Efficient Global Optimization of Actuator Based on A Surrogate Model Assisted Hybrid Algorithm

Bo Liu, *Senior Member, IEEE*, Vic Grout, *Senior Member, IEEE*, Anna Nikolaeva

***Abstract*—Computationally expensive numerical techniques are often involved in the actuator design optimization process, with efficiency a major issue. Although surrogate-based optimization is a promising solution, challenge to the optimization efficiency is still considerable. Aiming to address this challenge, a new method, called the Parallel Adjoint Sensitivity and Gaussian Process-assisted Hybrid Optimization Technique (PAGHO), is presented. The central concept is a new optimization framework employing computationally cheap partial derivatives obtained by the adjoint sensitivity method to tackle computationally expensive infill sampling for surrogate-based optimization. A silicon micro-actuator and a mathematical benchmark problem with different kinds of challenges are selected as the test cases. Comparison results show that PAGHO can obtain comparable results with popular global optimization methods, while at the same time having significant advantages in efficiency compared to standard global optimization methods and state-of-the-art surrogate-based optimization methods.**

***Index Terms*—Design optimization, Actuator design, Efficient global optimization, Simulation-driven optimization, Gaussian Process, Adjoint sensitivity**

I. Introduction

Simulation-driven shape design optimization plays an important role in the actuator design process. Given a predefined structure, geometric parameters are optimized to satisfy the design specifications and minimize/maximize a design objective. Among various optimization methods, evolutionary algorithms (EAs) are attracting much attention because of their global optimization ability, free of a good initial design, generality and robustness [1], [2]. Although there exist computationally cheap analytical equation/behavioral models for some design cases, to obtain accurate performance of an actuator, numerical techniques (e.g., finite element analysis) are often unavoidable in simulation, which are computationally expensive. EAs often need thousands to several tens of thousands of simulations to find the optimal design. Therefore, the actuator optimization process can be very time-consuming [3], [4].

To address this challenge, surrogate model-assisted evolutionary algorithms (SAEAs) seem to be a promising method. In the context of engineering design optimization, a surrogate model, which is often constructed by statistical learning techniques, is a computationally cheap mathematical model approximating the output of numerical simulations. By coupling surrogate models with an EA, some of the computationally expensive simulations can be replaced by the surrogate model predictions; the computational cost can, therefore, be reduced significantly.

Some SAEAs use an off-line surrogate model. An initial random sampling is firstly carried out, then a surrogate model is constructed using the samples and used for prediction. The main drawback of such methods is lack of scalability. When the number of design variables is more than a few, obtaining an accurate enough surrogate model may need a considerable number of initial samples. For example, in [5], the 6 design variables for a MEMS device are cut down to 3 (other design variables are kept constant) to obtain an accurate enough artificial neural network model in a reasonable time.

To address this problem, on-line SAEAs [6], [7], [8] have been investigated and employed in engineering design. In terms of model management method, available SAEAs mainly include conservative SAEAs [7], [9], active SAEAs [10], [11] and non-standard EA structure-based SAEAs [8]. Conservative SAEAs can often obtain highly optimal designs comparable to standard EAs at the cost of reduced efficiency improvement [7], [12]. Active SAEAs, in contrast, are often efficient but the optimization ability becomes a weakness [10] although prescreening methods may help [13], [11]. In the last category, a typical example is the surrogate model-aware evolutionary search (SMAS) framework [8]. SMAS-based SAEAs show comparable results with conservative SAEAs but use 10% to 40% of the number of exact function evaluations compared to some popular conservative and active SAEAs [7], [9], [10] based on benchmark problems [8]. They are also employed in electromagnetic and actuator design optimization and obtain excellent results [14], [15], [4]. Recently, SMAS has also been employed in multi-fidelity design optimization [16], [17].

However, even SMAS-based SAEAs are not efficient enough. For example, [14] is arguably the current most efficient method for antenna global optimization based on comparisons, but the authors acknowledge that it is only suitable for problems with less than 30 minutes/simulation. In actuator design optimization, it is normal that each simulation costs several tens of minutes and the computational cost may even be higher when involving multiphysics models. Therefore, further efficiency improvement is in great need.

The reason why SAEAs are more efficient than standard EAs is that additional information (i.e., prediction model) that supports optimization is available. Hence, to further improve the efficiency, a natural idea is to obtain more supportive information. A straightforward one is the derivative information, which is essential in many traditional optimization methods. However, approximating the partial derivatives of a single candidate design for a d-dimensional design problem needs $d+1$ simulations using the finite difference method, which is

Manuscript received April 18th, 2017. B. Liu is with Wrexham Glyndwr University and The University of Birmingham, U.K.. V. Grout is Wrexham Glyndwr University, UK. A. Nikolaeva is with Bauman Moscow State Technical University, Russia, and Wrexham Glyndwr University, U.K.. (email: liubo168@gmail.com, b.liu@glyndwr.ac.uk, v.grout@glyndwr.ac.uk, Nikolaeva@bmstu.ru).

very computationally expensive.

Nevertheless, adjoint sensitivity techniques [18] may provide substantial help in terms of efficiency. Adjoint sensitivity techniques aim to obtain partial derivatives with little extra computing overhead beyond a single simulation [19] so as to efficiently analyze the robustness of a design. While known in the numerical techniques domain for decades, its use for simulation-driven optimization is only seen in recent years [20], [21]. Moreover, adjoint partial derivatives have become available in optimizers of a few commercial software tools recently, e.g., COMSOL Multiphysics [22] and CST Microwave Studio [23]. Given the advantages of the adjoint sensitivity techniques which make the idea of utilizing partial derivatives in an SAEA computationally affordable, the key challenge locates at the new framework to make derivative information and surrogate model work harmonically.

To address this problem, a new method is proposed, called Parallel Adjoint Sensitivity and Gaussian Process-assisted Hybrid Optimization Technique (PAGHO). Note that the working environment of PAGHO is normal multi-core desktop workstations. Its capacity is often 3-5 simulations in parallel with shared memory. High-Performance Computing facilities that are often not available to many engineers are not considered in this paper. PAGHO aims to:

- Achieve comparable results with methods which directly embed numerical simulations into a standard EA for actuator design optimization (often considered the best in terms of solution quality);
- Considerably improve the efficiency compared with employing standard EAs as well as state-of-the-art on-line SAEAs;
- Be general enough to handle actuator design optimization without any ad-hoc analysis or initial designs.

