME with direct GP performs a little bit better. GP+DR wins on F13.
- On all the four problems with 50 variables, i.e., F3, F6, F9 and F12, GP+DR outperforms direct GP.

Therefore, we can conclude that it is reasonable to use the

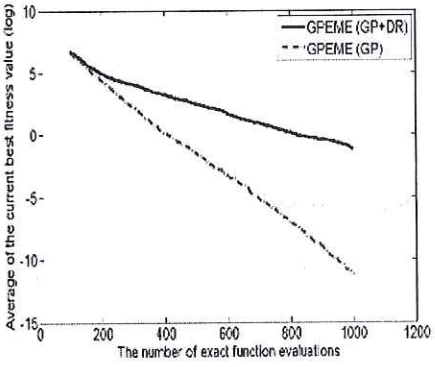

Fig. 5. Convergence curves of the objective function for F1 (20-D Ellipsoid)

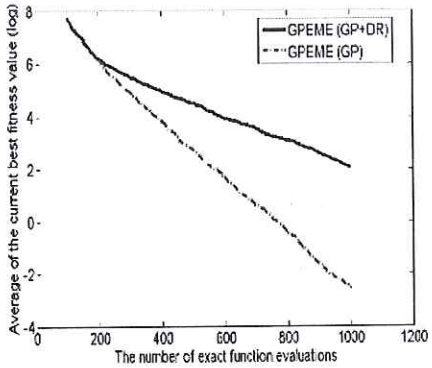

Fig. 6. Convergence curves of the objective function for F2 (30-D Ellipsoid)

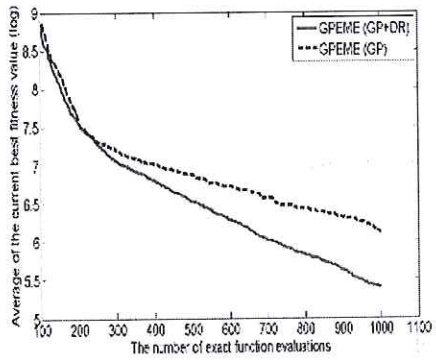

Fig. 7. Convergence curves of the objective function for F3 (50-D Ellipsoid)

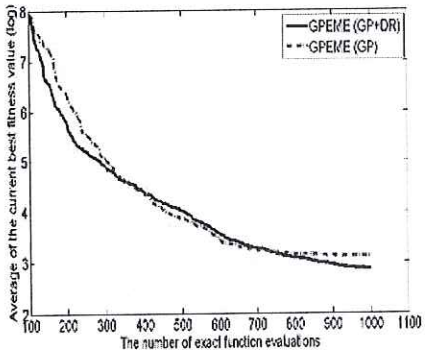

Fig. 8. Convergence curves of the objective function for F4 (20-D Rosenbrock)

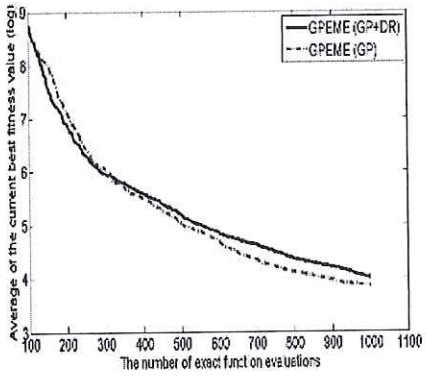

Fig. 9. Convergence curves of the objective function for F5 (30-D Rosenbrock)

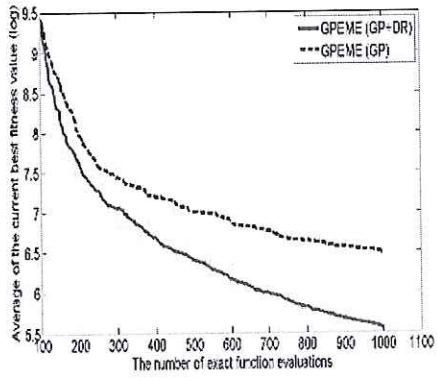

Fig. 10. Convergence curves of the objective function for F6 (50-D Rosenbrock)

