Multi-Objective Optimization for High-Dimensional Expensively Constrained Black-Box Problems

George H. Cheng
Product Design and Optimization Laboratory (PDOL),
Simon Fraser University,
Surrey, BC, V3T 0A3, Canada
e-mail: ghc2@sfu.ca

G. Gary Wang[1]
Product Design and Optimization Laboratory (PDOL),
Simon Fraser University,
Surrey, BC, V3T 0A3, Canada
e-mail: gwa5@sfu.ca, gary_wang@sfu.ca

Yeong-Maw Hwang
Metal Forming Technology Laboratory,
National Sun Yat-sen University,
Kaohsiung 80424, Taiwan
e-mail: ymhwang@mail.nsysu.edu.tw

Multi-objective optimization (MOO) problems with computationally expensive constraints are commonly seen in real-world engineering design. However, metamodel-based design optimization (MBDO) approaches for MOO are often not suitable for high-dimensional problems and often do not support expensive constraints. In this work, the situational adaptive Kreisselmeier and Steinhauser (SAKS) method was combined with a new multi-objective trust region optimizer (MTRO) strategy to form the SAKS-MTRO method for MOO problems with expensive black-box constraint functions. The SAKS method is an approach that hybridizes the modeling and aggregation of expensive constraints and adds an adaptive strategy to control the level of hybridization. The MTRO strategy uses a combination of objective decomposition and K-means clustering to handle MOO problems. SAKS-MTRO was benchmarked against four popular multi-objective optimizers and demonstrated superior performance on average. SAKS-MTRO was also applied to optimize the design of a semiconductor substrate and the design of an industrial recessed impeller.
[DOI: 10.1115/1.4050749]

Keywords: approximation-based optimal design, design optimization, metamodeling, multidisciplinary design and optimization, multi-objective optimization, structural optimization

Introduction

Computer simulations are widely used in engineering design as a form of digital prototyping, and optimization can be used in conjunction with simulation models to solve design challenges and constraints. However, these simulation models are typically computationally expensive, which make many optimization methods not viable because they require tens to hundreds of thousands of simulation calls. For example, Pérez et al. in 2016 simulated the heat distribution of an engine room in an advanced electrical propulsion ship using computational fluid dynamics (CFD), which took 5.5 h per simulation [1] and Díaz-Ovalle in 2017 used CFD to simulate a convection oven, which took up to five hours per simulation [2]; 10,000 runs of such models require 5 years of run time to complete, which is simply infeasible.

Finite element analysis (FEA) and CFD simulations are considered black-box, where the inner structure is hidden to the user or external programs and where gradients are not readily available or unreliable. This makes classical gradient-based optimization methods very costly or unsuitable to use since the gradient must be approximated and the cost of gradient approximation increases dramatically with the number of variables and objectives. Furthermore, problems of higher dimensionality (i.e., 10 variables or greater) have exponentially larger search spaces [3,4], which increases the difficulty of optimization. The combination of the above traits forms a class of problems that are high-dimensional, expensive (computationally), and black-box (High-dimensional, Expensive, Black-box (HEB)) [5].

Figure 1 presents an overview of black-box multi-objective optimization methods, which can be approximately segmented between evolutionary and metamodel-based methods. Because the optimum for multi-objective optimization (MOO) problems is usually a set of designs, evolutionary algorithms (EAs) such as NSGA [6], NPGA [7], and NSGA-II [8] have proved to be well suited to multi-objective problems since they natively work with populations of solutions. NSGA-II is a genetic algorithm that incorporates fast nondominated sorting and crowding-distance-based Pareto set diversity preservation. The nondominated sorting method alleviates some of the computational inefficiencies of prior EA algorithms, while the diversity preservation method enables NSGA-II to evolve evenly distributed Pareto sets. According to Zhou et al., NSGA-II is a framework that the majority of Multi-Objective EAs (MOEAs) are based on [9]. Many other types of MOEAs exist such as MOEA/D [10,11] that natively decompose the MOO problem into scalar optimization subproblems, the Indicator-based Evolutionary Algorithm (IBEA) [12,13] that uses an indicator such as generational distance or hypervolume to compare candidate designs, and Multi-Objective Feasibility Enhanced Particle Swarm Optimization (MOFEPSO) [14] that maintains separate feasible and

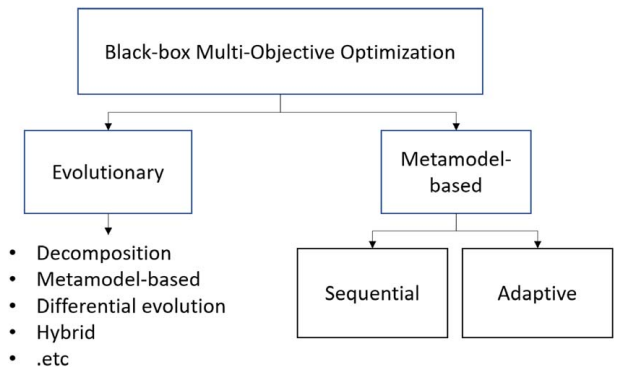

Fig. 1 Overview of black-box multi-objective optimization approaches

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received November 30, 2020; final manuscript received March 29, 2021; published online May 28, 2021. Assoc. Editor: Ping Zhu.

infeasible particles to handle constrained optimization problems. Metamodel-based evolutionary methods are becoming more common in recent years to address the issue of expensive black-box problems [15]. For example, MOEA/D was extended using a radial basis function (RBF) in Ref. [16] and NSGA-II was extended using Kriging in Ref. [17], among many other methods [18,19]. While a large body of work is present in evolutionary optimization, Chugh et. al. noted that there was a lack of algorithms that "considered constraints besides the bounds for the decision variables" [15].

Another class of MOO methods is non-nature metamodel-based methods, where approximation models such as RBF and Kriging are used to approximate computationally expensive black-box models or the Pareto Frontier directly. From Tabatabaei et al. [20], metamodel-based MOO methods are generally classified as either sequential or adaptive. Sequential methods focus on the construction of accurate metamodels, which are then used to solve for the Pareto frontier. Wilson et al. developed a sampling-based method where the expensive models are approximated using a second-order polynomial and a Kriging model, which are made accurate via an iterative process of sampling [21]. They then use a large number of samples to find the Pareto frontier using the approximation models. Su et al. hybridized a polynomial response surface model and a Gaussian RBF to create the hybrid RBF (HRBF) model, which they used to approximate a noisy crashworthiness model [22]. They then applied an MOEA on the HRBF model to solve for the Pareto frontier. Adaptive methods, on the other hand, improve the accuracy of the metamodels throughout the optimization process by iteratively reconstructing the metamodel using newly obtained design points. Yang et al. combined a single-objective optimizer and a multi-objective optimizer to solve for the Pareto frontier using Kriging metamodels [23]. The single-objective method helps to find the extreme points on the Pareto frontier, while the multi-objective optimizer helps to develop the entire frontier. Shan and Wang developed the PSP method which uses a quadratic polynomial and an RBF to approximate the expensive functions [24]. A discriminative sampling process is used in two stages to bias the sampling towards the Pareto frontier as well as the extreme points. Although there have been more metamodel-based MOO methods in recent years, they have overwhelmingly focused on low-dimensional optimization and inexpensive constraints [20].

Survey papers such as Refs. [9,15,20,25] can be consulted for detailed reviews of the relevant literature. The motivation of this work is to develop a non-nature metamodel-based MOO method designed to handle expensive objectives and inequality constraints as well as high numbers of design variables, which is an important area of research. We aim to solve problems with the following mathematical formulation

$$\begin{aligned} &\min_{x} \boldsymbol{F}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), \ldots, f_i(\boldsymbol{x})\} \\ &s.t.\ g_j(\boldsymbol{x}) \le 0 \\ &i = 1, \ldots, m; j = 1, \ldots, p \end{aligned} \tag{1}$$

where x are the design variables; $\boldsymbol{F}$ is the set of m objective functions; and g_j are expensive inequality constraints. The constraint handling method used is the situational adaptive Kreisselmeier and Steinhauser (SAKS) method [26], which uses an adaptive enveloping function to avoid modeling all constraints using the surrogate. The SAKS method is combined with a new MOO global optimization strategy that uses a combination of objective function decomposition and *K*-means clustering in a trust region framework to form the SAKS-multi-objective trust region optimizer (SAKS-MTRO), an adaptive metamodel-based approach. SAKS-MTRO is then compared to a set of popular MOO methods on a suite of benchmark functions with inequality constraints and applied to the design of a semiconductor substrate and an industrial recessed impeller. This work demonstrates that our method can perform well on both unconstrained and constrained MOO problems without changing the code or hyperparameters.

Radial Basis Function Surrogate

The chosen surrogate method for this work is an RBF composed of a sum of thin-plate splines and a linear polynomial based on success with prior work [26,27]

$$\begin{aligned} &\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \beta_i |\boldsymbol{x} - \boldsymbol{x}_i|^2 \log|\boldsymbol{x} - \boldsymbol{x}_i| + P(\boldsymbol{x}) \\ &\sum_{i=1}^{n} \beta_i \boldsymbol{p}(\boldsymbol{x}) = 0, \quad P(\boldsymbol{x}) = \boldsymbol{p}\alpha = [p_1, p_2, \ldots, p_q][\alpha_1, \alpha_2, \ldots, \alpha_q]^T \end{aligned} \tag{2}$$

where $\boldsymbol{x}_i$ are the evaluated n number of center points; and $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are the resultant coefficients of the model fitting process. $P(\boldsymbol{x})$ is the linear polynomial where $q = d + 1$, with d being the number of variables.

Review of the SAKS Method

The SAKS method is a strategy to intelligently aggregate black-box inequality constraints to reduce the number of constraints being modeled. The method leverages the modified KS function, developed by Raspanti et al. [28] based on the original work by Kreisselmeier and Steinhauser [29], for function aggregation and employs an adaptive strategy to determine which constraints to aggregate during the optimization process. The modified KS function is a continuous aggregation function with the following formulation:

$$KS[\boldsymbol{g}(\boldsymbol{x})] = \boldsymbol{g}_{\max}(\boldsymbol{x}) + \frac{1}{\rho} \ln\left[\sum_{i=1}^{n} e^{\rho(g_i(\boldsymbol{x}) - \boldsymbol{g}_{\max}(\boldsymbol{x}))}\right] \tag{3}$$

where $\boldsymbol{g}(\boldsymbol{x})$ are the constraint values at $\boldsymbol{x}$ design points, $\boldsymbol{g}_{\max}(\boldsymbol{x})$ is the maximum constraint value at design point $\boldsymbol{x}$, and ρ is a shape parameter. KS is a conservative envelope function that tends to stay above the maximum constraint value for each design point. ρ controls the degree of conservatism of the KS function, with a smaller ρ generating more conservative estimates. The KS function can conservatively envelope the feasible space and aggregate any number of constraints into one function [30].

The SAKS method uses an RBF surrogate to model the KS function to construct aggregated constraint functions that reduce the number of surrogate models required. Because the KS enveloping function is smooth and conservative, it is suitable for use as a constraint aggregator and helps to compensate for surrogate modeling inaccuracies.

SAKS hybridizes the two strategies of individual constraint modeling and constraint aggregation. In each iteration, SAKS classifies constraints for either individual modeling or aggregation, based on the constraint violation history of each constraint function as follows:

$$\begin{aligned} &\{\boldsymbol{g}_{ind}\}_1 = \{\boldsymbol{g}\}, \quad \{\boldsymbol{g}_{agg}\}_1 = \varnothing \\ &\{\boldsymbol{g}_{ind}\}_i = \{\boldsymbol{g}_{viol}\}_{i-l,i-l+1,\ldots,i-1} \in \{\boldsymbol{g}\} \\ &\{\boldsymbol{g}_{agg}\}_i = \{\boldsymbol{g}\} \backslash \{\boldsymbol{g}_{ind}\}_i \end{aligned} \tag{4}$$

where $\{\boldsymbol{g}\}$ is the full set of expensive inequality constraints; $\{\boldsymbol{g}_{ind}\}_i$ is the set of expensive inequality constraints that are to be individually modeled in the ith iteration, $\{\boldsymbol{g}_{viol}\}_{i-l,i-l+1,\ldots,i-1}$ is the set of violated expensive inequality constraints for the last l iterations, and $\{\boldsymbol{g}_{agg}\}_i$ is the set of expensive inequality constraints that are to be aggregated in the ith iteration. This classification strategy biases the algorithm toward individually modeling constraints at the beginning and transitioning to aggregation bias as the optimization progresses.

Next, the set $\{\hat{\boldsymbol{c}}\}_i$ of RBF surrogates is constructed at the ith iteration where $\{\hat{\boldsymbol{c}}_{ind}\}_i$ is the set of individually modeled constraints and $\hat{\boldsymbol{c}}^i_{agg}$ is the RBF surrogate of the KS aggregate of $\{\boldsymbol{g}_{agg}\}_i$.