The remainder of this paper is organized as follows. Section II introduces the basic techniques. Section III introduces the PAGHO algorithm, including its main ideas, its general framework, the design of key algorithm components, discussions on model management and the parameter settings. Section IV presents the performance of PAGHO through a micro-actuator and a challenging mathematical benchmark problem. Comparisons with a standard differential evolution algorithm and an SMAS-based on-line SAEA are provided. Section V offers conclusions.

II. Basic Techniques

A. Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is arguably the most successful technique for constrained nonlinear continuous optimization [24]. Although it is a derivative-based method and aims to find a local optimum, convergence speed is its clear advantage [24]. SQP can also be considered as a surrogate-based optimization method, although its surrogate model is based not on statistical learning but on a Taylor series. A brief introduction is as follows; more details are in [24].

In SQP, the constrained optimization problem is modeled as a quadratic programming (QP) problem in each iteration. A Lagrangian function is firstly constructed (1):

$$L(x,\beta) = f(x) + \sum_{i=1}^{m} \beta_i g_i(x) \tag{1}$$

where $f(x)$ is the objective function, $g_i(x), i = 1, 2, \ldots, m$ are constraints and $\beta_i, i = 1, 2, \ldots, m$ are Lagrange multipliers. The QP problem in the k^{th} iteration for (1) is:

$$\begin{array}{ll} min & \nabla f(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 (L_k) d_k \\ s.t. & \nabla g_i(x_k)^T d_k + g_i(x_k) \leq 0, i = 1, 2, \ldots, m \end{array} \tag{2}$$

To reduce the computing overhead, the Hessian matrix of the Lagrangian function L can be approximated by first-order partial derivatives. A widely used method is BFGS [25], which is used in this paper.

(2) can be solved by using any QP algorithm. The solution d_k can be used to form a new iteration:

$$x_{k+1} = x_k + a_k d_k \tag{3}$$

where a_k is the step length, which can be obtained by a line search method. In this work, the implementation is based on the MATLAB Optimization Toolbox.

B. Differential Evolution (DE)

The DE algorithm outperforms many EAs for continuous optimization problems [26] and is widely used in engineering design optimization. In DE, mutation is the main approach to exploring the design space. There are a few different DE mutation strategies trading off the convergence speed and the population diversity (implying higher global exploration ability) in different ways. This paper involves the following DE mutation strategies ((4) to (6)):

(1) mutation strategy: DE/rand/1

$$v^i = x^{r_1} + F \cdot (x^{r_2} - x^{r_3}) \tag{4}$$

where x^{r_1} and x^{r_2} and x^{r_3} are three different solutions randomly selected from P (the current population). v^i is the i^{th} mutant vector in the population after mutation. $F \in (0, 2]$ is a control parameter, often called the scaling factor.

(2) mutation strategy: DE/current-to-best/1

$$v^i = x^i + F \cdot (x^{best} - x^i) + F \cdot (x^{r_1} - x^{r_2}) \tag{5}$$

where x^i is the i^{th} vector in the current population and x^{best} is the best candidate in the current population.

(3) mutation strategy: DE/rand/2

$$v^i = x^{r_1} + F \cdot (x^{r_2} - x^{r_3}) + F \cdot (x^{r_4} - x^{r_5}) \tag{6}$$

where x^{r_4} and x^{r_5} are two different solutions randomly selected from P and are different from x^{r_1}, x^{r_2} and x^{r_3}.

Crossover is then applied to the population of mutant vectors to produce the child population U, which works as follows:

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:

$$u_j^i = \begin{cases} v_j^i, & \text{if } (rand \leq CR) | j = j_{rand} \\ x_j^i, & \text{otherwise} \end{cases} \tag{7}$$

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

For constraint handling, the penalty function method [27] is used.

C. Basics of Gaussian Process (GP)

GP machine learning [28] is used to construct surrogate models in PAGHO. The main reason for selecting GP is that the prediction uncertainty has a sound mathematical background, which is able to take advantage of prescreening methods [10] for surrogate model-based optimization. An intuitive introduction is as follows. More details are in [28].

Given a set of observations $x = (x^1, \ldots, x^n)$ and $y = (y^1, \ldots, y^n)$, GP predicts a function value $y(x)$ at some design point x by modeling $y(x)$ as a Gaussian distributed stochastic variable with mean μ and variance σ. If the function is continuous, the function values of two points x^i and x^j should be close if they are highly correlated. In this work, we use the Gaussian correlation function to describe the correlation between two variables:

$$corr(x^i, x^j) = exp(-\sum_{l=1}^{d} \theta_l |x_l^i - x_l^j|^2) \tag{8}$$

where d is the dimension of x and θ_l is the correlation parameter, which determines how fast the correlation decreases when x^i moves in the l direction. The values of μ, σ and θ are determined by maximizing the likelihood function that $y = y^i$ at $x = x^i (i = 1, \ldots, n)$. The optimal values of μ and σ can be found in a closed form, which are as follows:

$$\hat{\mu} = (I^T R^{-1} y)^{-1} I^T R^{-1} y \tag{9}$$

$$\hat{\sigma^2} = (y - I\hat{\mu})^T R^{-1} (y - I\hat{\mu}) n^{-1} \tag{10}$$

where I is a $n \times 1$ vector of ones, R is the correlation matrix and

$$R_{i,j} = corr(x^i, x^j), i, j = 1, 2, \ldots, n. \tag{11}$$

Using the GP model, the function value $y(x^*)$ at a new point x^* can be predicted as (x^* should be included in R):

$$\hat{y}(x^*) = \hat{\mu} + r^T R^{-1} (y - I\hat{\mu}) \tag{12}$$

where

$$r = [corr(x^*, x^1), corr(x^*, x^2), \ldots, corr(x^*, x^n)]^T \tag{13}$$

The measurement of the uncertainty of the prediction (mean square error), which is used to assess the model accuracy, can be described as:

$$\hat{s^2}(x^*) = \hat{\sigma^2}[I - r^T R^{-1} r + (I - r^T R^{-1} r)^2 (I^T R^{-1} I)^{-1}] \tag{14}$$

In this work, we use the ooDACE toolbox [29] to implement the GP surrogate modeling.