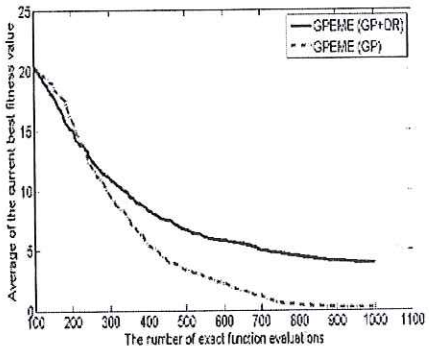

Fig. 11. Convergence curves of the objective function for F7 (20-D Ackley)

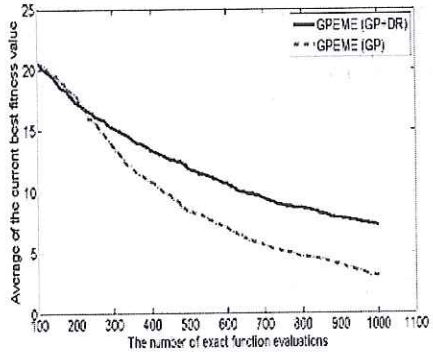

Fig. 12. Convergence curves of the objective function for F8 (30-D Ackley)

direct GP modeling for problems with 30 or less variables, and use the GP+DR approach for problems with 50 variables.

It is interesting to investigate if the models built by these two approaches are different in terms of quality and if the model quality differences are consistent with the performance differences of GPEME. Recalling that only one model is built and one solution is selected from the λ candidate solutions for exact function evaluation in each iteration, it is desirable that the selected solution is one of the best (top-ranked) candidate solutions in terms of exact function values. We divide the 900 iterations after "Initialization" (i.e., Step 1) into 45 phases, each of which contains 20 consecutive iterations. To measure

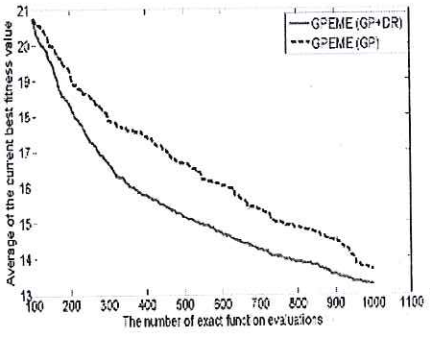

Fig. 13. Convergence curves of the objective function for F9 (50-D Ackley)

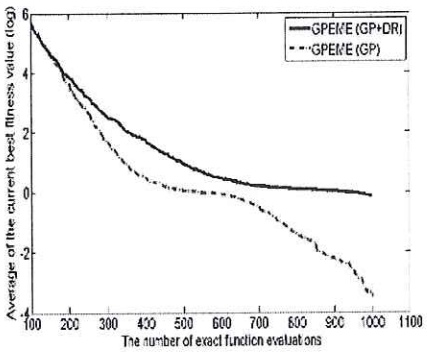

Fig. 14. Convergence curves of the objective function for F10 (20-D Griewank)

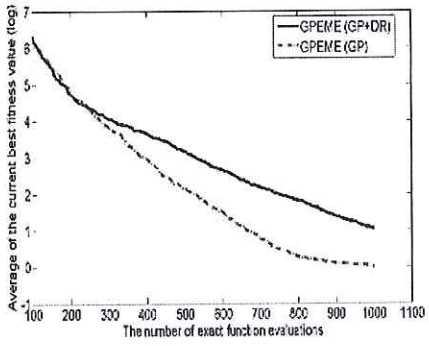

Fig. 15. Convergence curves of the objective function for F11 (30-D Griewank)

the quality of modeling in a particular phase, we define its performance score as follows:

$$\Phi \quad 3 \times N_1 + 2 \times N_2 + N_3 \tag{11}$$

where

- N_1 is the number of iterations in a phase and in which the selected solution is among the top 2 of the λ candidate solutions.

