An Adaptive Aggregation-Based Approach for Expensively Constrained Black-Box Optimization Problems

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

Timothy Gjernes
Hevvy/Toyo Pumps North America Corporation,
Coquitlam, BC V3K 7C1, Canada

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

Expensive constraints are commonly seen in real-world engineering design. However, metamodel based design optimization (MBDO) approaches often assume inexpensive constraints. In this work, the situational adaptive Kreisselmeier and Steinhauser (SAKS) method was employed in the development of a hybrid adaptive aggregation-based constraint handling strategy for expensive black-box constraint functions. The SAKS method is a novel approach that hybridizes the modeling and aggregation of expensive constraints and adds an adaptive strategy to control the level of hybridization. The SAKS strategy was integrated with a modified trust region-based mode pursuing sampling (TRMPS) algorithm to form the SAKS-trust region optimizer (SAKS-TRO) for single-objective design optimization problems with expensive black-box objective and constraint functions. SAKS-TRO was benchmarked against five popular constrained optimizers and demonstrated superior performance on average. SAKS-TRO was also applied to optimize the design of an industrial recessed impeller. [DOI: 10.1115/1.4040485]

Introduction

In engineering, computer simulations such as finite element analysis and computational fluid dynamics (CFD) are commonly used for product and process design. However, simulation modeling is often complex and computationally expensive. Using vehicle crash-worthiness modeling as an example, in 2001 Gu stated that "it takes about 36 h for single simulation in SGi Origin 2000" [1]. Duddeck in a 2008 publication stated that "a single crash computation on 8 or 16 CPUs require currently about 12–20 h" [2].

Optimization of finite element analysis and CFD simulations is inherently black-box as the objective and/or constraint functions are output by the simulation, making exact gradient information not readily available or unreliable due to numerical noise. This largely prevents the use of classical gradient-based optimization algorithms that use the finite differencing method to estimate gradients. This is especially the case for optimization problems where the constraints are also outputs of the same simulation for the objective function, which increases the number of simulation runs needed to compute the gradients via finite differencing. Optimization of such simulation-based problems is also expensive as the simulation itself is typically computationally expensive, and simulation runs are needed to compute the objective and/or constraint functions. Expensive black-box optimization is especially challenging, because the computational cost of the functions makes large numbers of function evaluations impractical and the lack of gradient information makes efficient convergence to good solutions challenging. While a variety of evolutionary and nonsurrogate-based global optimization methods [3–6] have been applied to address these problems, metamodel-based design optimization (MBDO) methods such as mode pursuing sampling (MPS) [7], trust region-based MPS (TRMPS) [8], and efficient global optimization (EGO) [9] have demonstrated good performance on expensive black-box optimization problems. MBDO methods use surrogate models such as Kriging and the radial basis function (RBF) to approximate black-box functions, thereby reducing the number of required calls to the black-box functions during optimization [10]. Current MBDO methods can solve rudimentary constrained problems if the search space is not highly constrained and the constraints are not computationally expensive to evaluate. However, when the constraint functions are outputs of the black-box simulation, MBDO methods such as MPS, TRMPS, and EGO perform poorly as they require extremely high numbers of constraint evaluations [11]. Efficient constraint handling strategies are needed when solving expensively constrained problems.

A standard constrained single objective optimization problem has the following mathematical formulation:

$$\begin{aligned} &\min_{x} \ f(x) \\ &\text{s.t. } g_i(x) \le 0, \quad h_j(x) = 0 \\ &i = 1, \ldots, m; \quad j = 1, \ldots, p \end{aligned} \tag{1}$$

where x are the design variables; f is the objective function; g_i are inequality constraints; h_j are equality constraints. Just as there is a broad range of constrained problems, the constraint handling methods in the field are very diverse. Some of the most popular categories of constraint handling methods are penalty functions [11–18], repair methods [19,20], separation of objectives and constraints [21–24], constraint aggregation [25–28], and surrogate modeling of constraints [29–32]. For more detailed surveys of constraint handling methods, consult Refs. [33] and [34]. Penalty functions transform constrained problems into unconstrained problems by allowing the algorithm to sample and retain infeasible solutions, while applying a penalty onto the objective function value based on the constraint violation. These techniques typically relax the problem and prevent optimization algorithms from becoming trapped by infeasible regions [34]. Penalty methods encourage algorithmic search toward feasible areas by augmenting the objective function value of design points with a penalty value for points with violated constraints. Exterior penalty methods are relatively easy to implement and can be applied in a wide variety of algorithms. Kazemi et al. [11] developed a static penalty method that allows the user to predefine a penalty value. Michalewicz and Attia [14] incorporated the concept of simulated annealing with penalties by tuning the penalty coefficients according to constraint violation and increasing the penalty values over