$\{\boldsymbol{g}_{agg}(\boldsymbol{x})\}_i$, which represents the constraint values of $\{\boldsymbol{g}_{agg}\}_i$ at $\boldsymbol{x}$, is normalized to prevent bias

$$\boldsymbol{g}^j_{agg,norm}(\boldsymbol{x}) = \begin{cases} \text{normalized to } [-1, 0], & \text{if } \boldsymbol{g}^j_{agg}(\boldsymbol{x}) \le 0 \\ \text{normalized to } (0, 1], & \text{if } \boldsymbol{g}^j_{agg}(\boldsymbol{x}) > 0 \end{cases} \tag{5}$$

where $\boldsymbol{g}^j_{agg}(\boldsymbol{x})$ is the output of the jth function in $\{\boldsymbol{g}_{agg}\}_i$. $\{\boldsymbol{g}_{agg,norm}(\boldsymbol{x})\}$ is the set of outputs for all $\{\boldsymbol{g}_{agg}\}_i$, which is then aggregated into a single function using the KS method. Equation (3) can then be modified to

$$KS[\{\boldsymbol{g}_{agg,norm}(\boldsymbol{x})\}] = \boldsymbol{g}_{\max}(\boldsymbol{x}) + \frac{1}{\rho}\ln\left[\sum_{j=1}^{n} e^{\rho(\boldsymbol{g}^j_{agg,norm}(\boldsymbol{x}) - \boldsymbol{g}_{max}(\boldsymbol{x}))}\right] \tag{6}$$

$$\boldsymbol{g}_{\max}(\boldsymbol{x}) = \max(\{\boldsymbol{g}_{agg,norm}(\boldsymbol{x})\})$$

$KS[\{\boldsymbol{g}_{agg,norm}(\boldsymbol{x})\}]$ is then used to construct $\hat{c}^j_{agg}$ based on Eq. (6). ρ is computed at each iteration as follows:

$$\rho^{k+1} = \begin{cases} 2 \cdot \rho^k & \text{if } \{\boldsymbol{g}(\boldsymbol{x}^k)\} \le 0 \text{ or } \boldsymbol{x}^k = \varnothing \\ 0.5 \cdot \rho^k & \text{if any } \{\boldsymbol{g}(\boldsymbol{x}^k)\} > 0 \end{cases} \tag{7}$$

$$1 \le \rho \le 8192$$

where k is the current iteration and $\boldsymbol{x}^{k-1}$ is the previous iteration's candidate design. The second term in Eq. (6) is set to zero when the value of ρ reaches the maximum value. For further details on the SAKS method, refer to Ref. [26].

MTRO Methodology

The MTRO method adds two strategies that build on the dual trust region strategy introduced in TRMPS [31], which help to balance exploitation and exploration in a single-objective optimization. The two new strategies extend the dual trust region strategy into the multi-objective space by adding two trust regions to form two pairs of trust regions. One pair of trust regions (T_{A1} and T_{A2}) focus on exploitation and uses the first strategy, Random Objective Decomposition (ROD), which concentrates on extreme point generation. The other pair (T_{B1} and T_{B2}) concentrates on exploration and uses the second strategy, K-Means Opposition Search (K-Opp), which advances the overall frontier. Each pair of trust regions has the same radius but can have different centroids.

Random Objective Decomposition. ROD is used in the exploitation trust regions and employs two different decomposition approaches. The first approach focuses on extreme point generation using single-objective optimization. Given m objectives, ROD randomly selects an objective o_i at each iteration. The design with minimum value in o_i in the current frontier F_c is selected as the centroid for T_{A1} for the current iteration. Sampling and selection of candidate designs are focused on achieving lower values in o_i. In addition, a small distance-based penalty is applied to prevent new designs from being too close to existing designs, which has been observed to create overly-concentrated pockets of designs on the frontier. Search using the first approach can be expressed as

$$\begin{aligned} &\min[(1 - w_p)\boldsymbol{f}^*_{oi} + w_p(1 - \boldsymbol{d})] \\ &d_j = \min(\|\boldsymbol{x}^*_j - \boldsymbol{x}\|) \end{aligned} \tag{8}$$

where $w_p = 0.1$ is the penalty weight, $\boldsymbol{f}^*_{oi}$ are the normalized surrogate predicted o_i objective values, $\boldsymbol{d}$ are the normalized minimal Euclidian distances between candidate designs and existing designs, d_j is the jth design in $\boldsymbol{d}$, $\boldsymbol{x}^*_j$ is the location vector of the jth candidate design, and $\boldsymbol{x}$ is the location matrix of all existing designs. The value for w_p was chosen because of good performance after testing several values between 0 and 1. Further examination of the effect of w_p on performance should be done in the future.

The second approach enables targeted frontier exploitation by combining all objectives using randomized weights into a single-objective problem. The approach used is inspired by the weighted sum approach as applied in MOEA/D [10]. At each iteration, a set of weights $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$, where m is the number of objectives, is randomly generated such that $\lambda_i \ge 0$ and $\sum \boldsymbol{\lambda} = 1$. A new objective $o_\lambda = \sum_{i=1}^{m} \lambda_i o_i$ is formed and the design with minimum value in o_λ in the current frontier F_c is selected as the centroid for T_{A2} for the current iteration. Search using this second approach can be expressed as

$$\min\left(\sum_{i=1}^{m}\sum_{j=1}^{n} \lambda_i f^*_{i,j}\right) \tag{9}$$

where $f^*_{i,j}$ is the normalized surrogate predicted value at the ith objective and jth candidate design.

K-Means Opposition Search. K-Opp is used in the exploration trust regions and advances the frontier iteratively. It uses the k-means clustering method [32] and opposition-based learning (OBL) [33] to choose centroids for T_{B1} and T_{B2} in a randomized and unbiased manner, and the G function [34] is used to select candidate designs for evaluation with the black-box functions.

K-means is a popular clustering method that partitions a set of designs into clusters such that designs are in clusters with the nearest mean, as calculated using the Euclidian distance. The method first initializes the coordinates of the centroids of K clusters, where K is user-defined. Then, the method iteratively assigns points to their closest clusters and re-computes the cluster centroids until the centroids exhibit minimal change. This process can be expressed as the minimization of the sum of squared error (SSE) within clusters [35]

$$\min \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{c}_i\|^2 \tag{10}$$

where C is the set of all clusters, and $\boldsymbol{c}_i$ is the centroid of cluster C_i. The implementation of k-means used for this work is MATLAB's k-means clustering which, in addition to the standard iterative process, also uses the k-means++ method [36] to initialize cluster centroids.

K-Opp uses k-means to select the trust region centroid of T_{B1} from the designs in the current frontier. The frontier designs are clustered into K clusters

$$K = \begin{cases} \left\lceil \dfrac{N_f}{2} \right\rceil & \text{if } N_f < 2K_{\max} \\ K_{\max} & \text{otherwise} \end{cases} \tag{11}$$

where N_f is the number of frontier points and $K_{\max} = 50$ is the maximum number of clusters. A random cluster is chosen, and a random design is selected from within the cluster to be the trust region centroid. This method discretizes the frontier into more uniform clusters and thus minimizes bias toward heavily concentrated areas of the frontier in the process of selecting the trust region centroid. A design chosen at random would favor more concentrated areas of the frontier, which means new designs are also more likely to be sampled in those areas.

While k-means is used for within-region centroid selection, the concept of opposition is used to select the centroid of T_{B2} in relation to T_{B1}. OBL is a theory of sampling where the opposites of a set of candidates are also sampled [37]. These opposite candidates are defined in relation to the bounded search space. From Fig. 2, given a point P with coordinates (x_1, x_2), it's opposite point $\check{P}$ has coordinates $(\check{x}_1, \check{x}_2)$ where $\check{x} = a + b - x$. a and b represent the lower and upper bounds, respectively, of the search space for the given dimension.

The objective for OBL-based sampling methods is to increase the probability of sampling closer to the optimum. However, for

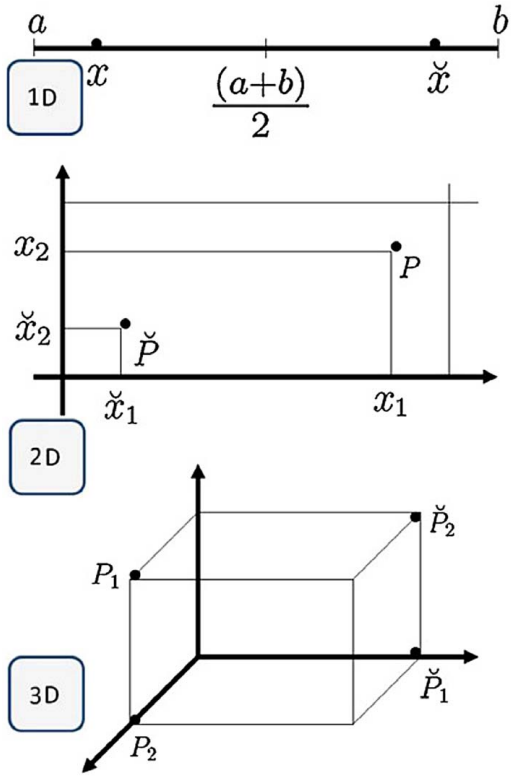

Fig. 2 Illustration of opposition-based sampling [37] (Reprinted with permission from Elsevier © 2012)

K-Opp, OBL is used to reduce sampling bias and maintain more uniform coverage of the search space by T_{B1} and T_{B2}. Using opposition, the centroid for T_{B2} is a design selected from the frontier F_c that is closest to the opposite of the centroid for T_{B1}.

The search is guided in each of the trust regions using the G function, as defined below

$$G_i = [1 - \max_{j \neq i} (\min (f_{i1}^* - f_{j1}^*, f_{i2}^* - f_{j2}^*, \ldots, f_{im}^* - f_{jm}^*))]^s \quad (12)$$

where G_i represents the fitness of the ith candidate relative to the other candidates, f_{im}^* is the normalized surrogate predicted kth objective value of the ith candidate for $k = 1, \ldots, m$, and l is a frontier exponent. $s = 1$, as defined in PSP [24]. Values of G range between [0, 2], where values below 1 predict dominated designs and values at or above 1 predict nondominated frontier designs. The goal of the search in T_{B1} and T_{B2} is to select candidate designs that have G values above 1 to iteratively push the frontier toward the Pareto frontier. In each iteration and for each trust region, after drawing candidate designs $\{x^*\}$ from the trust region, predicted objective values are generated for those designs using the RBF surrogate model and normalized to form the set $\{f^*\}$. $\{f^*\}$ is combined with the set of normalized frontier designs $F_{c\text{-}norm}$ and evaluated using the G function. The design with maximal G value that has a value above 1 is selected for evaluation with the black-box. This method maximizes the probability of advancing the frontier.

Effect of Random Objective Decomposition and *K*-Opp. Figure 3 shows the contribution of the ROD and *K*-Opp strategies to the frontier during optimization of a constrained two-objective problem, with points contributed by ROD represented by circles and those contributed by *K*-Opp represented by squares. The left plot shows the frontier after 100 function evaluations, with many points on the extremes of the frontier contributed by ROD. At the 200 function evaluations mark, the right plot shows the *K*-Opp contribution of filling in and gradually advancing the frontier. These plots show that ROD and *K*-Opp work effectively in tandem, with ROD aggressively expanding the frontier via extreme point generation early on and *K*-Opp advancing the entire frontier.

SAKS-MTRO

This work augments the SAKS-TRO algorithm with the MTRO strategy to form the SAKS-MTRO multi-objective optimizer. SAKS-MTRO has a total of four trust regions during regular operation compared to the two in SAKS-TRO. The four trust regions divided into exploitation and exploration trust regions allow the balancing of extreme points generation with targeted and broad frontier advancement. When there are no feasible solutions, SAKS-MTRO operates exactly like SAKS-TRO with two trust regions. Unlike TRMPS and SAKS-TRO which maintain a single centroid, in SAKS-MTRO, each trust region maintains its own centroid while the trust region radius for each pair of regions is identical. This enables the trust regions to be more distributed in the search space and enable broader advancement of the frontier. The trust region radii update procedure has also been modified to use frontier advancement as the criterion for assessing iterative progress.