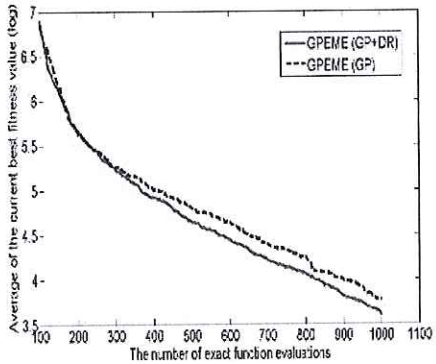

Fig. 16. Convergence curves of the objective function for F12 (50-D Griewank)

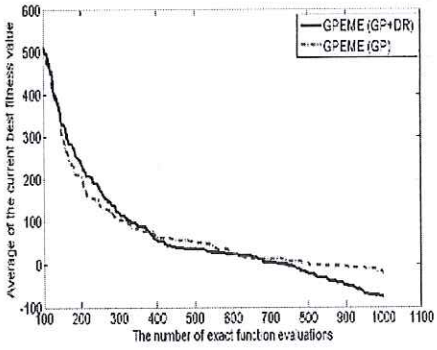

Fig. 17. Convergence curves of the objective function for F13 (30-D Shifted Rotated Rastrigin)

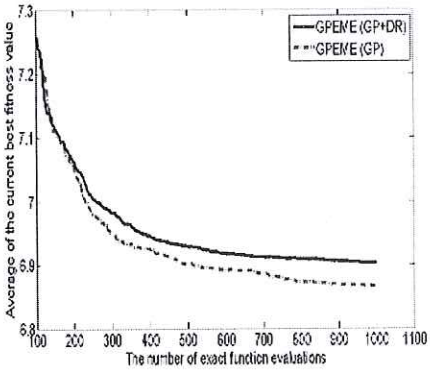

Fig. 18. Convergence curves of the objective function for F14 (30-D Rotated Hybrid Composition Function, F19 in [1])

- N_2 is the number of iterations in a phase and in which the selected solution is among top 3 to 5.
- N_3 is the number of iterations in a phase and in which the selected solution is among top 6 to 10.

If the selected solution is out of the top 10, it is not very useful for the search and thus it is not counted in the score. Clearly, the higher the performance score is, the better the modeling and prescreening performs in this phase. Fig. 19 to Fig. 22 plot the scores of two modelings (direct GP versus GP+DR) in GPEME for F1, F2 and F3 and F14. Except Fig. 22 (F14), the scores for the other problems are qualitatively similar to Fig. 19 to Fig. 21. For some instances, such as F1 and F2, the two modeling methods are quite different in terms of scores. More importantly, one can find that the score differences are consistent with the algorithm performance differences. For example, the scores of GP+DR are higher than direct GP on F3 as shown in Fig. 21, and GPEME with GP+DR outperforms the one with direct GP as shown in Fig. 7. The scores of direct GP are higher on F1 as shown in Fig. 19, and correspondingly, GPEME with it wins as shown in Fig. 5. F14 is the only exception. But as said above, F14 is an artificially constructed hard problem and it is difficult to optimize. Using the scores, we may be able to improve GPEME by developing more effective and efficient modeling techniques.