To make use of the prediction uncertainty to assist SAEA, the lower confidence bound prescreening [10], [13] is selected. We consider the minimization of $y(x)$ in this paper. Given the predictive distribution $N(\hat{y}(x), s^2(x))$ for $y(x)$, a lower confidence bound prescreening of $y(x)$ can be defined as [13]:

$$\begin{aligned} &y_{lcb}(x) = \hat{y}(x) - \omega s(x) \\ &\omega \in [0, 3] \end{aligned} \tag{15}$$

where ω is a constant, which is often set to 2 to balance exploration and exploitation ability [10].

III. The PAGHO Algorithm

A. Main Ideas of PAGHO

Although surrogate modeling makes SAEAs much more efficient than standard EAs, SAEAs face a dilemma. In general, prediction uncertainty degrades the optimization ability, because some optimal designs may be predicted wrongly and thus the SAEA search may be guided in incorrect directions. To make a good prediction, more training data points through expensive numerical simulations are necessary to maintain the surrogate model quality, which decrease the efficiency. Despite some SAEAs calling for an appropriate balance between optimization quality and efficiency, a number of candidate designs far from the optimal region still have to be simulated to understand the search space (providing a reasonably good global surrogate model), which is unavoidable for almost all kinds of available SAEAs to the best of our knowledge. Reducing such simulations is the main idea of PAGHO.

Clearly, derivative information showing the direction of a possible optimum is useful. With the help of the adjoint sensitivity method, the efficiency is no longer a challenge, the central problem therefore becomes the appropriate way to use derivative information in an SAEA. Related works include [30], [9]. One type of methods (e.g., [30]) uses partial derivatives to improve the reliability of space mapping (a multi-fidelity surrogate-based local optimization method), which has a different purpose compared to PAGHO. Another type of methods follows the memetic algorithm structure [31]; in particular, SQP is used to improve the promising designs generated by a conservative SAEA in [9]. However, an SMAS-based SAEA is more efficient than this [8].

Rather than using derivative-based optimization as local refinements, in PAGHO, derivative-based optimization is *firstly* employed aiming to find a few local optimal solutions. In this process, many non-optimal regions are escaped efficiently with the help of the derivatives. An SAEA is then started from the suboptimal region obtained. This SAEA should be much more efficient than starting from random sampling. Clearly, a fast convergence derivative-based optimization method is needed for the initial stage of PAGHO: SQP is therefore the choice. The limitation of this new framework is that it is not fit for problems which are not suitable for derivative-based optimization. For example, problems with optimum located in a narrow valley (e.g., microwave filters). For this kind of problems, derivative-based optimization from random starting points is unlikely to reach a local optimal region [32].

The construction of this new framework is NOT trivial. For an SAEA, several tens of *well distributed* individuals are often needed in the population, which cannot be provided by SQP. Various pilot experiments using different methods to initialize the population of SAEA with the SQP result have been carried out. Results show that, although SQP can provide the SAEA with a relatively good starting region, the SAEA is often trapped in a local optimum near/at the starting point. The central question then becomes how to make the SAEA

jump out of local optima, while at the same time taking the benefits of the SQP result? This is solved by a new search framework (Section III (B)), for which, the two key operators are described in Section III (C) and Section III (D).

B. The General Framework of PAGHO

The flow diagram of the PAGHO algorithm is shown in Fig. 1, which consists of the following steps:

Step 1: Select λ solutions from the design space using the Latin Hypercube sampling method [33]. Numerical simulation is not performed.

Step 2: Cluster the λ points into h clusters using the k-means method [34] (h is the number of simulations that can be run in parallel). Select the point (from the λ samples) that is the nearest to the centroid of each cluster to generate h starting points for SQP.

Step 3: Carry out SQP (Section II (A)) from each starting point in parallel and save all the simulated candidate designs and their performances in the database. Partial derivatives are obtained by the adjoint sensitivity method.

Step 4: Use the method in Section III (C) to obtain λ designs as the initial population of the SAEA.

Step 5: If a preset stopping criterion (e.g., computing budget) is met, output the best solution from the database; otherwise, go to Step 6.

Step 6: Select the λ best solutions from the database to form a population P.

Step 7: Use the method in Section III (D) to generate h groups of child solutions (each group has λ child solutions).

Step 8: For each group, calculate the median of the λ child solutions. Take the τ nearest solutions to the median from the database (based on Euclidean distance) and their function values (performances) as the training data points to construct GP surrogate models.

Step 9: For each group, prescreen the λ child solutions generated in Step 8 using the LCB method (Section II (C)) and obtain the estimated best child solution in each group (h solutions in total). Simulate the h solutions in parallel. Add the solutions and their performances (via numerical simulation) to the database. Go back to Step 5.

It can be seen that PAGHO begins with a multiple start SQP. To improve the diversity for constructing the initial population of the SAEA (i.e., implying higher exploration ability), it is desirable that the starting points should represent complementary areas of the design space and should be far away from each other. Therefore, Latin Hypercube sampling and k-means clustering are used. Note that PAGHO considers only a few (e.g., 3-5) parallel simulations as described in Section I, so the solution diversity provided by multiple start SQP is still limited. Hence, a key operator is developed to generate an initial population for the SAEA in Step 4.

Steps 5-9 are the SAEA part of PAGHO. Some key concepts are borrowed from the SMAS framework [8], [12] (Steps 6 and 9). The central idea of SMAS is to improve the locations of