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received November 23, 2017; final manuscript received May 29, 2018; published online June 26, 2018. Editor: Wei Chen.

time. The drawback of penalty methods is the calculation of the penalty to be applied. Penalty values should not be too large, or the objective space will be too distorted for effective optimization over the objective. If penalty values are too small, they will not be effective at encouraging algorithms to discover feasible regions. Adaptive penalty methods address this problem by situational adaptation of the penalty factors [15–17], but general performance for expensive black-box problems is lacking [31]. Repair methods, on the other hand, are typically used for combinatorial problems [34] or problems where domain knowledge can be applied in the design of effective repair heuristics. Separation of objectives and constraints is effective for problems with few constraints, but with large numbers of constraints the method quickly becomes limited by the ability of the multi-objective algorithm in handling large numbers of objectives. In fact, obtaining good performance on problems with four or more objectives is an active and challenging area of research [35]. A more scalable approach is one proposed by Camponogara and Talukdar [23], where a two-objective problem is formulated with one objective being the original objective function and the second objective being a function that aggregates all constraints. Constraint aggregation has shown promise, with the Kreisselmeier and Steinhauser (KS) function [36] being commonly used and demonstrating good performance [25].

Currently, there is a lack of MBDO methods that are designed to handle expensive inequality constraints. A common technique is to use surrogates to model each expensive constraint. Kleijnen et al. used Kriging [29], Regis used RBF surrogates [30], and Rashid et al. applied multiquadric RBF [31]. Rashid et al. stated that "to improve the process for constrained optimization, each expensive nonlinear constraint must be individually modeled alongside the objective function" [31]. By modeling each constraint individually, the feasible and infeasible spaces can be more accurately approximated. While it is simpler to model every expensive constraint, it is premature to assert that no other method is capable of efficiently optimizing problems with expensive constraints. A common alternative is to model all the constraints using a single surrogate (constraint aggregation) or to apply penalties to the objective function. Basudhar et al. developed a support vector machine-based strategy integrated with the EGO method to model the boundary of the feasible space [32]. Holmström et al. used a penalty function to encourage the optimizer to search toward feasible areas [18].

The motivation of this work is to develop an adaptive aggregation-based MBDO method that can optimize high-dimensional expensively constrained black-box problems and adaptively reduce the number of constraints being modeled, thereby eliminating inactive constraints dynamically. The constraint handling method uses the KS function as the aggregation method and is combined with an adaptive strategy to form the situational adaptive Kreisselmeier and Steinhauser (SAKS) method. The SAKS method combined with a modified TRMPS global optimizer forms the SAKS-trust region optimizer (SAKS-TRO). SAKS-TRO is then compared to a set of popular constrained optimizers on a suite of benchmark functions with inequality constraints and applied to the design of an industrial recessed impeller.

Kreisselmeier and Steinhauser Function

The KS function, developed by Kreisselmeier and Steinhauser [36], is a continuous aggregation function with the following formulation:

$$\mathrm{KS}[\boldsymbol{g}(\boldsymbol{x})] = \frac{1}{\rho}\ln\left[\sum_{i=1}^{n} e^{\rho g_i(\boldsymbol{x})}\right] \tag{2}$$

where $\boldsymbol{g}(\boldsymbol{x})$ are the constraint values at $\boldsymbol{x}$ design points, 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. Some useful properties of the KS function summarized by Poon and Martins are [25]:

(1) $\mathrm{KS}(\boldsymbol{x},\rho) \geq \max[\boldsymbol{g}(\boldsymbol{x})]$ for all $\rho > 0$
(2) $\lim_{\rho\to\infty} \mathrm{KS}(\boldsymbol{x},\rho) = \max[\boldsymbol{g}(\boldsymbol{x})]$
(3) $\mathrm{KS}(\boldsymbol{x},\rho_2) \geq \mathrm{KS}(\boldsymbol{x},\rho_1)$ for all $\rho_2 > \rho_1 > 0$
(4) $\mathrm{KS}(\boldsymbol{x},\rho)$ is convex if and only if all constraints are convex

These properties state that the KS function is always greater or equal to the maximum constraint value and that increasing ρ makes the KS function more closely follow the constrained space. The KS function also does not change the convexity of the constrained space [25]. These properties indicate that the KS function can conservatively envelope the feasible space and aggregate any number of constraints into one function. Figure 1 shows the shape of the KS function given different values of ρ. For larger values of ρ, the curves at the corners of the feasible space are narrower.

Radial Basis Function Surrogate

The chosen surrogate method for this work is an RBF, which is constructed using a linear combination of approximating functions using the original function values and distances to the center points $\boldsymbol{x}_i$ [37,38]. The RBF surrogate is a good and computationally inexpensive general purpose approximator that can perform well with a small number of center points [39]. A variety of different $\varphi(.)$ kernels and distance metrics can be used [37], each with different properties. We use a variant composed of a sum of thin plate splines and a linear polynomial based on success with prior work [40]

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n}\beta_i|\boldsymbol{x}-\boldsymbol{x}_i|^2\log|\boldsymbol{x}-\boldsymbol{x}_i| + P(\boldsymbol{x})$$

$$\sum_{i=1}^{n}\beta_i\boldsymbol{p}(\boldsymbol{x}) = 0, P(\boldsymbol{x}) = \boldsymbol{p}\boldsymbol{\alpha} = [p_1, p_2, \ldots, p_q][\alpha_1, \alpha_2, \ldots, \alpha_q]^T \tag{3}$$

where $\boldsymbol{x}_i$ are the evaluated 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.

Situational Adaptive Kreisselmeier and Steinhauser Method

The SAKS method takes advantage of the smooth and conservative enveloping properties of the KS function combined with a

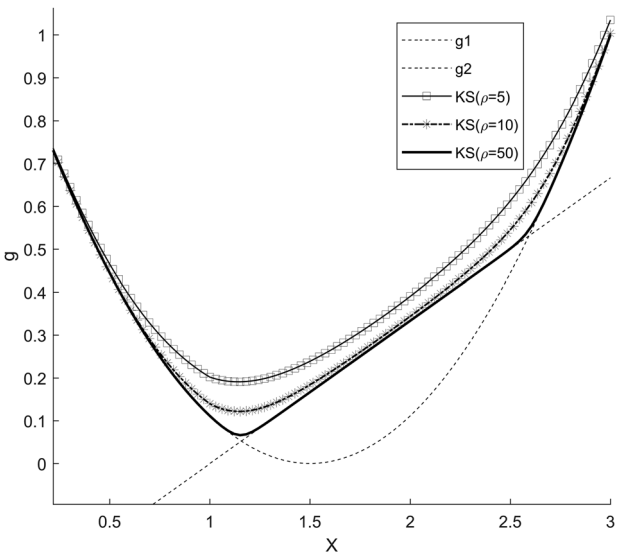

Fig. 1 Kreisselmeier and Steinhauser function of two inequality constraints for increasing ρ