In GPEME, we use Sammon mapping to do dimension

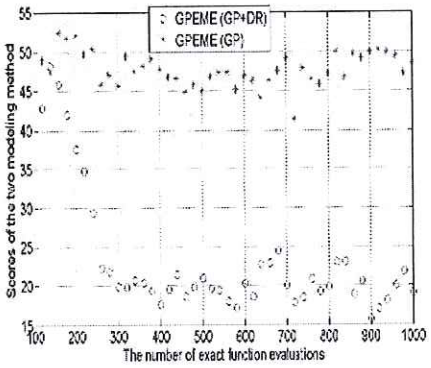

Fig. 19. Scores of the two modeling approaches for F1 (20-D Ellipsoid)

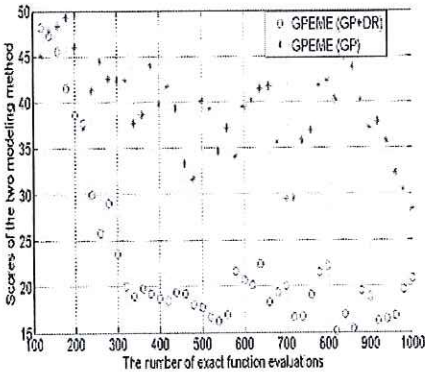

Fig. 20. Scores of the two modeling approaches for F2 (30-D Ellipsoid)

reduction. To study the effectiveness of Sammon mapping, some other dimension reduction methods are compared with Sammon mapping in GPEME on F7. These methods are PCA, Linear Discriminant Analysis (LDA), Local Linear Embedding (LLE) and Neighborhood Components Analysis (NCA) [23]. The average best function values found with 1,000 exact function evaluations over 10 runs are shown in Table VIII. It can be seen that Sammon mapping is the best.

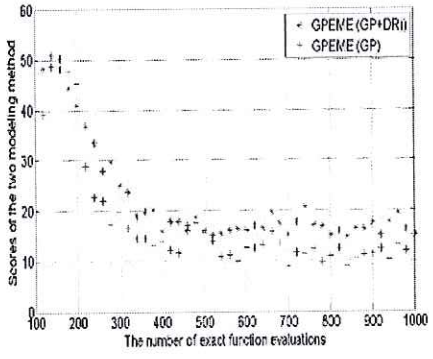

Fig. 21. Scores of the two modeling approaches for F3 (50-D Ellipsoid)

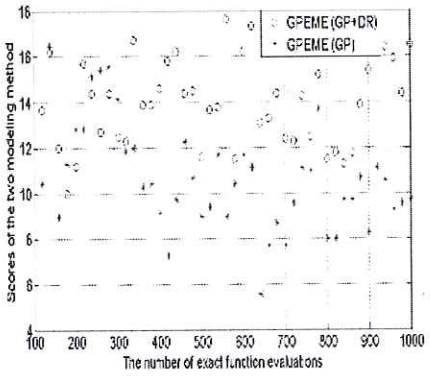

Fig. 22. Scores of the two modeling approaches for F14 (30-D Rotated Hybrid Composition Function, F19 in [1])

TABLE VIII
RESULT OF F7 USING FIVE DIMENSION REDUCTION METHODS

Sammon	PCA	LDA	LLE	NCA
3.57	14.93	12.5	18.73	13.29

F. mm-wave Integrated Circuit Optimization

GPEME has been designed for expensive engineering optimization problems. This subsection provides the experimental results of GPEME for dealing with an mm-wave integrated circuit (IC) design optimization problem. In recent years, high-frequency integrated circuit design is attracting more and more attention, both in academia and industry. At high frequencies, simple equivalent circuit models for passive components are no longer accurate to be used, and a "trial and error" method is inevitably tedious. Therefore, automatic design optimization of mm-wave ICs is of great practical importance. However, electromagnetic simulation is required to evaluate candidate designs, which is computationally expensive. Many mm-wave ICs have from 10 to 30 design parameters (i.e., decision variables) to be optimized.

We consider a design optimization problem of a 60GHz 65nm power amplifier with 17 design parameters. The evaluation of one candidate design requires 10-12 minutes using ADS-Momentum, which is a popular electromagnetic simulator. The synthesis time available to this problem is typically one to three days. In this problem, the design parameters are the inner diameters and metal width of the primary and secondary inductors of every transformer. There are three transformers with two inductors in each. The feasible ranges are $20\mu m$ to $100\mu m$ for the inner diameter, and $3\mu m$ to $10\mu m$ for the metal width. There are also 5 biasing voltages with ranges from 0.5V to 2V. The optimization problem is shown in (12). This is a simulation-based optimization problem, so no explicit analytical formulations for its objective and constraints are available.