In a major departure from TRMPS and SAKS-TRO, SAKS-MTRO uses all available expensive designs to construct

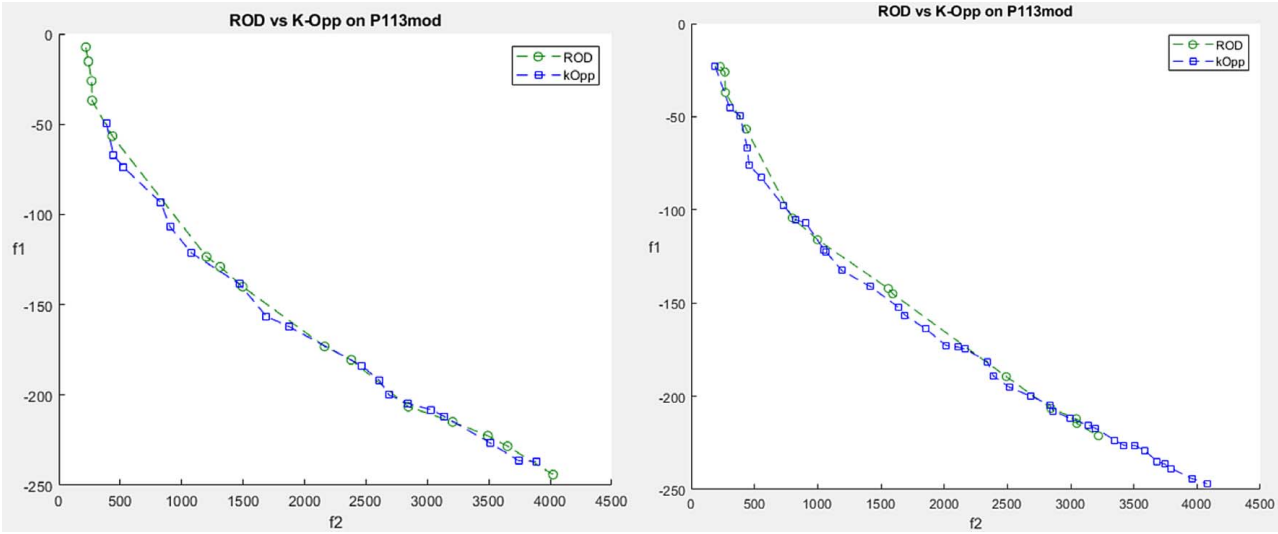

Fig. 3 Contribution of ROD and *K*-Opp after 100 designs (left) and 200 designs (right)

RBF surrogates. This is due to the difference in nature between single and multi-objective problems. MOO seeks broad and even frontier advancement, which is very different from the focused nature of single-objective optimization. Thus, constructing RBF surrogates using all available designs improves the modeling accuracy over the entire search space.

SAKS-MTRO Algorithm

Here, we follow the nomenclature established in Ref. [26] with several differences. We define sets of points $\{y\}$, $\{y_{A1}\}$, $\{y_{A2}\}$, $\{y_{B1}\}$, and $\{y_{B2}\}$, where $\{y_{A1}\}$, $\{y_{A2}\}$, $\{y_{B1}\}$, $\{y_{B2}\} \subseteq \{y\}$ and overlaps are allowed between sets. Define the best set of designs of the ith iteration as frontier F_i, and ρ_A^i and ρ_B^i are the conservativeness parameters for the pairs of exploitation and exploration trust regions at the ith iteration, respectively.

Step 1: Initial Sampling

- Sample n_{init} uniform random points $\{y\}$ in the design spce, and evaluate $\{y\}$ with the black-box function.
- For each point in $\{y\}$, place it in every trust region that it fits in, and place it in frontier F_i if it is feasible and nondominated.
- Set $\rho_S^1 = \rho_B^1 = 50$.

Step 2: Constraint Classification

- Classify the expensive constraint functions according to Eq. (4) using the constraint function values of $\{y\}$.

Step 3: Constraint Surrogates Construction

- Construct the RBF surrogates $\{\hat{c}_{ind}\}_i$ according to Eq. (2) using expensive points $\{y_{A1}\}$ in T_{A1}. Construct the aggregate RBF surrogate $\hat{c}_{agg}^i$ by first aggregating constraint values using Eqs. (5) and (6), then building the RBF model with Eq. (2) and expensive points $\{y_{A1}\}$.
- Repeat for T_{A2}.

Step 4: Mode Selection

- If a feasible point is present, go to *Step 5*, else go to *Step 7*.

Step 5: Surrogate Point Accumulation

- Set the centroids of T_{A1} and T_{A2} according to the ROD strategy. Sample a relatively large number of uniform random points, e.g., $n_0 = 5000$, for each region T_{A1} and T_{A2}. Fit the points that satisfy the cheap constraints onto the corresponding $\{\hat{c}_{ind}\}_i$ and $\hat{c}_{agg}^i$ surrogates. The points that satisfy the constraint surrogates $\{\hat{c}_{ind}\}_i$ and $\hat{c}_{agg}^i$ are added to the point sets $\{z_{A1}\}$ and $\{z_{A2}\}$. This process is repeated until a good number of (e.g., 500) feasible samples have been accumulated in each set.

Step 6: Objective Surrogate Construction

- Construct the RBF surrogates $\hat{f}$ of the objective function using Eq. (2) and expensive points $\{y_{A1}\}$ and $\{y_{A2}\}$.
- Go to *Step 9a*.

Step 7: Constraint-penalizing Merit Function

- Set the centroid of T_{A1} as the design in $\{y_{A1}\}$ that is closest to feasibility. Sample $n_0 = 5000$ uniform random points in T_{A1} and fit the ones that satisfy the cheap constraints onto the $\{\hat{c}_{ind}\}_i$ and $\hat{c}_{agg}^i$ surrogates. Use Eq. (2) to compute candidate point fitness values.
- Repeat for T_{A2}.
- Go to *Step 8*.

Step 8: Contour Selection (Constraint-Penalty)

- Use the discriminative sampling procedure [31] and the constraint-penalizing merit function in Ref. [26] to select a design $\boldsymbol{x}_{A1i}$ from $\{z_{A1}\}$. Evaluate $\boldsymbol{x}_{A1i}$ with the black-box to obtain results $\boldsymbol{f}_{x_{A1i}}$ and $\boldsymbol{c}_{\boldsymbol{x}_{A1i}}$. Place $\boldsymbol{x}_{A1i}$ in $\{y_{A1}\}$. Repeat for T_{A2}.

Step 9a: Contour Selection (Exploitation)

- Use the discriminative sampling procedure to select designs $\boldsymbol{x}_{A1i}$ and $\boldsymbol{x}_{A2i}$ from $\{z_{A1}\}$ and $\{z_{A2}\}$, respectively. Use the ROD strategy with Eqs. (8) and (9) as the merit functions for T_{A1} and T_{A2}, respectively. Evaluate $\boldsymbol{x}_{A1i}$ with the black-box to obtain results $\boldsymbol{f}_{x_{A1i}}$ and $\boldsymbol{c}_{\boldsymbol{x}_{A1i}}$. Place $\boldsymbol{x}_{A1i}$ in $\{y_{A1}\}$. Do the same for $\boldsymbol{x}_{A2i}$ and $\{y_{A2}\}$.
- Go to *Step 10*.

Step 9b: Contour Selection (Exploration)

- Use the discriminative sampling procedure to select designs $\boldsymbol{x}_{B1i}$ and $\boldsymbol{x}_{B2i}$ from $\{z_{B1}\}$ and $\{z_{B2}\}$, respectively. Use the K-Opp strategy with Eq. (12) as the merit function. Evaluate $\boldsymbol{x}_{B1i}$ with the black-box to obtain results $\boldsymbol{f}_{x_{B1i}}$ and $\boldsymbol{c}_{\boldsymbol{x}_{B1i}}$. Place $\boldsymbol{x}_{B1i}$ in $\{y_{B1}\}$. Do the same for $\boldsymbol{x}_{B2i}$ and $\{y_{B2}\}$.

Step 10: Update SAKS Conservativeness

- Using $\boldsymbol{c}_{\boldsymbol{x}_{A1i}}$ and $\boldsymbol{c}_{\boldsymbol{x}_{A2i}}$, update ρ_A^{i+1} according to Eq. (7).

Step 11: B Regions

- Perform *Steps 2–10* for T_{B1} and T_{B2}, and going to *Steps 9b* instead of *9a*.

Step 12: Region Updates

- Update frontier F_i.
- For T_{A1}, if the frontier has been improved at this iteration with the induction of new nondominated points, $R_{A1} = R_{A1}/k_{reduction}$, where R_{A1} is the normalized region size and $k_{reduction}$ is a region shrinkage factor.
- Else if no improvement for stall iterations, $R_A = R_A \cdot k_{reduction}$.
- Repeat the above for T_{A2}, T_{B1}, and T_{B2}.
- $\{y_{A1}\}$, $\{y_{A2}\}$, $\{y_{B1}\}$, $\{y_{B2}\} = \varnothing$
- For each pt in $\{y\}$, place in every trust region that it fits in.
- $i = i + 1$

Step 13: Convergence Criteria

- If either the max *nfe* (number of function evaluations) or minimum *fval* (function value) criteria is met, then stop. Otherwise, go to *Step 2*.

Benchmark Process and Results

Frontier Evaluation Metrics. A variety of different metrics exist to evaluate the quality of Pareto frontiers [38]. The metrics used in this work are the hypervolume indicator (S-metric) [39] and a modified version of set coverage (C-metric) [10] called mean set coverage (CMean). These two metrics are chosen because they are complementary, and both do not require knowledge of the true Pareto frontier. As the true Pareto frontier is not known for many of the benchmark problems and all the simulation-based problems, the S and C metrics combined provide a good indication of algorithm performance that can be applied universally.

Hypervolume Indicator (S-metric). The S-metric measures the volume in the objective space that is encapsulated by the nondominated solutions and a reference point. The reference point should have objective values that are at least the maximum of all objective values in the set of nondominated solutions. When comparing multiple algorithms, the reference point for each problem should be defined as the maximum of all objective values in all sets of nondominated solutions to establish a consistent metric. The method generates a hypercube v_i for each nondominated point i in a set of nondominated points such that point i and the reference point are diagonal corners of the hypercube [40]. The method then calculates the union of all hypercubes for the set of nondominated points to determine the S-metric value:

$$S_{metric} = \text{volume}\left(\bigcup_{i=1}^{|F|} v_i\right) \tag{13}$$

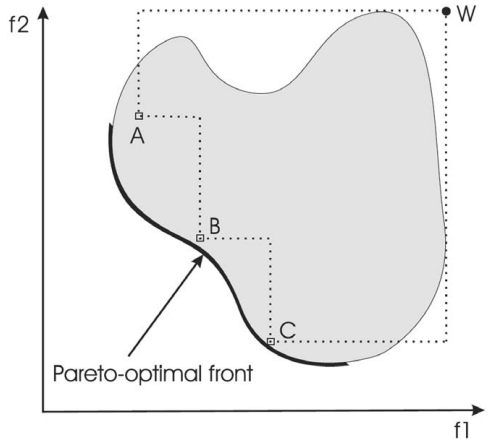

Fig. 4 Illustration of the hypervolume indicator [40] (Reprinted with permission from Elsevier © 2011)

where F is the set of nondominated solutions. Figure 4 is a two-dimensional (2D) illustration of the hypervolume indicator. The area within the dotted lines represents the S-metric volume that is calculated. Frontiers with larger S-metric values are closer to the Pareto frontier, and thus, larger S-metric values indicate better algorithm performance. The implementation of the S-metric used in this work is a hypervolume approximator by Bringmann and Friedrich [41] implemented in the software package PaGMO [42].

Mean Set Coverage (CMean). The C-metric compares two frontiers A and B by computing the percentage of designs in frontier B that are dominated by at least one design in frontier A [10]

$$C_{metric}(A, B) = \frac{|u \in B \exists v \in A: v \text{ dominates } u|}{|B|} \tag{14}$$

This metric complements the S-metric because a frontier that has a higher hypervolume value may have a large percentage of its designs being dominated by a different frontier with a lower hypervolume value. The hypervolume indicator favors convex frontiers [39], which C-metric can counterbalance.

One drawback of C-metric is that it can only compare two frontiers. When algorithms are run for multiple runs as part of a stochastic benchmarking process, it is desirable to compare two sets of frontiers. This work introduces the mean set coverage (CMean) metric, where C-metric is calculated for all combinations of frontiers between the two sets, and the results are averaged to obtain a true comparison between two algorithms. The CMean metric is computed as

$$CMean(\{A\}, \{B\}) = \frac{\sum_{i}^{h} \frac{\sum_{j}^{k} C_{metric}(A(i), B(j))}{k}}{h} \tag{15}$$

where $\{A\}$ and $\{B\}$ are two sets of frontiers generated by two different algorithms, and h and k are the numbers of frontiers in each set, respectively.

Test Methodology. SAKS-MTRO was compared to four different MOO algorithms in two different test suites. In the unconstrained test suite, SAKS-MTRO was compared to PSP [24], and the PaGMO [43,44] implementations of MOEA/D [10] and NSGA-II [8] on 11 unconstrained MOO benchmark problems. In the constrained test suite, SAKS-MTRO was compared to NSGA-II Program in Matlab(NPGM) [45], an implementation of Deb's NSGA-II that supports expensive constraints, and MOFEPSO [14] on seven constrained MOO benchmark problems. The unconstrained benchmark problems range from 10 to 30 variables and between two and five objectives. The constrained benchmark problems range from 8 to 30 variables, two to three objectives, and 1 to 21 constraints. Table 1 summarizes the properties of the test problems. Because NPGM and MOFEPSO struggled to find any feasible solutions for some constrained problems, four of the constrained problems were simplified by removing some constraints. These problems have the "mod" postfix. See the Appendix for details on each of the benchmark problems. Each of the algorithms was run with each of the benchmark problems 30 times to minimize the effect of random variation and starting point location on algorithm performance. To test the constraint handling efficiency of the algorithms, all constraints are considered to be computationally expensive, calculated along each evaluation of the objective function. In this work, NFE is the total number of function evaluations, with each function evaluation computing the results for all functions. All benchmark runs were limited to 500 NFEs because many industrial simulations are very computationally expensive, and thus, it is impractical to perform optimization with thousands of evaluations. Due to this limit, most benchmark results in this study do not resemble the true Pareto frontiers.