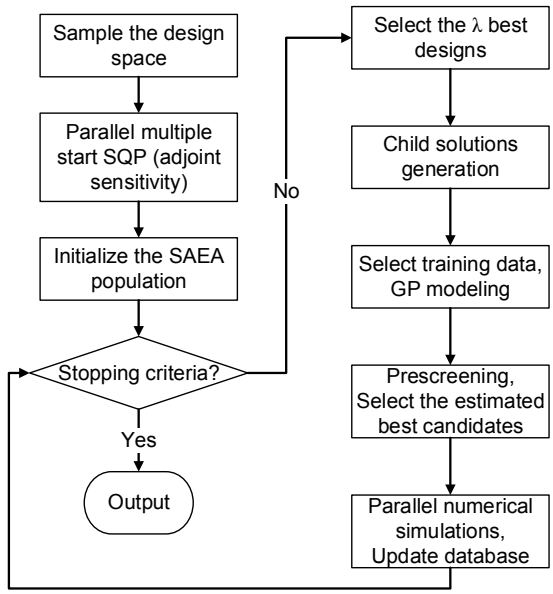

Figure 1. The flow diagram of PAGHO

training data points. With the same number of training data points, it is intuitive that using training data points located near to the points waiting to be predicted (child population in Step 7) can obtain a better surrogate model. Hence, in each iteration, the λ current best candidate solutions construct the parent population and a few best candidates based on prescreening in the child population are selected to replace the worst ones in the parent population. Hence, only at most a few candidates are changed in the parent population in each iteration, so the best candidates in the child solutions in several consecutive iterations may be reasonably near to each other (they will then be simulated and are used as training data points). Therefore, the training data points describing the current promising region can be much denser compared to those generated by a standard EA population updating mechanism. It can be seen that this property is inherited in PAGHO in Steps 6 and 9.

However, experiments show that directly employing standard SMAS in [8] causes PAGHO to be trapped in local optima. The reason is that the initial population is provided based on the SQP result instead of Latin Hypercube sampling of the whole design space as [8]. Step 4 helps substantially but is not a complete solution. Therefore, a new method (Step 7) is developed to address this problem. Steps 4 and 7 are the main innovations of PAGHO compared to SMAS besides the new framework of introducing partial derivatives into SAEAs.

C. Initial Population Generation of SAEA

The input to this operator (Step 4 of PAGHO) is the database comprising all the candidate designs visited by SQP and their performances. The output is the initial population P for the SAEA with λ candidate designs. The procedure consists of the following steps:

Step 1: Refine the database by keeping the top 75% of the candidate designs in terms of performance optimality to construct a selection pool S.

Step 2: Initialize P with the best candidate design. Remove the best candidate design from S.

Step 3: If $|P|$ is equal to λ, output P. If $|P|$ is smaller than λ and $|S|$ is larger than 0, go to Step 4. Otherwise; go to Step 5.

Step 4: Calculate the scores of all the candidates in S using the following equation:

$$score = 1 - \frac{f(x)-min(f(x))}{max(f(x))-min(f(x))} + \frac{dist(x)-min(dist(x))}{max(dist(x))-min(dist(x))} \quad (16)$$

where $f(x)$ is the fitness function (a minimization problem is considered), $dist(x)$ is the Euclidean distance between a candidate x in S and the nearest candidate to x in P.

Add the one with the highest score to P and remove it from S. Go back to Step 3.

Step 5: Apply the DE/rand/1 mutation (4) and the crossover operators (7) on P to generate $|P|$ child solutions.

Step 6: Use the database as the training data points to construct a GP surrogate model.

Step 7: Prescreen the $|P|$ child solutions generated in Step 6 using the LCB method in Section II (C), obtain the estimated best child solution and simulate it. Add the solution and its performance (via simulation) to P and the database. Go back to Step 3.

The algorithm starts by removing the SQP visited solutions that are far from optimal from S (Step 1). Those solutions often do not contain good patterns in terms of optimality and should not be included in new candidate generation, although they can provide more diversity. Due to the fast convergence of SQP, the first few steps of SQP are often large, and the number of far from optimal solutions is often small. This is the reason for using 75% as the threshold.

For the next step, the remaining candidates in S are selected balancing their contribution to optimality (performance values) and diversity (shortest distance to P). Normalization is used to calculate the score. However, sometimes the candidates in S are not enough to fill P. In such cases, instead of using solutions far away from the optimal region, new candidates are generated to fill the vacancies. DE/rand/1 is used to generate new candidates, which trades off the optimality and diversity of P. Experiments show that using DE/rand/1 provides better result than using DE/current-to-best/1 (5) and DE/rand/2 (6).

D. Child Solution Generation

h groups of child solutions are generated from P in Step 7 of PAGHO using DE mutations, which is another key operator. As described in Section III (B), employing the traditional SMAS framework often cannot jump out of local optima even with the improved initial population. Hence, selecting the correct DE mutation strategy, compensating the population diversity while maintaining convergence speed, is the key of this operator. Each DE mutation operator ((4) to (6)) trades off the exploration ability (population diversity) and convergence speed (optimality) to a certain extent. Among them, DE/current-to-best/1 (5) has the least diversity consideration but the convergence speed is the fastest, DE/rand/2 (6) emphasizes the promotion of population diversity but with the slowest convergence speed. DE/rand/1 (4) is in the middle in both respects.

It is observed that, due to different problem landscapes, the solutions provided by SQP are of diverse characteristics and, after Step 4 of PAGHO, the initial population constructed also shows diverse optimality and diversity. Hence, it is difficult to select a universal DE mutation operator for all SQP outputs for different kinds of function landscapes and initial populations. To address this challenge, this section provides a self-adaptive method, which works as follows:

For each child population $i = 1, 2, \ldots, h$

Step 1: If the algorithm is within the learning period (the current number of iterations is smaller than a threshold L), the rate of using DE/rand/1 (4), DE/current-to-best/1 (5) and DE/rand/2 (6) is $\frac{1}{3}$. Otherwise; use the rates in Step 5.

Step 2: Perform a roulette wheel selection [35] based on the rates to determine a DE mutation method and generate a child population C_i comprising λ child solutions.

Step 3: Calculate the median of the λ child solutions. Take the τ nearest solutions to the median from the database (based on Euclidean distance) and their function values (performances) as the training data points to construct a GP surrogate model.

Step 4: Compare the predicted value of each solution in C_i and the current best solution (simulated value). Add the number of solutions that are better than the current best solution to N_s (the number of successes of (4), (5) or (6)) and add λ to N_u (the number of uses of (4), (5) or (6)).

Until all the h groups of child solutions are generated.

Step 5: Update the rates of using DE/rand/1 (4), DE/current-to-best/1 (5) and DE/rand/2 (6) by N_s/N_u based on the previous L iterations. Update the number of iterations.

It can be seen that the DE mutation strategies are employed self-adaptively and the most frequently fitted will be used with a greater chance. Experimental results show that, for different types of functions and initial populations, the rates can be drastically different, which verifies the effectiveness of this operator. Experiments also show that, if without the improved initial population from Step 4 but with this operator, the success rate becomes low and some runs may cost many function evaluations. This shows the necessity to combine both operators.