REFERENCES

[1] D. Lim, Y. Jin, Y. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, 2010.

[2] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[3] D. Allstot, K. Choi, and J. Park, *Parasitic-aware optimization of CMOS RF circuits*. Springer Netherlands, 2003.

[4] T. Milligan and J. Wiley, *Modern antenna design*. Wiley Online Library, 2005.

[5] L. Mercado, S. Kuo, T. Lee, and R. Lee, "Analysis of RF MEMS switch packaging process for yield improvement," *IEEE Transactions on Advanced Packaging*, vol. 28, no. 1, pp. 134–141, 2005.

[6] H. Chan and P. Englert, *Accelerated stress testing handbook*. IEEE Press, 2001.

[7] D. Jones, M. Schonlau, and W. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[8] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 1, pp. 3–12, 2005.

[9] B. Liu, D. Zhao, P. Reynaert, and G. Gielen, "Synthesis of integrated passive components for high-frequency RF ICs based on evolutionary computation and machine learning techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 10, pp. 1458–1468, 2011.

[10] M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 421–439, 2006.

[11] Z. Zhou, Y. Ong, P. Nair, A. Keane, and K. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 1, pp. 66–76, 2007.

[12] D. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[13] K. Giannakoglou, "Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence," *Progress in Aerospace Sciences*, vol. 38, no. 1, pp. 43–76, 2002.

[14] Y. Ong, K. Y. Lum, and P. Nair, "Hybrid evolutionary algorithm with hermite radial basis function interpolants for computationally expensive adjoint solvers," *Computational Optimization and Applications*, vol. 39, no. 1, pp. 97–119, 2008.

[15] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.

[16] I. Voutchkov and A. Keane, "Multi-objective optimization using surrogates," *Computational Intelligence in Optimization*, pp. 155–175, 2010.

[17] J. Dennis and V. Torczon, "Managing approximation models in optimization," *Multidisciplinary design optimization: State-of-the-art*, pp. 330–347, 1997.

[18] E. Ackermann, J. de Villiers, and P. Cilliers, "Nonlinear dynamic systems modeling using Gaussian processes: Predicting ionospheric total electron content over south africa," *J. Geophys. Res*, vol. 116, no. A10, p. A10303, 2011.

[19] D. Buche, N. Schraudolph, and P. Koumoutsakos, "Accelerating evolutionary algorithms with Gaussian process fitness function models," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 35, no. 2, pp. 183–194, 2005.

[20] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, and K. Crombecq, "A surrogate modeling and adaptive sampling toolbox for computer based design," *The Journal of Machine Learning Research*, vol. 11, pp. 2051–2055, 2010.

[21] C. Rasmussen, "Gaussian processes in machine learning," *Advanced Lectures on Machine Learning*, pp. 63–71, 2004.

[22] J. Sacks, W. Welch, T. Mitchell, and H. Wynn, "Design and analysis of computer experiments," *Statistical science*, vol. 4, no. 4, pp. 409–423, 1989.

[23] L. Maaten, E. Postma, and H. Herik, "Dimensionality reduction: A comparative review," *Published online*, vol. 71, no. January, pp. 2596–2603, 2008.

[24] J. Sammon Jr, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. 100, no. 5, pp. 401–409, 1969.

[25] M. Avriel, *Nonlinear programming: analysis and methods*. Dover Pubns, 2003.

[26] K. Price, R. Storn, and J. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer-Verlag New York Inc, 2005.

[27] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[28] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *Nanyang Technological University, Singapore, Tech. Rep*, vol. 2005005, 2005.