Pareto Set Pursuing. PSP [24] is a surrogate-based MOO that relies on a discriminative sampling of the surrogates to iteratively achieve frontier improvement. During each iteration, PSP constructs a surrogate model to generate many predictions at random designs. Discriminative sampling and the G function are used to select promising candidate designs to be evaluated using the black-box functions.

MOEA/D. MOEA/D is a decomposition-based evolutionary MOO algorithm [10]. It decomposes a multi-objective problem into a finite set of single-objective subproblems, with the number of subproblems corresponding to the population size. It iteratively evolves the population by using genetic operators and combines good solutions from neighboring problems to achieve frontier improvement. The PaGMO implementation of MOEA/D uses the rand/2/exp Differential Evolution operator and a polynomial mutation for population reproduction, and the Tchebycheff method for decomposition [11]. In the benchmarks, the population size and number of generations for MOEA/D vary according to Table 2. This results in at least 500 NFEs for each case as the PaGMO implementation does not count the initial population as a generation. Because of a limitation in the PaGMO implementation on population sizes, the population size needs to be selected from a discrete set of values that differ according to the number of objectives. The values in Table 2 were chosen because they generally produced better results than other randomly tried population sizes.

NSGA-II. NSGA-II is a popular classic evolutionary MOO that combines a fast nondominated sorting approach with a crowded-comparison operator to iteratively improve the frontier [8]. At each iteration, it uses crossover and mutation operators to generate offspring. The offspring are then selected using the crowded-comparison operator which combines the domination and crowding-distance criteria to choose the new population. In the benchmarks, the population size was set to 24 and the number of generations was set to 20 for NSGA-II, which results in 504 NFEs per run. As with MOEA/D, this implementation does not count the initial population as a generation. A population size of 24 was selected as it gave better results than other choices. The PaGMO implementation also limits the population size to a multiple of four.

NPGM. NPGM is a MATLAB implementation of NSGA-II that adds rudimentary expensive constraint handling [45]. The method handles constraints by computing the sum of violated constraints for each expensive candidate design and penalizing their domination level during the nondominated sorting process. This

significantly reduces the likelihood that a candidate will be selected to form the new population at each iteration. In the benchmarks, the population size was set to 20 and the number of generations was set to 25 for NPGM, which results in 500 NFEs per run; 20 was selected for population size as it gave better results than other randomly tried choices. NPGM also limits the population size to even numbers.-OFEPSO

MOFEPSO is a partical swarm optimization (PSO)-based approach that employs different strategies for feasible and infeasible particles to handle constrained MOO problems [14]. Infeasible particles aim to reach feasibility by prioritizing a single violated constraint. The movement of infeasible particles is restricted to the parameters for which the chosen constraint is sensitive. The implementation of MOFEPSO was obtained from MATLAB Central File Exchange [46]. In the benchmarks, the swarm size was set to 10 and the number of generations was limited to achieve a target average NFEs of 500 per run.

Benchmark Results

Unconstrained Benchmark Results

Tables 3 and 4 contain the S-metric and CMean results, respectively, for the unconstrained benchmark cases. The results show that SAKS-MTRO achieves the best S-metric results for six of the 11 cases and has the best CMean performance in seven of the 11 cases. While the S-metric performance is worse than the CMean performance, SAKS-MTRO trails the best results by relatively small margins in the cases where it does not have the best S-metric result. In the cases where SAKS-MTRO has the best S-metric result, it generally does so by very wide margins. SAKS-MTRO is followed by NSGA-II in terms of overall performance. NSGA-II has the best S-metric results for three cases and beats SAKS-MTRO in CMean performance in two cases. NSGA-II is followed by MOEA/D, with PSP having the worst overall performance. While PSP and MOEA/D both have the one case with best performing S-metric results, PSP performs worse than MOEA/D on most problems. In many of the problems the results for our method, NSGA-II, and PSP demonstrate crowding in knee regions of the frontier with only MOEA/D demonstrating relatively even distributions of points.

In the bi-objective Zitzler Deb Thiele (ZDT) test suite, SAKS-MTRO performs well in all cases except for the ZDT3 and ZDT4 problems where it trails NSGA-II in performance. Figures 5–7 are single-run frontier plots for the ZDT test suite that reflect the average performance of each method. Figure 5 shows that SAKS-MTRO performs well for the standard convex (ZDT1)- and non-convex (ZDT2)-type problems. Figure 6 shows that SAKS-MTRO is close to NSGA-II's performance for ZDT3 but lags substantially for ZDT4, which is a highly multimodal problem with 21^9 local Pareto-optimal fronts [47]. Figure 7 shows SAKS-MTRO performing substantially better than the other three algorithms on ZDT6, which is a problem that has a lower density of solutions near the Pareto-optimal front [47].

SAKS-MTRO performs similarly as other algorithms for the DTLZ1 test problems with three and five objectives. For DTLZ1-3 and DTLZ1-5, SAKS-MTRO has very close results in the S-metric compared to the other methods but comes third and second in the CMean results. Performance in the DTLZ2 problems is much better, with SAKS-MTRO beating the other methods in both S-metric and CMean for DTLZ2-3. Although it has slightly worse S-metric performance on DTLZ2-5 compared to PSP, it has much better CMean performance. For the DTLZ7 problems, SAKS-MTRO has a much better performance compared to the other methods across the board.

Constrained Benchmark Results

Table 5 contains all numerical results for the constrained benchmark cases. The Success Rate column indicates the percentage of runs that result in feasible solutions. The "*" symbol indicates that feasible solutions are found by random sampling during the initial sampling stage of the algorithm, while the "-" symbol indicates that no feasible solutions were found on any runs. The results show that SAKS-MTRO has the best results overall compared to NPGM and MOFEPSO, oftentimes with very large performance leads. NPGM performs well for loosely constrained problems but performs poorly for problems with more or tighter constraints. In some cases, it fails to find feasible solutions in the given computational budget. MOFEPSO generally traded blows with NPGM depending on the problem. In the following results, all figures are single-run results that reflect the average performance for each algorithm on the given problems.

Figure 8 shows one trial run of SAKS-MTRO, NPGM, and MOFEPSO on the bi-objective single-constraint CF6 and CF7 problems. While SAKS-MTRO has a solid performance lead, NPGM comes close in performance due to the presence of only a single constraint with MOFEPSO lagging. SAKS-MTRO and NPGM both have significant crowding around knee regions while MOFEPSO has better-distributed results.

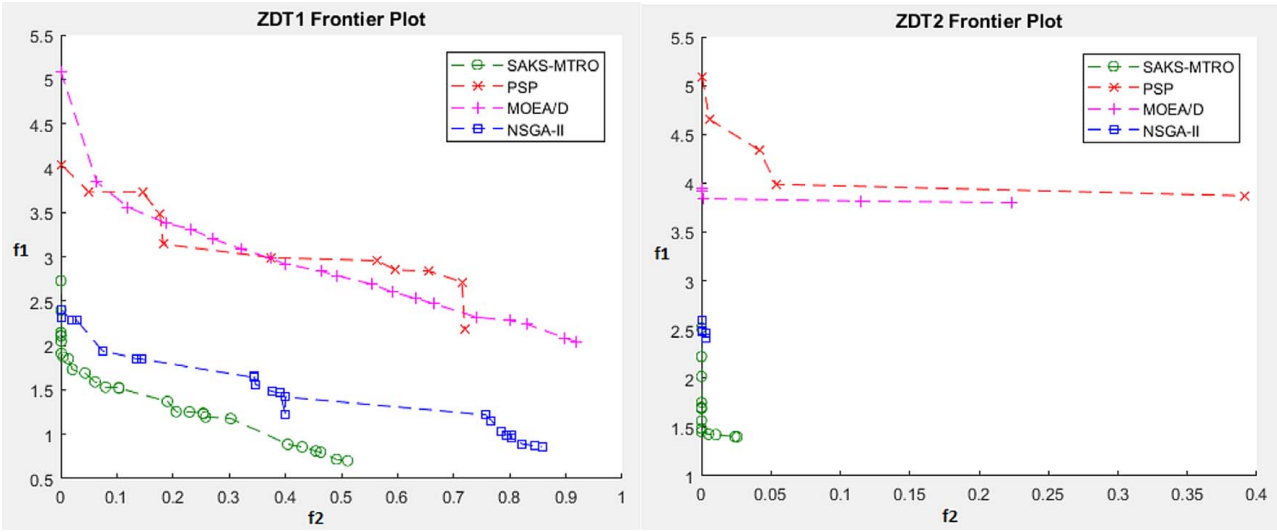

Fig. 5 Comparison of SAKS-MTRO on ZDT1 (left) and ZDT2 (right) frontier plots

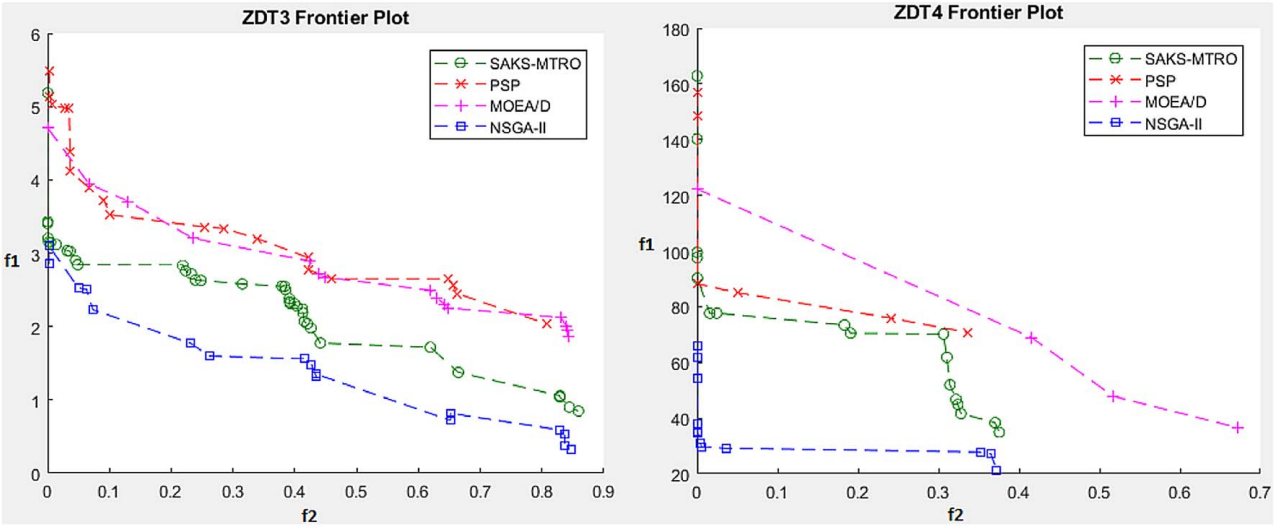

Fig. 6 Comparison of SAKS-MTRO on ZDT3 (left) and ZDT4 (right) frontier plots

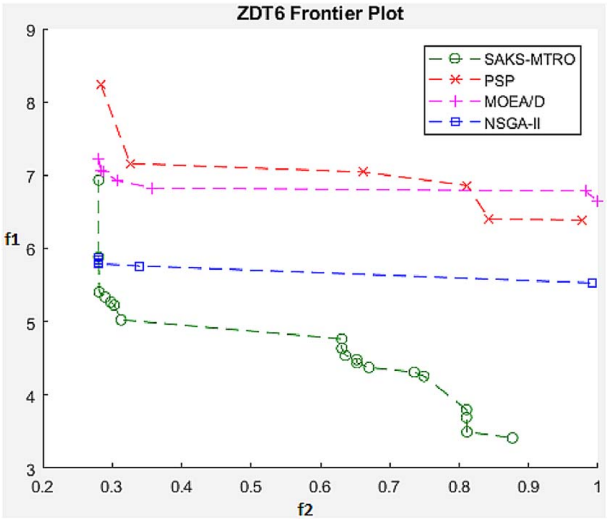

Fig. 7 Comparison of SAKS-MTRO on ZDT6 frontier plot

From Fig. 9, NPGM has difficulty exploring the search space with the presence of multiple expensive constraints. This has resulted in NPGM producing very short fronts. Table 5 also shows that 30% of NPGM runs for the P113mod problem were unable to produce feasible solutions and that it takes hundreds of NFEs to achieve feasibility for both problems during the runs that were successful. MOFEPSO performed much better on P113mod but failed to obtain any feasible designs in all 30 runs on the TP3mod problem despite the presence of only three constraints. This contrasts with SAKS-MTRO, which was able to generate broad and well-distributed frontiers under the same circumstances. Figure 10 shows NPGM struggling to achieve feasibility on the P106mod problem. NPGM only manages to achieve feasibility in 16.7% of runs. It does better on the Beam problem but lags SAKS-MTRO significantly. MOFEPSO does well on P106mod while performing similarly to NPGM for the Beam problem.