E. Discussions on Surrogate Model Management in PAGHO

Surrogate model management is important in any SAEA. The surrogate model usage and management in PAGHO are summarized as follows: In each iteration, h GP models are built in Step 8 of the PAGHO framework (Section III (B)). In the first iteration, the initial surrogate model is built based on

the training data points obtained in Step 4 (the method is in Section III (C)) as well as the simulated points in the parallel SQP search process in Step 3. After that, in each iteration, h new training data points (top h solutions based on prediction) are supplemented by Step 9 of the PAGHO framework for updating the GP models in the next iteration. The goals of this surrogate model management method are to make SMAS [8] work harmonically with SQP and adapt to the small-scale parallel simulation environment.

The challenges to achieve the above two goals are: (1) Using Latin Hypercube sampling of the whole design space to obtain well distributed initial samples is essential for the success of SMAS, but this is in conflict with the key idea for efficiency improvement of PAGHO. PAGHO uses SQP to avoid surrogate modeling (i.e., samples based on computationally expensive simulations) of many non-optimal regions and well distributed initial samples are not straightforwardly available. (2) SMAS is a sequential method. Only a single candidate solution is simulated in each iteration and the surrogate model is then updated. In contrast, PAGHO aims to support parallel simulations of several candidate designs and the surrogate model is updated based on multiple points in each iteration.

To address the first challenge mentioned above, the initial population generation method starting from the local optima obtained by SQP is developed (Section III (C)). A scoring method is introduced aiming to select points which have good fitness values and far from existing selected points without additional simulations (if possible). The goal is to approach the well distributed initial training data points (but in the optimal region) as much as possible. To address the second challenge, three DE mutation strategies are carefully selected (Section III (D)). The aim is to provide solutions with good optimality and diverse distribution in the optimal region (instead of optimal but clustered together) in order to improve the surrogate model quality as much as possible. Our pilot experiments using 4 mathematical benchmark problems show that the selected combination performs the best among all the combinations of the 5 widely used DE mutation strategies (i.e., DE/best/1, DE/rand/1, DE/current-to-best/1, DE/best/2, DE/rand/2 [26]).

F. Parameter Settings

PAGHO introduces only the new parameter L beyond the SQP and SMAS parameters. The parameters in SQP and SMAS are investigated and those parameters are shown to be insensitive by experimental verifications. More details are provided in [24], [12]. Following the rules, we set $F = 0.8$, $CR = 0.8$, $\lambda = 50$, $\tau = 8 \times d$ (d is the number of design variables). In particular, λ is recommended to be set between $5\times d$ to $8\times d$ to obtain the stability of the performance. Clearly, the learning period L is not sensitive, which is recommended to be between 20 and 40 related to the dimensionality of the problem. For simplicity, 30 is used for the problems in Section IV.

The setting of the stopping criterion for SQP needs attention. The goal of SQP in PAGHO is to efficiently converge to solutions that are near to local optima rather than finding the exact local optima. Therefore, the tolerance of x and $f(x)$ is not recommended to be very small as with standard SQP. For example, if the expected improvement of $f(x)$ is less than 0.1 or the movement of x for the next step is within 0.01, then continued SQP optimization is not necessary. The tolerance values depend on the design problem and are easy to estimate by the designer. Clearly, this number is insensitive considering the goal of SQP in PAGHO.

IV. Experimental Results and Comparisons

In this section, PAGHO's effectiveness is demonstrated by experimental verification. Two complementary test cases are carefully selected to exhibit typical challenges that may be encountered. We suggest that, if PAGHO performs well for these two difficult test cases, it should also work well for easier problems, as was verified by our pilot experiments. For real-world problem tests, a silicon micro-actuator [36] is selected, which is modeled in COMSOL Multiphysics. The major challenge here is that the adjoint partial derivatives are inaccurate for the objective function due to the unavoidable numerical error, although reasonably accurate for the constraints. The purpose is to indicate the performance of PAGHO for design cases where the estimated derivative information is inaccurate. In addition, the constraints are tight. Thus, PAGHO's ability to handle tight constraints is also shown.

The mathematical benchmark problem test aims, instead, to study the property of PAGHO using known landscape characteristics and optimal solutions. The central challenge for PAGHO is its exploration ability. Therefore, in the constructed problem, the Levy function [37] serves as the objective function and the Ellipsoid function [37] serves as the constraint. The Levy function is a typical landscape in which to test the exploration ability of a global optimization method. The function has many local optima, which are relatively far away from each other and the global optimum is located in a flat valley. This landscape may make the parallel SQP optimization (Steps 1-3 of PAGHO) arrive at local optimal regions far from the global optimum, which causes the SAEA part (Steps 4-9) of PAGHO to be easily trapped in a local optimum or become slow. The purpose of this test case is to show PAGHO's ability to jump out of local optima. Note that some very rugged or discontinuous mathematical benchmark problems (e.g., the Rastrigin function) are not appropriate to be used because they are not fit for using partial derivatives and are rarely seen in actuator design landscapes.

PAGHO, a state-of-the-art SMAS-based SAEA [8], [12] and the standard DE algorithm [26] are compared. In [8], the reference SMAS-based SAEA shows significant speed improvement compared to several popular conservative and active SAEAs. [12] provides a minor improved version of [8], which is used as the reference method. DE, the Particle Swarm Optimization method and the Covariance Matrix Adaptation Evolution Strategy for engineering design optimization are compared in various papers, and generally comparable performance has been shown. Hence, DE is used to represent typical standard EAs.

The parameters of SMAS (λ,τ,F and CR) and the parameters of DE (F,CR) are the same as those used in

PAGHO. The number of initial samples of SMAS is set to λ, which is often used by SMAS-based SAEAs [12]. The penalty function method [27] is used to handle the constraints and the penalty coefficient is set to 50. The examples are run on a workstation with Intel Xeon CPU E5-2420 and 24 GB RAM. In parallel SQP, different starting points lead to different numbers of simulations (or function evaluations) and the largest one is used as the number of function evaluations in parallel SQP. A parallel simulation/function evaluation of several candidate solutions is straightforward for DE. SMAS, on the other hand, is a sequential method, and does not support parallel function evaluations except for the initial Latin Hypercube sampling [8], [12]. Hence, the number of simulations/evaluations shown in the following subsections is parallel simulations/evaluations for PAGHO and DE, while for SMAS, parallel simulations/evaluations are only applied to the 50 Latin Hypercube sampling points.