[29] D. Zhao, Y. He, L. Li, D. Joos, W. Philibert, and P. Reynaert, "A 60 GHz 14 dBm power amplifier with a transformer-based power combiner in 65 nm CMOS," *International Journal of Microwave and Wireless Technologies*, vol. 3, no. 2, pp. 99–105, 2011.

Bo Liu received the B.S. degree from Tsinghua University, P. R. China, in 2008. He received his Ph.D. degree at the MICAS laboratories of the Katholieke Universiteit Leuven, Belgium, in 2012.

He is currently a lecturer (Assistant Professor) with Department of Computing, Glyndwr University, U.K. From October 2012 to March 2013, he is a Humboldt research fellow and is working with Technical University of Dortmund and Technical University of Munich, Germany. His research interests lie in evolutionary computation, machine learning, fuzzy logic and design automation methodologies of analog / RF integrated circuits and antennas. He has authored or coauthored 1 book and more than 20 papers in international journals and conference proceedings.

Dr. Liu was a session organizer on computationally expensive optimization and optimization in uncertain environments in IEEE World Congress on Computational Intelligence 2012. He is in the editorial board or as a reviewer in artificial intelligence and electronic design automation fields, such as IEEE Transactions on Evolutionary Computation, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Journal of Engineering. He is also a book reviewer of Elsevier and Bentham Science Publishers.

Qingfu Zhang (M'01-SM'06) received the BSc in mathematics from Shanxi University, China in 1984, the MSc in applied mathematics and the PhD in information engineering from Xidian University, China, in 1991 and 1994, respectively.

He is currently a Professor with the School of Computer Science and Electronic Engineering, University of Essex, UK. From 1994 to 2000, he was with the National Laboratory of Parallel Processing and Computing, National University of Defence Science and Technology, China, Hong Kong Polytechnic University, Hong Kong, the German National Research Centre for Information Technology (now Fraunhofer-Gesellschaft, Germany), and the University of Manchester Institute of Science and Technology, Manchester, U.K. He holds two patents and is the author of many research publications. His main research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications.

Dr. Zhang is an Associate Editor of the IEEE Transactions on Evolutionary Computation and the IEEE Transactions on Systems, Man, and Cybernetics: Part B. He is also an Editorial Board Member of three other international journals. MOEA/D, a multobjevitve optimization algorithm developed in his group, won the Unconstrained Multiobjective Optimization Algorithm Competition at the Congress of Evolutionary Computation 2009, and was awarded the 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award.

Georges G. E. Gielen (S'87-M'92-SM'99-F'02) received his M.Sc. and Ph.D. degrees in Electrical Engineering from the Katholieke Universiteit Leuven, Belgium, in 1986 and 1990, respectively. In 1990, he was appointed as a postdoctoral research assistant and visiting lecturer at the department of Electrical Engineering and Computer Science of the University of California, Berkeley. In 1993, he was appointed assistant professor at the Katholieke Universiteit Leuven, where he was promoted to full professor in 2000.

His research interests are in the design of analog and mixed-signal integrated circuits, and especially in analog and mixed-signal CAD tools and design automation (modeling, simulation and symbolic analysis, analog synthesis, analog layout generation, analog and mixed-signal testing). He has authored or coauthored two books and more than 400 papers in edited books, international journals and conference proceedings. He received the 1995 Best Paper Award in the John Wiley international journal on Circuit Theory and Applications, and was the 1997 Laureate of the Belgian Royal Academy on Sciences, Literature and Arts in the discipline of Engineering. He received the 2000 Alcatel Award from the Belgian National Fund of Scientific Research for his innovative research in telecommunications, and won the DATE 2004 Best Paper Award. He is a Fellow of the IEEE, served as elected member of the Board of Governors of the IEEE Circuits And Systems (CAS) society and as chairman of the IEEE Benelux CAS chapter. He served as the President of the IEEE Circuits And Systems (CAS) Society in 2005.