SAKS-MTRO demonstrates the best overall performance in unconstrained MOO problems while losing to NSGA-II and MOEA/D on a few problems (notably, ZDT3, ZDT4, and DTLZ1). Overall, NSGA-II also performed very well, often performing just behind or just ahead of SAKS-MTRO. However, when there are expensive constraints present, SAKS-MTRO

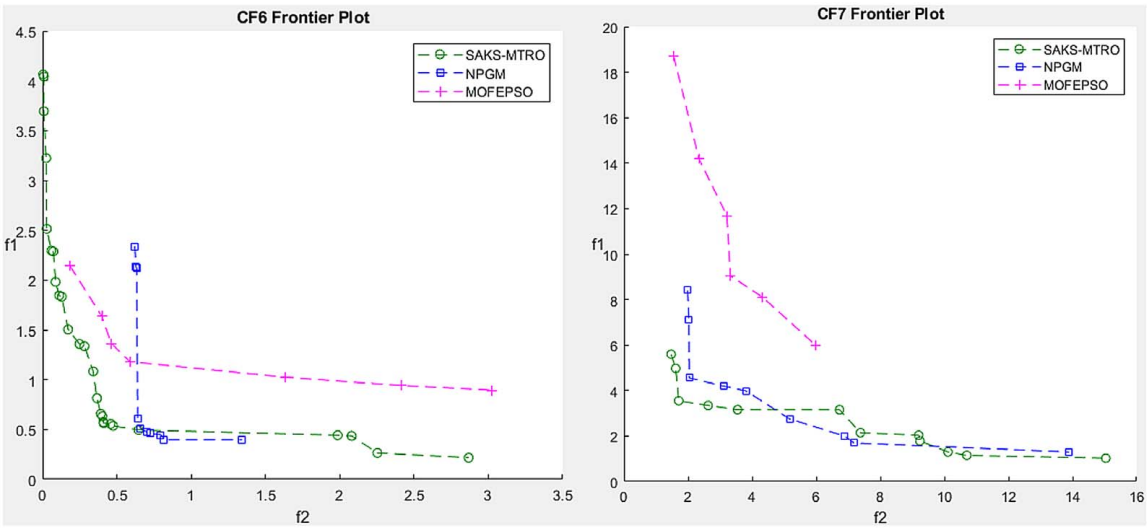

Fig. 8 Comparison of SAKS-MTRO on CF6 (left) and CF7 (right) frontier plots

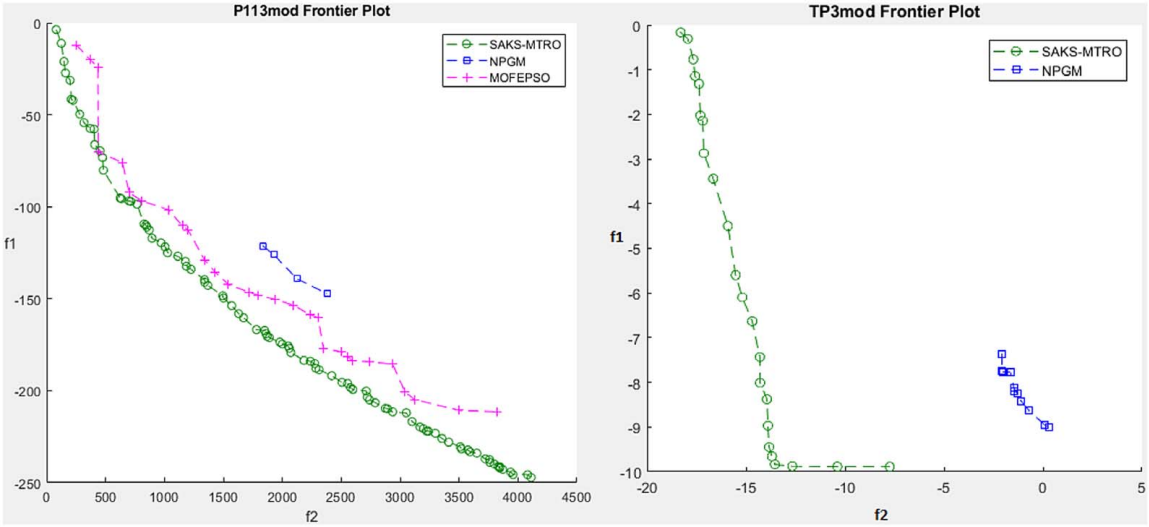

Fig. 9 Comparison of SAKS-MTRO on P113mod (left) and TP3mod (right) frontier plots

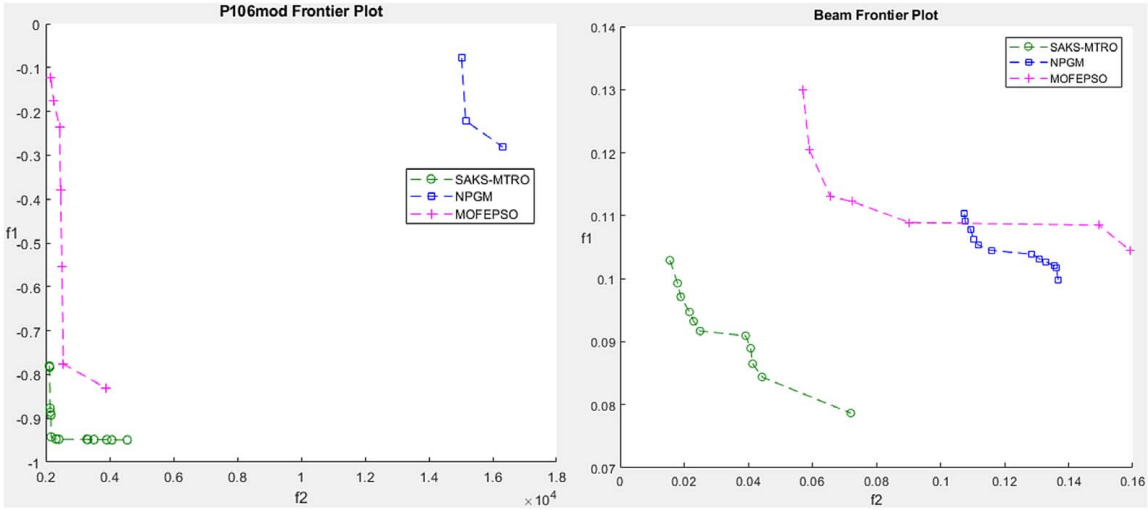

Fig. 10 Comparison of SAKS-MTRO on P106mod (left) and Beam (right) frontier plots

demonstrates a large and consistent performance lead compared to NPGM, the constrained version of NSGA-II, and MOFEPSO.

Industrial Applications of SAKS-MTRO

As part of this work, SAKS-MTRO is used to optimize the designs of an embedded copper trace substrate and an industrial recessed impeller for use in slurry pumping applications.

Embedded Copper Trace Substrate Optimization. Substrate packaging technology has been moving to coreless designs to reduce package size. Coreless technology relies on a temporary carrier platform as a base to build up material, which is removed at the end of the process. While coreless technology has advantages such as thinner packages, higher interconnect density, and better electrical performance [48], a disadvantage of removing the core is the increased warpage due to reduced structural rigidity [49] and the high temperatures of the layering process [48]. The warpage produces gaps between the substrate and silicon (see Fig. 11), which can lead to bonding failure because of the solder balls failing to connect the silicon to the copper traces. Even if the bonding was successful during the packaging process, the packages need to undergo rigorous stress and drop tests to ensure

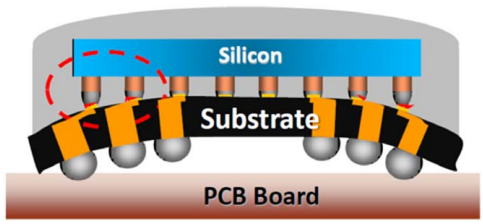

Fig. 11 Substrate warpage [50]

reliability and quality [51]. A warped substrate with weak bonds is more likely to fail during a drop test.

To address the warpage issue, SAKS-MTRO was used to optimize the substrate properties represented by an ANSYS Mechanical FEA model developed by Hwang et al. [50] that simulates the thermal deformation of a coreless embedded trace substrate during the packaging process.

Simulation Model. Because the structure of an embedded copper trace substrate is complicated [52], the substrate model geometry was simplified to reduce the computational cost of

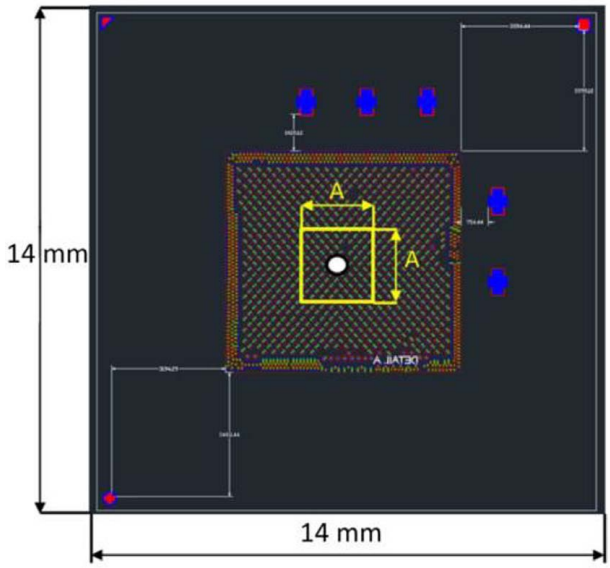

Fig. 12 Top view of the substrate [50]

Fig. 13 Substrate cross section [50]

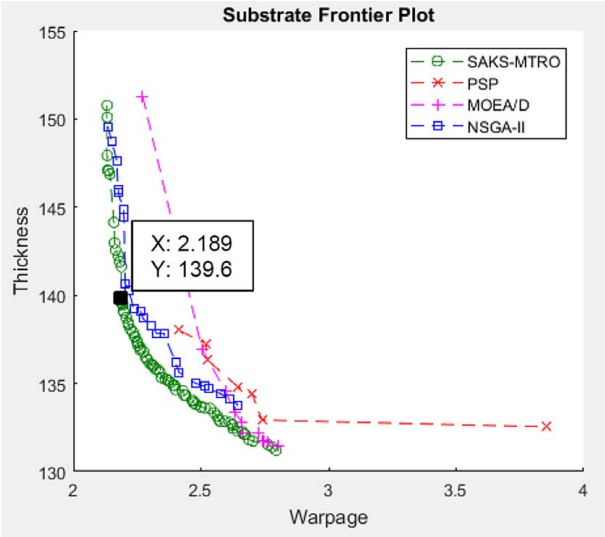

Fig. 14 Substrate optimization frontier plot

analysis [50]. Figure 12 shows the top view of the substrate, with the area indicated in the center with length $A = 2.8$ mm representing the actual geometry simulated using FEA.

The model is composed of three kinds of materials: copper (Cu), polypropylene (PP), and solder resist (SR). The substrate structure is composed primarily of five layers, as shown in Fig. 13: top solder resist (SR1), top copper trace (CT1), polypropylene (PP), bottom copper trace (CT2), and bottom solder resist (SR2).

The SR1, SR2, and PP layers depend on the geometry of CT1 and CT2 as the copper trace layers are partially embedded in the SR1, SR2, and PP layers. The layers are parameterized using the following nine unique variables:

SR1 (1 variable): thickness H_{SR1};
SR2 (2 variables): thickness H_{SR2}, circular opening diameter D_{SR2};
CT1 (3 variables): thickness H_{CT1}, the length l_{CT1} and width w_{CT1} of each rectangular trace;
CT2 (2 variables): thickness H_{CT2}, diameter D_{CT2} of each circular trace; and
PP (1 variable): thickness H_{pp}.