A. 7-variable silicon micro-actuator

The silicon (modulus of elasticity $E = 1.35 \times 10^5$ MPa, Poisson's ratio $\nu = 0.33$) micro-actuator is shown in Fig. 2. In this actuator, the design parameters for snap-through under particular conditions of pressure translate into a predetermined value for deflection. The membrane of the corrugated actuator is rigidly fixed along the external radius $w = 0, w' = 0$ (where w is a displacement along Z axis, w' is a rotation angle in ZOR plane) and has a rigid center ($w' = 0$). The corrugated membrane has a sinusoidal profile. For this membrane, the pressure is uniformly loaded. The micro-actuator can be described by equilibrium equations, geometry equations and Hooke's law [38], which are as follows:

$$\sigma_{ij,j} + \rho f_i = 0 \tag{17}$$

where σ_{ij} is the stress tensor components, ρ is the density and f_i is the loading vector components.

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i} + \sum_l u_{l,i}u_{l,j}) \tag{18}$$

where ε is the deformation tensor components, u is the displacement component.

$$\sigma_{ij} = \sum_{kl} C_{ijkl}\varepsilon_{kl} \tag{19}$$

where C is the elastic constants tensor components.

The design variables and their ranges are in Table I. The optimization problem is shown in (20).

$$\begin{aligned} &max \quad D_s@P_c \\ &s.t. P_c \in [2.575, 2.625]MPa \\ &r_2 \leq 580\mu m \end{aligned} \tag{20}$$

where D_s is the displacement, P_c is the critical pressure and r_2 is the radius.

Finite element analysis and adjoint sensitivity analysis are carried out by COMSOL Multiphysics. The axisymmetric element formulation is used. The cross-section of the shell is modeled using free triangular mesh and the average element size is $0.25 \times h_1$. The nonlinear analysis is performed using the global equation method. The meshing procedure and solution are performed for each new candidate design. Each simulation takes approximately 6-13 minutes, in which, the sensitivity analysis only takes less than 30 seconds. When using PAGHO, three simulations are performed simultaneously using the workstation. In parallel SQP, the tolerance of $f(x)$ is set to 1, which means that, if the expected improvement of displacement is smaller than $1\mu m$, SQP will terminate.

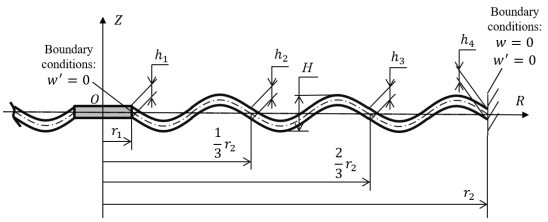

Figure 2. The silicon micro-actuator

Table I
RANGES OF DESIGN VARIABLES OF THE MICRO-ACTUATOR (IN μm)

$Variables$	H	r_1	r_2	h_1	h_2	h_3	h_4
Lower bound	65	30	550	6	6	6	6
Upper bound	85	50	650	7.5	7.5	7.5	7.5

The convergence trends of PAGHO and SMAS are shown in Fig. 3. 10 runs are carried out for PAGHO and SMAS. 1 run of DE is also carried out, costing 11 days. The computing budget of PAGHO is 80 parallel simulations and that for SMAS is 467 simulations (17 parallel simulations are used for the initial sampling and the other 450 simulations are not able to be parallelized in SMAS). Note that SMAS needs 50 initial samples (3 samples are simulated in parallel), so the convergence trend begins from the 18^{th} simulation.

In the 10 runs of PAGHO, one of them fails to get a feasible solution and all the others satisfy the design constraints. The average objective function value (excluding the infeasible solution) is $173.6\mu m$ with a standard deviation of $1.27\mu m$. In particular, 8 out of 10 runs obtain the best found value $174.0\mu m$. The average runtime is about 9 hours. In the 10

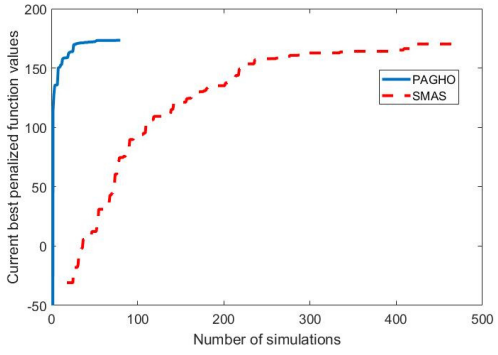

Figure 3. The convergence trends of PAGHO (average of 9 runs without the failed run) and SMAS (average of 10 runs) for test case 1

runs of SMAS, all of them find a feasible solution and the average objective function value is $170.2\mu m$ with a standard deviation of $2.40\mu m$. The average runtime is about 2.3 days. It can be seen that PAGHO is much more efficient than a state-of-the-art SMAS-based SAEA. The best found result of DE is also $174\mu m$, using 917 parallel simulations. To obtain the average result of PAGHO, DE needs 834 parallel simulations. Hence, PAGHO is 10 times faster than DE. If DE does not use parallel simulations, PAGHO is about 30 times faster.

As expected, thanks to the partial derivatives, PAGHO can often obtain a near feasible solution in the parallel SQP process using a very small number of simulations (maximum 19 simulations among the 10 runs), providing a much better starting region than SMAS, as is shown in Fig 3. It is reasonable that feasible solutions are not provided by SQP optimization due to the tight constraints and not very accurate partial derivatives. The SAEA part of PAGHO then jumps out of local optima and obtains high-quality solutions, verifying the effectiveness of key operators. It can also be seen that the inaccurate partial derivatives of the objective function do not have much impact, since only 1 out of the 10 runs fails.

To further verify PAGHO's optimization capacity and the key operators, two experiments are carried out. In the first experiment, 10 runs of SQP using randomly generated starting points are carried out. A small enough tolerance of $f(x)$, $0.01\mu m$, is used. In the 10 runs, only 1 run obtains a feasible design with a $160\mu m$ displacement, which is a local optimum compared to the results of DE and PAGHO, while all the others converge to infeasible local optimal points.