Table 6 shows the bounds for each variable. To shorten the computational cost of the simulation, the symmetrical properties of the model were leveraged to reduce the simulation geometry to one-eighth of the original size. To simulate the packaging process, the temperature was set to 25 °C initially and then increased to 183 °C during the heating process. The goal of the optimization is to minimize substrate warping as well as substrate thickness. Substrate warping data can be obtained from the simulation by extracting the maximum Z-axis displacement. Substrate thickness can be trivially calculated by summing the thickness variables. The full optimization problem formulation is as follows:

$$\min_{\substack{wrt\ H_{CT1},H_{CT2},\\ H_{SR1},H_{pp},H_{SR2},\\ D_{CT2},D_{SR2},w_{CT1},l_{CT1}}} \begin{aligned} F_1 &= |\max(Z - \text{displacement})| \\ F_2 &= H_{CT1} + H_{CT2} + H_{SR1} + H_{pp} + H_{SR2} \end{aligned}$$

$$\begin{aligned} s.t.\ 13 &\le H_{CT1} \le 23 \\ 15 &\le H_{CT2} \le 20 \\ 13 &\le H_{SR1} \le 23 \\ 75 &\le H_{PP} \le 85 \\ 15 &\le H_{SR2} \le 20 \\ 135 &\le D_{CT2} \le 145 \\ 110 &\le D_{SR2} \le 120 \\ 40 &\le w_{CT1} \le 50 \\ 70 &\le l_{CT1} \le 90 \end{aligned} \tag{16}$$

Unconstrained Multi-Objective Optimization. The substrate optimization problem was solved using SAKS-MTRO and the unconstrained optimizers PSP, MOEA/D, and NSGA-II. Because the simulation model ran quickly, 10 optimization runs were performed for each algorithm with a target NFE of 500. Table 7 shows the results for both S-metric and CMean. The results show that SAKS-MTRO obtained the best results for both the S-metric and CMean results. Figure 14 is a single-run frontier plot where, for each algorithm, the run that has the closest S-metric value to the S-metric average for that algorithm was selected for the plot. The figure clearly shows the high quality of the SAKS-MTRO frontier, with designs that are well distributed along the front. The plot shows a clear trade-off behavior between substrate thickness and warpage, with warpage accelerating as the substrate becomes thinner. From the SAKS-MTRO frontier, a solution with a good compromise between thickness and warpage has 139.6 μm thickness and 2.189 μm warpage.

Constrained Optimization of an Industrial Recessed Impeller. Slurry pumps are a type of centrifugal pump that are commonly used in the oil and gas, mining, and power generation industries. The slurry is a semiliquid mixture where solid particles are suspended in a liquid. Until recently, the design of slurry pumps has relied primarily on engineers' knowledge and experience, and physical prototyping and testing [53]. In this work, we used the same CFD model as in Ref. [26], which is a full model of the impeller and volute [54]. ANSYS BladeModeler was used to build the impeller and volute geometry. ANSYS Meshing was

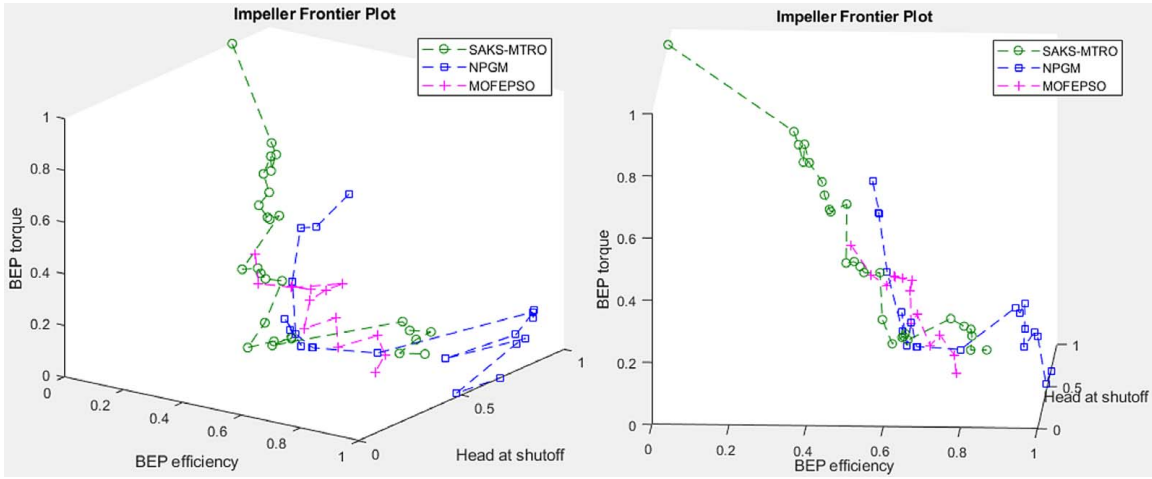

Fig. 15 Normalized frontier plots for impeller optimization

used to generate a mesh of the fluid domain of the impeller and volute. ANSYS CFX was used to setup and run the simulation. CFX simulations were run at two different volumetric flowrates, with the higher flowrate at approximately the best efficiency point (BEP).

SAKS-MTRO was applied to explore the trade-off behavior between the high flowrate efficiency, head at lowest flowrate (head at shutoff), and torque at high flowrate. Head is the ability of the pump to raise water to a certain height while shutoff is the lowest pump flowrate. The high flowrate efficiency and torque are referred to as BEP efficiency and torque. Inequality constraints are placed on the output torques to ensure they are positive, as the CFD model sometimes output erroneous results when the torque values are at 0 or below. Also, the head at all flowrates are constrained to be within reasonable bounds as there were cases of head values being extremely high, which were anomalies of the low-fidelity simulation model as the head values obtained on the same designs using the high-fidelity model were quite low. Since head, efficiency, and torque values are all output from the CFD simulation, these constraints are all computationally expensive. In addition, nine math constraints were added to constrain the input parameters to reduce the incidence of meshing and solver failures due to bad geometry. We defined 12 input parameters representing different geometric features of the impeller blades as specified in Ref. [26]. Due to confidentiality, the input parameters and operating velocities are generalized and values are expressed relative to nominal. The multi-objective problem formulation is as follows:

$$
\begin{aligned}
&f_1(\boldsymbol{x}) = -\text{efficiency at mid flow rate}\\
&f_2(\boldsymbol{x}) = -\text{head at low flow rate}\\
&f_3(\boldsymbol{x}) = -\text{torque at mid flow rate}\\
&g_{1,2}(\boldsymbol{x}) \rightarrow \text{head at low and mid flow rates} \leq \text{maximum head limit}\\
&g_{3,4}(\boldsymbol{x}) = \text{head at low and mid flow rates} \geq 0\\
&g_{5,6}(\boldsymbol{x}) = \text{torque at low and mid flow rates} \geq 0\\
&g_{7\rightarrow 15}(\boldsymbol{x}) = \text{various mathematical constraints on input parameters}
\end{aligned}
\tag{17}
$$

The multi-objective impeller optimization problem was solved using the constrained optimizers SAKS-MTRO, NPGM, and MOFEPSO. Because of the lengthy simulation runs, a single optimization run was performed for each algorithm with a target NFE of 200.

Table 8 shows the results for both S-metric and C-metric. For confidentiality reasons, the S-metric result for SAKS-MTRO is shown as a percentage of the NPGM and MOFEPSO S-metric values. The results show that SAKS-MTRO obtained the best results for both the S-metric and C-metric results. Figure 15 shows the normalized frontier plot for the impeller optimization for SAKS-MTRO, NPGM, and MOFEPSO. There is a clear relationship between BEP torque and efficiency, with torque increasing significantly only when efficiency has dropped past the 0.6 normalized mark. On the other hand, there is not a clear trade-off between head at shutoff and BEP efficiency, which means there is the possibility to keep both high without sacrificing one or the other.

Final Remarks

This work proposed the SAKS-Multi-objective Trust Region Optimizer (SAKS-MTRO) for expensively constrained and high-dimensional black-box multi-objective optimization. The Random Objective Decomposition (ROD) and *K*-Means Opposition Search (*K*-Opp) strategies were introduced, which balances exploitation and exploration in the multi-objective space. The SAKS-TRO method was also modified to accommodate four semi-independent trust regions that use ROD and K-Opp to perform constrained multi-objective optimization. The method was benchmarked against three other unconstrained multi-objective optimizers and two constrained multi-objective optimizers on a suite of unconstrained and constrained benchmark problems. The method was also benchmarked on two industrial optimization applications.

The benchmarks show that SAKS-MTRO performs well compared to PSP, MOEA/D, and NSGA-II on the unconstrained problems, and has a significant performance lead compared to NPGM and MOFEPSO on the constrained problems. The results show that our method can perform well for both unconstrained and constrained MOO problems without tuning the hyperparameters. However, some limitations of our method are issues with concentration of frontier points around knee regions and a lack of testing around many-objective problems, which will be the focus of future work.

Acknowledgment

The authors are grateful for the financial support of the National Sciences and Engineering Research Council (NSERC), without which this work would not have been possible. The authors also are grateful for access to ANSYS academic licenses from CMC Microsystems.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data sets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request. The authors attest that all data for this study are included in the paper. Data provided by a third party are listed in Acknowledgment.

Appendix

Unconstrained Benchmark Problems

ZDT Test Suite [47]

ZDT1

$$\begin{aligned} F_1(\boldsymbol{x}) &= x_1 \\ F_2(\boldsymbol{x}) &= g(\boldsymbol{x})\left(1-\sqrt{\frac{F_1(\boldsymbol{x})}{g(\boldsymbol{x})}}\right) \\ g(\boldsymbol{x}) &= 1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \\ \boldsymbol{x} &\in [0,\ 1] \\ n &= 30 \end{aligned} \tag{A1}$$

ZDT2

$$\begin{aligned} F_1(\boldsymbol{x}) &= x_1 \\ F_2(\boldsymbol{x}) &= g(\boldsymbol{x})\left(1-\left(\frac{F_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2\right) \\ g(\boldsymbol{x}) &= 1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \\ \boldsymbol{x} &\in [0,\ 1] \\ n &= 30 \end{aligned} \tag{A2}$$

ZDT3

$$\begin{aligned} F_1(\boldsymbol{x}) &= x_1 \\ F_2(\boldsymbol{x}) &= g(\boldsymbol{x})\left(1-\sqrt{\frac{F_1(\boldsymbol{x})}{g(\boldsymbol{x})}}-\left(\frac{F_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)\sin(10\pi F_1(\boldsymbol{x}))\right) \\ g(\boldsymbol{x}) &= 1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \\ \boldsymbol{x} &\in [0,\ 1] \\ n &= 30 \end{aligned} \tag{A3}$$

ZDT4

$$\begin{aligned} F_1(\boldsymbol{x}) &= x_1 \\ F_2(\boldsymbol{x}) &= g(\boldsymbol{x})\left(1-\sqrt{\frac{F_1(\boldsymbol{x})}{g(\boldsymbol{x})}}\right) \\ g(\boldsymbol{x}) &= 1+10(n-1)+\left(\sum_{i=2}^{n} x_i^2-10\cos(4\pi x_i)\right) \\ x_1 &\in [0,\ 1];\ x_2,\ldots,x_{\boldsymbol{n}} \in [-5,\ 5] \\ n &= 10 \end{aligned} \tag{A4}$$

ZDT6

$$\begin{aligned} F_1(\boldsymbol{x}) &= 1-\exp(-4x_1)\sin^6(6\pi x_1) \\ F_2(\boldsymbol{x}) &= g(\boldsymbol{x})\left(1-\left(\frac{F_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2\right) \\ g(\boldsymbol{x}) &= 1+9\left(\frac{\left(\sum_{i=2}^{n} x_i\right)}{(n-1)}\right)^{0.25} \\ \boldsymbol{x} &\in [0,\ 1] \\ n &= 10 \end{aligned} \tag{A5}$$

DTLZ Test Suite [55]

DTLZ1

$$\begin{aligned} F_1(\boldsymbol{x}) &= 0.5x_1x_2\ldots x_{M-1}(1+g(X_M)) \\ F_2(\boldsymbol{x}) &= 0.5x_1x_2\ldots(1-x_{M-1})(1+g(X_M)) \\ &\vdots \\ F_{M-1}(\boldsymbol{x}) &= 0.5x_1(1-x_2)(1+g(X_M)) \\ F_M(\boldsymbol{x}) &= 0.5(1-x_2)(1+g(X_M)) \\ g(X_M) &= 100\left[|X_M|+\sum_{x_i\in X_M}(x_i-0.5)^2-\cos(20\pi(x_i-0.5))\right] \\ \boldsymbol{x} &\in [0,\ 1];\ |X_M|=5 \\ n &= M+4 \end{aligned} \tag{A6}$$

Table 1 Properties of multi-objective test problems

Problem	# Variables	# Objectives	# Constraints
ZDT1	30	2	0
ZDT2	30	2	0
ZDT3	30	2	0
ZDT4	10	2	0
ZDT6	10	2	0
DTLZ1-3	7	3	0
DTLZ1-5	9	5	0
DTLZ2-3	12	3	0
DTLZ2-5	14	5	0
DTLZ7-3	22	3	0
DTLZ7-5	24	5	0
CF6	10	2	2
CF7	10	2	2
P113mod	10	2	4
TP3mod	13	2	3
P106mod	8	2	3
P116mod	13	3	8
Beam	30	2	21

Table 2 MOEA/D parameters

# Objectives	Population size	# Generations
2	25	19
3	28	17
5	35	15

Table 3 SAKS-MTRO unconstrained benchmark results (S-metric)

	SAKS-MTRO		PSP		MOEA/D		NSGA-II	
Problem	Mean	STD	Mean	STD	Mean	STD	Mean	STD
ZDT1	**5.372**	0.093	3.486	0.0628	3.807	0.302	4.913	0.184
ZDT2	**6.012**	0.0808	3.414	0.151	3.539	0.280	4.685	0.290
ZDT3	4.893	0.155	3.911	0.0937	4.100	0.253	**5.360**	0.162
ZDT4	180.0	8.15	152.7	7.54	147.7	28.3	**201.5**	7.83
ZDT6	**3.647**	0.671	1.847	0.266	1.979	0.695	2.735	0.457
DTLZ1-3	9.937×10^{7}	1.40×10^{5}	9.927×10^{7}	1.14×10^{5}	9.940×10^{7}	1.00×10^{5}	$\mathbf{9.951\times10^{7}}$	6.61×10^{4}
DTLZ1-5	2.075×10^{13}	6.50×10^{10}	2.075×10^{13}	4.40×10^{10}	$\mathbf{2.079\times10^{13}}$	3.86×10^{10}	2.070×10^{13}	8.71×10^{10}
DTLZ2-3	**33.39**	0.0481	33.13	0.0252	30.63	0.490	32.96	0.328
DTLZ2-5	393.2	7.59	**398.0**	0.912	348.0	14.9	376.1	9.94
DTLZ7-3	**17.89**	0.517	15.37	0.189	8.508	2.17	16.99	0.685
DTLZ7-5	**27.67**	1.19	21.73	0.434	13.03	3.54	19.94	3.94

Note: Bold text indicates the algorithm that performed best for that metric.

Table 4 SAKS-MTRO unconstrained benchmark results (CMean, %)

	SAKS-MTRO			PSP	MOEA/D	NSGA-II
Problem	CMean versus PSP	CMean versus MOEA/D	CMean versus NSGA-II	CMean versus SAKS-MTRO	CMean versus SAKS-MTRO	CMean versus SAKS-MTRO
ZDT1	**99.92**	**99.93**	**91.04**	0.000	0.000	3.54
ZDT2	**100**	**99.72**	**98.59**	0.000	0.01	0.05
ZDT3	99.68	93.76	6.61	0.17	1.86	**73.34**
ZDT4	94.67	57.47	1.90	1.70	6.67	**50.79**
ZDT6	**94.71**	**89.56**	**51.98**	1.73	5.07	14.57
DTLZ1-3	69.63	8.38	7.20	7.53	69.44	**57.64**
DTLZ1-5	47.68	0.37	15.38	11.81	**66.46**	15.22
DTLZ2-3	**72.41**	**30.94**	**31.57**	4.39	1.54	12.53
DTLZ2-5	**48.62**	**9.61**	**37.29**	2.56	1.47	0.60
DTLZ7-3	**97.56**	**83.15**	**49.8**	0.23	0.47	11.69
DTLZ7-5	**90.49**	**23.25**	**50.78**	0.53	1.72	3.18

Note: Bold text indicates the algorithm that performed best for that metric.

Table 5 SAKS-MTRO constrained benchmark results

	SAKS-MTRO					
Problem	S-Metric mean	S-Metric STD	CMean versus NPGM	CMean versus MOFEPSO	Success rate	Mean NFE to feasible
CF6	**84.64**	1.77	**59.90**	**0.8687**	100	*
CF7	**485.0**	21.8	**50.47**	**0.9169**	100	*
P113mod	$\mathbf{7.312\times10^{5}}$	3.55×10^{3}	**98.77**	**0.9964**	100	*
TP3mod	**181.4**	7.53	**96.28**	**1.000**	100	28.00
P106mod	$\mathbf{1.867\times10^{4}}$	16.7	**100**	0.04976	100	*
P116mod	**75.43**	0.843	**90.29**	**0.9791**	100	18.667
Beam	**0.02495**	0.00134	**99.42**	**0.9992**	100	11.333

	NPGM					MOFEPSO				
Problem	S-metric mean	S-metric STD	CMean versus SAKS-MTRO	Success rate	Mean NFE to Feasible	S-metric mean	S-metric STD	CMean versus SAKS-MTRO	Success rate	Mean NFE to Feasible
CF6	81.33	2.43	7.23	100	*	73.95	5.82	0.0202	100	152.9
CF7	475.0	24.5	28.13	100	*	400.2	31.4	0.0271	100	154.8
P113mod	3.611×10^{5}	9.85×10^{4}	0.10	70.0	257.0	6.000×10^{5}	3.14×10^{4}	3.440e−4	100	171.0
TP3mod	39.04	10.7	0.00	100	355.6	–	–	–	0	–
P106mod	613.9	722	0.00	16.7	368.2	1.311×10^{4}	5.82×10^{3}	**0.5681**	100	128.9
P116mod	41.36	16.7	0.11	100	42.07	18.35	11.6	9.486e−5	100	200.6
Beam	0.01243	0.00337	0.000	100	106.7	0.01310	0.00199	0.000	100	458.4

Note: Bold text indicates the algorithm that performed best for that metric.

Table 6 Substrate model variable bounds

Parameter name	Lower bound (μm)	Upper bound (μm)
H_{CT1}	13	23
H_{CT2}	15	20
H_{SR1}	13	23
H_{pp}	75	85
H_{SR2}	15	20
D_{CT2}	135	145
D_{SR2}	110	120
w_{CT1}	40	50
l_{CT1}	70	90

Table 7 Substrate optimization results for S-metric (top) and CMean (bottom, %)

S-metric							
SAKS-MTRO		PSP		MOEA/D		NSGA-II	
Mean	STD	Mean	STD	Mean	STD	Mean	STD
60.42	0.454	49.68	5.30	52.39	3.39	55.01	2.81

SAKS-MTRO			PSP	MOEA/D	NSGA-II
CMean versus PSP	CMean versus MOEA/D	CMean versus NSGA-II	CMean versus SAKS-MTRO	CMean versus SAKS-MTRO	CMean versus SAKS-MTRO
98.51	81.66	89.46	0.37	3.57	4.65

Table 8 Impeller optimization results for S-metric and C-metric

SAKS-MTRO				
S-metric versus NPGM	S-metric versus MOFEPSO	C-metric versus NPGM	C-metric versus MOFEPSO	NFE to Feasible
138.3%	125.4%	59.90%	57.14%	<20

NPGM		MOFEPSO	
C-metric versus SAKS-MTRO	NFE to Feasible	C-metric versus SAKS-MTRO	NFE to Feasible
7.23%	23	21.43%	52

DTLZ2

$$
\begin{aligned}
F_1(\boldsymbol{x}) &= (1+g(X_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\\
&\quad \cdots \cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2)\\
F_2(\boldsymbol{x}) &= (1+g(X_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\\
&\quad \cdots \cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2)\\
F_3(\boldsymbol{x}) &= (1+g(X_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\sin(x_{M-2}\pi/2)\\
&\vdots\\
F_{M-1}(\boldsymbol{x}) &= (1+g(X_M))\cos(x_1\pi/2)\sin(x_2\pi/2)\\
F_M(\boldsymbol{x}) &= (1+g(X_M))\sin(x_1\pi/2)\\
g(X_M) &= \sum_{x_i\in X_M}(x_i-0.5)^2\\
&\boldsymbol{x}\in[0,\ 1];\ |X_M|=10\\
&n=M+9
\end{aligned}
\tag{A7}
$$

DTLZ7

$$
\begin{aligned}
F_1(\boldsymbol{x}) &= x_1\\
F_2(\boldsymbol{x}) &= x_2\\
&\vdots\\
F_{M-1}(\boldsymbol{x}) &= x_{M-1}\\
F_M(\boldsymbol{x}) &= (1+g(X_M))h(F_1,F_2,\ldots,F_{M-1},g)\\
g(X_M) &= 1+\frac{9}{|X_M|}\sum_{x_i\in X_M}x_i\\
h(F_1,F_2,\ldots,F_{M-1},g) &= M-\sum_{i=1}^{M-1}\left[\frac{f_i}{1+g}(1+\sin(3\pi f_i))\right]\\
&\boldsymbol{x}\in[0,\ 1];\ |X_M|=20\\
&n=M+19
\end{aligned}
\tag{A8}
$$

KS-MTRO Constrained Benchmark Problems

CF6

$$F_1(\boldsymbol{x}) = x_1 + \sum_{j\in J_1} y_j^2$$

$$F_2(\boldsymbol{x}) = (1-x_1)^2 + \sum_{j\in J_2} y_j^2$$

$$g_1(\boldsymbol{x}) = x_2 - 0.8x_1 \sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - sign(0.5(1-x_1) - (1-x_1)^2)\sqrt{|0.5(1-x_1) - (1-x_1)^2|} \geq 0$$

$$g_2(\boldsymbol{x}) = x_4 - 0.8x_1 \sin\left(6\pi x_1 + \frac{4\pi}{n}\right) - sign(0.25(1-x_1) - 0.5(1-x_1))\sqrt{|0.25(1-x_1) - 0.5(1-x_1)|} \geq 0$$

$J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}$

$J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\}$

$$y_j = \begin{cases} x_j - 0.8x_1 \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) \text{ if } j \in J_1 \\ x_j - 0.8x_1 \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) \text{ if } j \in J_2 \end{cases}$$

$x_1 \in [0,\ 1]; x_2, \ldots, x_n \in [-2,\ 2]$

$n = 10$ (A9)

CF7

$$F_1(\boldsymbol{x}) = x_1 + \sum_{j\in J_1} h_j(y_j)$$

$$F_2(\boldsymbol{x}) = (1-x_1)^2 + \sum_{j\in J_2} h_j(y_j)$$

$$g_1(\boldsymbol{x}) = x_2 - \sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - sign(0.5(1-x_1) - (1-x_1)^2)\sqrt{|0.5(1-x_1) - (1-x_1)^2|} \geq 0$$

$$g_2(\boldsymbol{x}) = x_4 - \sin\left(6\pi x_1 + \frac{4\pi}{n}\right) - sign(0.25(1-x_1) - 0.5(1-x_1))\sqrt{|0.25(1-x_1) - 0.5(1-x_1)|} \geq 0$$

$J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}$

$J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\}$

$$y_j = \begin{cases} x_j - \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) \text{ if } j \in J_1 \\ x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) \text{ if } j \in J_2 \end{cases}$$

$h_2(t) = h_4(t) = t^2$

$h_j(t) = 2t^2 - \cos(4\pi t) + 1; \text{ for } j = 3, 5, 6, \ldots, n$

$x_1 \in [0,\ 1]; x_2, \ldots, x_n \in [-2,\ 2]$

$n = 10$ (A10)

P113mod. The P113 is a 10-variable test problem with 8 constraints from Hock and Schittkowski [54]. We modified it to take the following form:

$$f_1(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2 + (x_5-3)^2 + 2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45$$

$$f_2(\boldsymbol{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8$$

$$g_1(\boldsymbol{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

$$g_2(\boldsymbol{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

$$g_3(\boldsymbol{x}) = 3(x_1-2)^2 + 4(x_2-3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$x \in [-10,\ 10]$ (A11)

TP3 [56] and TP3mod. The following is the TP3 problem modified to have two objectives:

$$F_1(\boldsymbol{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

$$F_2(\boldsymbol{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10$$

$$g_1(\boldsymbol{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(\boldsymbol{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(\boldsymbol{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(\boldsymbol{x}) = -8x_1 + x_{10} \leq 0$$

$$g_5(\boldsymbol{x}) = -8x_2 + x_{11} \leq 0$$

$$g_6(\boldsymbol{x}) = -8x_3 + x_{12} \leq 0$$

$$g_7(\boldsymbol{x}) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(\boldsymbol{x}) = -2x_6 - 2x_7 + x_{11} \leq 0$$

$$g_9(\boldsymbol{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

$x_1, \ldots, x_9 \in [0,\ 1]; x_{10}, \ldots, x_{12} \in [0,\ 100];\ x_{13} \in [0,\ 1]$ (A12)

The TP3mod problem is the same as the above with the last 6 constraints disabled.

P106mod. The P106mod is based on the P106 problem modified to have two objectives [54]

$$f_1(\boldsymbol{x}) = x_1 + x_2 + x_3$$

$$f_2(\boldsymbol{x}) = 0.0025(x_4 + x_6) - 1$$

$$g_1(\boldsymbol{x}) = 0.0025(x_5 + x_7 - x_4) - 1 \leq 0$$

$$g_2(\boldsymbol{x}) = 0.01(x_8 - x_5) - 1 \leq 0$$

$x_1 \in [1e2,\ 1e4], \{x_2, x_3\} \in [1e3,\ 1e4],$

$\{x_4, x_5, x_6, x_7, x_8\} \in [10,\ 1e3]$ (A13)

P116mod. The P116mod is based on the P116 problem in Ref. [54] and modified to have three objectives

$$f_1(\boldsymbol{x}) = x_{11} + x_{12} + x_{13}$$

$$f_2(\boldsymbol{x}) = x_2 - x_3$$

$$f_3(\boldsymbol{x}) = x_1 - x_2$$

$$g_1(\boldsymbol{x}) = 0.002x_7 - 0.002x_8 - 1 \leq 0$$

$$g_2(\boldsymbol{x}) = 50 - x_{11} - x_{12} - x_{13} \leq 0$$

$$g_3(\boldsymbol{x}) = x_{11} + x_{12} + x_{13} - 250 \leq 0$$

$$g_4(\boldsymbol{x}) = 1.262626x_{10} - 1.231059x_3x_{10} - x_{13} \leq 0$$

$$g_5(\boldsymbol{x}) = 0.03475x_2 + 0.975x_2x_5 - 0.00975x_2^2 - x_5 \leq 0$$

$$g_6(\boldsymbol{x}) = 0.03475x_3 + 0.975x_3x_6 - 0.00975x_3^2 - x_6 \leq 0$$

$\{x_1, x_2, x_3\} \in [0.1,\ 1], x_4 \in [1e-4,\ 0.1],$

$\{x_5, x_6\} \in [0.1,\ 0.9], \{x_7, x_8\} \in [0.1,\ 1e3],$

$x_9 \in [500,\ 1000],\ x_{10} \in [0.1,\ 500],$

$x_{11} \in [1,\ 150], \{x_{12}, x_{13}\} \in [0.0001,\ 150]$ (A14)

Beam. Same as the Beam problem in Ref. [26] with the following added objective

$$\sum_{i=1}^{d} b_i h_i l_i \tag{A15}$$

where d is the number of sections. This objective represents the total volume of the structure.