In the second experiment, the initial population generation method (Section III (C)) of PAGHO is not used. Instead, the top λ candidate designs visited in the parallel SQP process are used to construct the initial population. Then, SMAS is carried out with this new population. In this way, many non-optimal regions are escaped thanks to the SQP search but the population diversity is not considered. 10 runs are carried out. In the 10 runs, 3 runs fail to satisfy the constraints (i.e., converge to infeasible local optimal points), 2 runs obtain the best found result, $174.0\mu m$, and other runs converge to local optima in the feasible region. In all the feasible runs, the average objective function value is $170.4\mu m$ with a standard deviation of $6.28\mu m$. Compared to PAGHO results, for which, 8 out of 10 runs obtain the best found objective function value, the global exploration capacity of PAGHO, in particular, the proposed initial population generation method, is verified.

B. Benchmark Problem Test

To study the exploration ability of PAGHO, the constructed problem is shown in (21). The search range is $[-20, 20]^{10}$. The function landscape characteristics are described before. The global optimum is 0 when all the elements in x are 1. 4 solutions are evaluated simultaneously for PAGHO and DE in this experiment. The tolerances of $f(x)$ and x are set to 0.1.

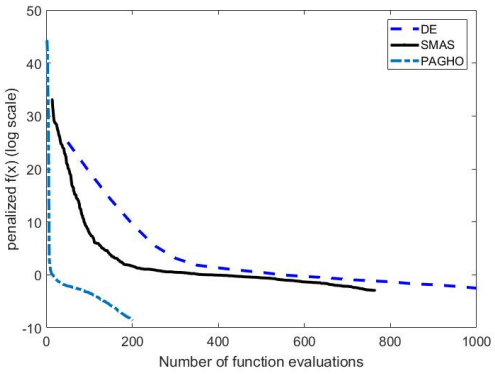

Figure 4. The convergence trends of PAGHO, SMAS and DE for test case 2 (average of 20 runs)

$$\begin{aligned} &min \quad f(x) = sin^2(\pi w_i) + \sum_{i=1}^{d-1}[1 + 10sin^2(\pi w_i + 1)] \\ &+(w_d - 1)^2[1 + sin^2(2\pi w_d)], w_i = 1 + 0.25(x_i - 1) \\ &s.t. g(x) = log(\sum_{i=1}^{d} i \times x_i^2) \leq 5 \\ &i = 1, \ldots, 10 \end{aligned} \tag{21}$$

20 runs are performed for PAGHO, SMAS and DE. The constraints are satisfied for all the algorithms. Fig. 4 shows the convergence trends. The computing budgets are 200 parallel function evaluations for PAGHO, 763 function evaluations for SMAS (13 parallel function evaluations are used for the initial sampling and the other 750 function evaluations are not able to be parallelized in SMAS) and 2500 parallel function evaluations for DE.

The following conclusions can be drawn: (1) The average $f(x)$ of PAGHO is at the 1e-3 level and the average Euclidean distance between the obtained optimal solutions and the real global optimum is 0.048. The high optimization quality of PAGHO is thus shown. (2) PAGHO shows significant speed improvement compared to SMAS. The SMAS curve shows that the convergence speed is very slow from 300 to 700 function evaluations. This can be explained as SMAS is "learning" the flat valley in this process. PAGHO, however, avoids this process. To obtain the result of PAGHO, DE needs 1463 parallel function evaluations, which means that PAGHO is 7 times faster. If DE does not use parallel evaluations as PAGHO, PAGHO is 29 times faster. (3) PAGHO works as expected. The parallel SQP part firstly provides a much better starting region than random sampling using very few function evaluations. The SQP part obtains feasible solutions in all the runs, and the local optima that it provides vary from 0.05 to 7.38. The SAEA part then jumps out of the local optima and obtains highly optimal solutions.

Note that this problem is difficult for SAEAs since the global optimum is located on a flat valley and the local optima are not near to it, making prediction difficult. For many other benchmark problems, PAGHO and SMAS show much greater speed enhancement than DE. The high exploration ability of PAGHO is shown by this example, justifying the initial population generation method and the self-adaptive child solution generation method.

V. Conclusions

In this paper, the PAGHO algorithm has been proposed for actuator shape design optimization, aiming to substantially improve the optimization efficiency. In contrast with existing works using derivative information for local refinement in SAEAs, the new concept of utilizing derivative information to escape non-optimal regions beforehand and then making SAEA work harmonically from the not well distributed local optimization results, is introduced for the first time, to the best of our knowledge.

Experiments show that even for challenging landscapes or inaccurate estimated derivative information, PAGHO can provide highly optimal designs, which are comparable to the DE algorithm, but uses a reasonable timeframe and is much more efficient. Moreover, PAGHO shows several times speed improvement over a state-of-the-art SMAS-based SAEA. This is achieved by our novel adjoint sensitivity-assisted SAEA framework, including the new hybrid structure, the initial SAEA population generation method and the self-adaptive child solution generation method. The limitation of PAGHO is that it is not fit for problems for which derivative information can hardly help. On the other hand, considering the wide applicability of derivative-based optimization in actuator design (although highly likely to only obtain an unsatisfactory local optimum), the scope of PAGHO is large. Future works will include deeper behavioral study and wider applications of PAGHO.

Acknowledgement

The authors would like to thank Dr. Edmund Dickinson, COMSOL Ltd, for valuable discussions.

References

[1] P. Di Barba and S. Wiak, "Evolutionary computing and optimal design of MEMS," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 4, pp. 1160–1167, 2015.

[2] Z. Fan, J. Liu, T. Sorensen, and P. Wang, "Improved differential evolution based on stochastic ranking for robust layout synthesis of MEMS components," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 4, pp. 937–948, 2009.

[3] J. K. Coultate, C. H. Fox, S. McWilliam, and A. R. Malvern, "Application of optimal and robust design methods to a MEMS accelerometer," *Sensors and Actuators A: Physical*, vol. 142, no. 1, pp. 88–96, 2008.

[4] B. Liu and A. Nikolaeva, "Efficient global optimization of MEMS based on surrogate model assisted evolutionary algorithm," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 555–558.