References

[1] Pérez, J., Orosa, J., and Grueiro, T., 2016, "A Three-Dimensional CFD Simulation Study to Reduce Heat Stress in Ships," Appl. Therm. Eng., **94**, pp. 413–420.
[2] Díaz-Ovalle, C., Martínez-Zamora, R., González-Alatorre, G., Rosales-Marines, L., and Lesso-Arroyo, R., 2017, "An Approach to Reduce the Pre-heating Time in a Convection Oven via CFD Simulation," Food Bioprod. Process., **102**, pp. 98–106.
[3] Rahnamayan, S., and Wang, G. G., 2009, "Toward Effective Initialization for Large-Scale Search Spaces," WSEAS Trans. Syst., **8**(3), pp. 355–367.
[4] Koch, P. N., Simpson, T. W., Allen, J. K., and Mistree, F., 1999, "Statistical Approximations for Multidisciplinary Design Optimization: The Problem of Size," J. Aircr., **36**(1), pp. 275–286.
[5] Shan, S., and Wang, G. G., 2010, "Survey of Modeling and Optimization Strategies to Solve High-Dimensional Design Problems with Computationally-Expensive Black-Box Functions," Struct. Multidiscipl. Optim., **41**(2), pp. 219–241.
[6] Srinivas, N., and Deb, K., 1995, "Multiobjective Function Optimization Using Nondominated Sorting Genetic Algorithms," Evol. Comput., **2**(3), pp. 221–248.
[7] Horn, J., Nafploitis, N., and Goldberg, D. E., 1994, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," Proceedings of the First IEEE Conference on Evolutionary Computation, Piscataway, NJ, June 27–29, pp. 82–87.
[8] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., 2002, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Trans. Evol. Comput., **6**(2), pp. 182–197.
[9] Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q., 2011, "Multiobjective Evolutionary Algorithms: A Survey of the State of the Art," Swarm Evol. Comput., **1**(1), pp. 32–49.
[10] Zhang, Q., and Li, H., 2007, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," IEEE Trans. Evol. Comput., **11**(6), pp. 712–731.
[11] Li, H., and Zhang, Q., 2009, "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II," IEEE Trans. Evol. Comput., **13**(2), pp. 284–302.
[12] Zitzler, E., and Künzli, S., 2004, "Indicator-Based Selection in Multiobjective Search," Parallel Problem Solving From Nature—PPSN VIII, Berlin, Sept. 18–22.
[13] Brockhoff, D., and Zitzler, E., 2007, "Improving Hypervolume-Based Multiobjective Evolutionary Algorithms by Using Objective Reduction Methods," IEEE Congress on Evolutionary Computation, Singapore, Sept. 25–28, pp. 2086–2093.
[14] Hasanoglu, M. S., and Dolen, M., 2018, "Multi-objective Feasibility Enhanced Particle Swarm Optimization," Eng. Optim., **50**(12), pp. 2013–2037.
[15] Chugh, T., Sindhya, K., Hakanen, J., and Miettinen, K., 2019, "A Survey on Handling Computationally Expensive Multiobjective Optimization Problems With Evolutionary Algorithms," Soft Comput., **23**(9), pp. 3137–3166.
[16] Coello, C., and Martinez, S., 2013, "MOEA/D Assisted by RBF Networks for Expensive Multi-objective Optimization Problems," Genetic and Evolutionary Computation Conference, New York.
[17] Zhu, J., Wang, Y.-J., and Collette, M., 2013, "A Multi-objective Variable-Fidelity Optimization Method for Genetic Algorithms," Engi. Optim., **46**(4), pp. 521–542.
[18] Regis, R., 2016, "Multi-objective Constrained Black-Box Optimization Using Radial Basis Function Surrogates," Comput. Sci., **16**, pp. 140–155.
[19] Singh, P., Couckuyt, I., Ferranti, F., and Dhaene, T., 2014, "A Constrained Multi-objective Surrogate-Based Optimization Algorithm," IEEE Trans. Evol. Comput., pp. 3080–3087.
[20] Tabatabaei, M., Hakanen, J., Hartikainen, M., Miettinen, K., and Sindhya, K., 2015, "A Survey on Handling Computationally Expensive Multiobjective Optimization Problems Using Surrogates: Non-nature Inspired Methods," Struct. Multidiscipl. Optim., **52**(1), pp. 1–25.
[21] Wilson, B., Cappelleri, D., Simpson, T. W., and Frecker, M., 2001, "Efficient Pareto Frontier Exploration Using Surrogate Approximations," Eng. Optim., **2**(1), pp. 31–50.
[22] Su, R., Gui, L., and Fan, Z., 2011, "Multi-objective Optimization for Bus Body With Strength and Rollover Safety Constraints Based on Surrogate Models," Struct. Multidiscipl. Optim., **44**(3), pp. 431–441.
[23] Yang, B. S., Yeun, Y. S., and Ruy, W.-S., 2002, "Managing Approximation Models in Multiobjective Optimization," Struct. Multidiscipl. Optim., **24**(2), pp. 141–156.
[24] Shan, S., and Wang, G. G., 2004, "An Efficient Pareto Set Identification Approach for Multi-objective Optimization on Black-Box Functions," ASME J. Mech. Des., **127**(5), pp. 866–874.
[25] Marler, R. T., and Arora, J. S., 2004, "Survey of Multi-objective Optimization Methods for Engineering," Struct. Multidiscipl. Optim., **26**(6), pp. 369–395.
[26] Cheng, G., Gjernes, T., and Gary Wang, G., 2018, "An Adaptive Aggregation-Based Approach for Expensively Constrained Black-Box Optimization Problems," ASME J. Mech. Des., **140**(9), p. 091402.
[27] Shan, S., and Wang, G. G., 2010, "Metamodeling for High Dimensional Simulation-Based Design Problems," ASME J. Mech. Des., **132**(5), p. 051009.
[28] Raspanti, C., Bandoni, J., and Biegler, L., 2000, "New Strategies for Flexibility Analysis and Design Under Uncertainty," J. Comput. Chem. Eng., **24**(9–10), pp. 2193–2209.
[29] Kreisselmeier, G., and Steinhauser, R., 1979, "Systematic Control Design by Optimizing a Vector Performance Index," IFAC Proceedings Volumes, **12**(7), pp. 113–117.
[30] Poon, N., and Martins, J., 2007, "An Adaptive Approach to Constraint Aggregation Using Adjoint Sensitivity Analysis," J. Struct. Multidiscipl. Optim., **34**(1), pp. 61–73.
[31] Cheng, G. H., Younis, A., Haji Hajikolaei, K., and Gary Wang, G., 2015, "Trust Region Based MPS Method for Global Optimization of High Dimensional Design Problems," ASME J. Mech. Des., **137**(2), p. 021407.
[32] Lloyd, S. P., 1982, "Least Squares Quantization in PCM," IEEE Trans. Inf. Theory, **28**(2), pp. 129–137.
[33] Tizhoosh, H. R., 2005, "Opposition-Based Learning: A New Scheme for Machine Intelligence," International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA-IAWTIC), Vienna, Austria, Nov. 28–30, pp. 695–701.
[34] Schaumann, E., Balling, R., and Day, K., 1998, "Genetic Algorithms with Multiple Objectives," Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, Sept. 2–4, pp. 2114–2123.
[35] Tan, P., Steinbach, M., and Kumar, V., 2006, *Introduction to Data Mining*, Pearson Education, Boston, MA.
[36] Arthur, D., and Vassilvitskii, S., 2007, "k-means++: The Advantages of Careful Seeding," Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, Jan. 7–9, pp. 1027–1035.
[37] Rahnamayan, S., Wang, G. G., and Ventresca, M., 2012, "An Intuitive Distance-Based Explanation of Opposition-Based Sampling," J. Appl. Soft Comput., **12**(9), pp. 2828–2839.
[38] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and da Fonseca, V., 2003, "Performance Assessment of Multiobjective Optimizers—An Analysis and Review," IEEE Trans. Evol. Comput., **7**(2), pp. 117–132.
[39] Zitzler, E., and Thiele, L., 1999, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," IEEE Trans. Evol. Comput., **3**(4), pp. 257–271.
[40] Durillo, J., Nebro, A., Luna, F., Dorronsoro, B., and Alba, E., 2011, "jMetal: A Java Framework for Multi-objective Optimization," Adv. Eng. Softw., **42**(10), pp. 760–771.
[41] Bringmann, K., and Friedrich, T., 2012, "Approximating the Least Hypervolume Contributor: Np-Hard in General, But Fast in Practice," Theor. Comput. Sci., **425**, pp. 104–116.
[42] Nowak, K., Märtens, M., and Izzo, D., 2014, "Empirical Performance of the Approximation of the Least Hypervolume Contributor," International Conference on Parallel Problem Solving From Nature, Ljubljana, Slovenia.
[43] Biscani, F., Izzo, D., and Yam, C., 2010, "A Global Optimization Toolbox for Massively Parallel Engineering Optimization," ICATT 2010: International Conference on Astrodynamics Tools and Techniques, Madrid, Spain, May 3-6.
[44] Izzo, D., 2012, "PyGMO and PyKEP: Open Source Tools for Massively Parallel Optimization in Astrodynamics (the Case of Interplanetary Trajectory Optimization)," Proceedings of the International Conference on Astrodynamics Tools and Techniques—ICATT (2012), Noordwijk, Netherlands.
[45] Lin, S., "NGPM—A NSGA-II Program in Matlab v1.4," https://www.mathworks.com/matlabcentral/fileexchange/31166-ngpm-a-nsga-ii-program-in-matlab-v1-4
[46] Hasanoglu, M. S., "MOFEPSO: Multi-objective feasibility enhanced particle swarm," MATLAB Central File Exchange, 1 Feb 2021, https://www.mathworks.com/matlabcentral/fileexchange/68990-mofepso-multi-objective-feasibility-enhanced-particle-swarm
[47] Zitzler, E., Deb, K., and Thiele, L., 2000, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," Evol. Comput., **8**(2), pp. 173–195.
[48] Hsu, B., Ho, C., Lee, F., and Chen, T., 2010, "A Coreless Technology Overview for Packaging Substrates," 2010 5th International Microsystems Packaging Assembly and Circuits Technology Conference, Taipei, Taiwan, Oct. 20–22, pp. 1–4.
[49] Tang, T., Lan, A., Tsai, J., Chang, I., and Chen, E., 2014, "Flip Chip Packaging With Pre-molded Coreless Substrate," 2014 IEEE 16th Electronics Packaging Technology Conference, Singapore, Dec. 3–5, pp. 200–203.
[50] Hwang, Y., Ou, T., and Su, W., 2017, "2nd International Conference on Precision Machinery and Manufacturing Technology," 2nd International Conference on Precision Machinery and Manufacturing Technology, Kenting, Taiwan, May 19–21.
[51] Lan, C.-Y., 2013, "A Trace-Embedded Coreless Substrate Technique," SiP Global Summit, Taichung.

[52] Chao, S.-H., Hung, C.-P., Chen, M., Lee, Y., Huang, J., Kao, G., and Luh, D.-B., 2015, *An Embedded Trace FCCSP Substrate Without Glass Cloth*, Department of Industrial Design, National Cheng Kung University, Tainan.
[53] Gjernes, T., 2014, "Optimization of Centrifugal Slurry Pumps Through Computational Fluid Dynamics," Master thesis, School of Mechatronic Systems Engineering, Simon Fraser University.
[54] Hock, W., and Schittkowski, K., 1981, *Test Examples for Nonlinear Programming Codes, Secaucus*, Springer-Verlag, New Jersey.
[55] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E., 2005, "Scalable Test Problems for Evolutionary Multiobjective Optimization," *Evolutionary Multiobjective Optimization*, Springer, London, pp. 105–145.
[56] Floudas, C., and Pardalos, P., 1990, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag New York, Inc., New York.