[5] Y. Lee, Y. Park, F. Niu, and D. Filipovic, "Design and optimisation of one-port RF MEMS resonators and related integrated circuits using ANN-based macromodelling approach," *IEE Proceedings-Circuits, Devices and Systems*, vol. 153, no. 5, pp. 480–488, 2006.

[6] R. Datta and R. G. Regis, "A surrogate-assisted evolution strategy for constrained multi-objective optimization," *Expert Systems with Applications*, vol. 57, pp. 270–284, 2016.

[7] 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.

[8] B. Liu, Q. Zhang, and G. Gielen, "A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 180–192, 2014.

[9] 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.

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

[11] 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.

[12] B. Liu, Q. Chen, Q. Zhang, G. Gielen, and V. Grout, "Behavioral study of the surrogate model-aware evolutionary search framework," in *IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 715–722.

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

[14] B. Liu, H. Aliakbarian, Z. Ma, G. A. Vandenbosch, G. Gielen, and P. Excell, "An efficient method for antenna design optimization based on evolutionary computation and machine learning techniques," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 1, pp. 7–18, 2014.

[15] B. Liu, D. Zhao, P. Reynaert, and G. G. Gielen, "GASPAD: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 2, pp. 169–182, 2014.

[16] B. Liu, S. Koziel, and Q. Zhang, "A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems," *Journal of Computational Science*, vol. 12, pp. 28–37, 2016.

[17] B. Liu, S. Koziel, and N. Ali, "SADEA-II: A generalized method for efficient global optimization of antenna design," *Journal of Computational Design and Engineering*, vol. 4, no. 2, pp. 86–97, 2017.

[18] J. Lee, E. M. Dede, and T. Nomura, "Simultaneous design optimization of permanent magnet, coils, and ferromagnetic material in actuators," *IEEE Transactions on Magnetics*, vol. 47, no. 12, pp. 4712–4716, 2011.

[19] S. Director and R. Rohrer, "The generalized adjoint network and network sensitivities," *IEEE Transactions on Circuit Theory*, vol. 16, no. 3, pp. 318–323, 1969.

[20] Y.-S. Chung, C. Cheon, I.-H. Park, and S.-Y. Hahn, "Optimal design method for microwave device using time domain method and design sensitivity analysis. II. FDTD case," *IEEE Transactions on Magnetics*, vol. 37, no. 5, pp. 3255–3259, 2001.

[21] S. Koziel, S. Ogurtsov, Q. S. Cheng, and J. W. Bandler, "Rapid electromagnetic-based microwave design optimisation exploiting shape-preserving response prediction and adjoint sensitivities," *IET Microwaves, Antennas & Propagation*, vol. 8, no. 10, pp. 775–781, 2014.

[22] COMSOL, "COMSOL multiphysics user's guide," *COMSOL Multiphysics Inc.*

[23] CST, "Studio-workflow and solver overview," *CST-Computer Simulation Technology AG*, 2016.

[24] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995.

[25] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

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

[27] D. M. Himmelblau, *Applied nonlinear programming*. McGraw-Hill Companies, 1972.

[28] T. J. Santner, B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*. Springer Science & Business Media, 2013.

[29] I. Couckuyt, T. Dhaene, and P. Demeester, "ooDACE toolbox: a flexible object-oriented kriging implementation." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3183–3186, 2014.

[30] S. Koziel, S. Ogurtsov, J. W. Bandler, and Q. S. Cheng, "Reliable space-mapping optimization integrated with EM-based adjoint sensitivities," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 10, pp. 3493–3502, 2013.

[31] W. E. Hart, N. Krasnogor, and J. E. Smith, *Recent advances in memetic algorithms*. Springer Science & Business Media, 2004, vol. 166.

[32] B. Liu, H. Yang, and M. J. Lancaster, "Global optimization of microwave filters based on a surrogate model-assisted evolutionary algorithm," *IEEE Transactions on Microwave Theory and Techniques*, 2017.

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

[34] S. Y. Kung, *Kernel methods and machine learning*. Cambridge University Press, 2014.

[35] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of Genetic Algorithms*, vol. 1, pp. 69–93, 1991.

[36] J. Han and D. P. Neikirk, "Deflection behavior of fabry-perot pressure sensors having planar and corrugated membrane," in *Micromachining and Microfabrication'96*. International Society for Optics and Photonics, 1996, pp. 79–90.
[37] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.
[38] G. T. Mase, R. E. Smelser, and G. E. Mase, *Continuum mechanics for engineers*. CRC press, 2009.

Bo Liu (M'15-SM'17) 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 University of Leuven (KU Leuven), Belgium, in 2012. From 2012 to 2013, he was a Humboldt research fellow and was working with Technical University of Dortmund, Germany. In 2013, he was appointed lecturer at Wrexham Glyndwr University, UK, where he was promoted to Reader in Computer-aided Design in 2016. He is an honorary fellow at The University of Birmingham. His research interests lie in design automation methodologies of analog/RF integrated circuits, microwave devices, MEMS, evolutionary computation and machine learning. He has authored or coauthored 1 book and more than 40 papers in international journals, edited books and conference proceedings.

Vic Grout (M'01-SM'05) has a BSc(Hons) in Mathematics and Computing from the University of Exeter and a PhD in Communication Engineering (Thesis title: Optimisation Techniques for Telecommunication Networks) from Plymouth Polytechnic. He is currently Professor of Computing Futures at Wrexham Glyndwr University, Wales; having been previously Professor of Network Algorithms, Head of Computing, Associate Dean for Research and Director of the Centre for Applied Internet Research. He is an approved British Computer Society accreditation assessor, an Institute of Engineering and Technology recommended speaker and a European Commission 'Horizon 2020' Expert Research Assessor and Ethics consultant. Vic has worked in senior positions in academia and industry for nearly 30 years and has published over 350 research papers, patents and books. His research interests span several areas of computational mathematics, including artificial intelligence and the application of heuristic principles to large-scale problems in Internet design, modelling, simulation, management and control.

Anna Nikolaeva received the M.Sc. degree from Bauman Moscow State Technical University, Russia, in 2012. She received her Candidate of Science degree from Bauman Moscow State Technical University, Russia, in 2016. She is currently a Ph.D. student at Glyndwr University, U.K. Her research interests lie in optimization techniques for the structural design of MEMS devices, nonlinear deformation problems and post-buckling analysis.