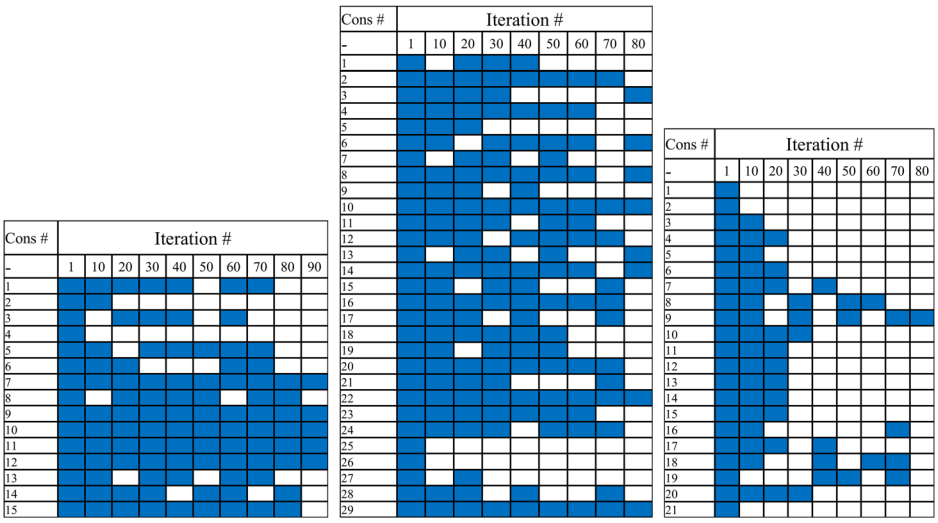

Fig. 2 Constraint classification for the P116 (left), P118 (middle), and beam (right) problems (filled = independent, blank = aggregated)

RBF surrogate to construct aggregated constraint functions that reduce the number of surrogate models required. The smoothness of the KS function makes it suitable for modeling with surrogate models, and its conservativeness helps to ensure feasibility of candidate designs. The SAKS method is composed of three main components, as follows:

(1) classification of constraints for aggregation;
(2) construction of RBF surrogates using KS-aggregated constraints;
(3) situational adaptation of KS function conservativeness.

One of the differences between SAKS and other existing constraint handling methods for MBDO is that 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 constraint violation of historical candidate expensive points as follows:

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

where $\{\boldsymbol{g}\}$ is the full set of expensive inequality constraints; $\{\boldsymbol{g}_{\text{ind}}\}_i$ is the set of expensive inequality constraints that are to be individually modeled in the ith iteration, $\{\boldsymbol{g}_{\text{viol}}\}_{i-n,i-n+1,\ldots,i-1}$ is the set of violated expensive inequality constraints for the last n iterations, and $\{\boldsymbol{g}_{\text{agg}}\}_i$ is the set of expensive inequality constraints that are to be aggregated in the ith iteration. This classification scheme ensures that the historically more troublesome constraints are modeled individually using the surrogate while inactive and irrelevant constraints are aggregated. At the beginning, the strategy is more conservative by individually modeling all expensive inequality constraints. As the optimization progresses, $\{\boldsymbol{g}_{\text{ind}}\}_i$ shrinks as it becomes clearer which constraints are more difficult. To demonstrate this, some sample optimization results using the SAKS-TRO algorithm are collected for the P116, P118, and Beam problems (see the Appendix for problem details). Figure 2 shows the dynamic constraint classification process for the three constrained problems over multiple iterations. Cons # stands for constraint number, and each row represents the aggregation status of an inequality constraint at the specified iteration numbers. At the beginning all constraints are modeled independently but as the optimization progresses more constraints are aggregated. The classification is also determined dynamically according to the immediate circumstances and is reflected in the constantly changing aggregation status of the constraints.

To determine an appropriate value for n, the number of independently modeled constraints was tracked over many iterations for two problems. Figure 3 shows the effect of different values of n on the aggregation pattern across iterations for the P106 problem, and Fig. 4 shows the same comparison for the P118 problem. The results indicate that a smaller value of n is associated with more aggressive aggregation in general. However, even a small value of n can result in many constraints being independently modeled as the optimization progresses, as can be seen after iteration 81 in Fig. 3 with a spike in the number of independent constraints for $n=5$. This shows that the dynamic nature of the aggregation method can mitigate the effects of the selection of n. n is set to 10 in this work as a reasonable tradeoff between aggregation and independent modeling.

Next, the set $\{\hat{\boldsymbol{c}}\}_i$ of RBF surrogates is constructed where $\{\hat{\boldsymbol{c}}_{\text{ind}}\}_i$ is the set of individually modeled RBF surrogate constraints, and $\hat{c}^i_{\text{agg}}$ is the RBF surrogate of the KS aggregate of $\{\boldsymbol{g}_{\text{agg}}\}_i$, all at the ith iteration. Due to the nature of function aggregation $\{\boldsymbol{g}_{\text{agg}}(\boldsymbol{x})\}_i$, which represents the constraint values of $\{\boldsymbol{g}_{\text{agg}}\}_i$ at $\boldsymbol{x}$, needs to be normalized to prevent bias

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

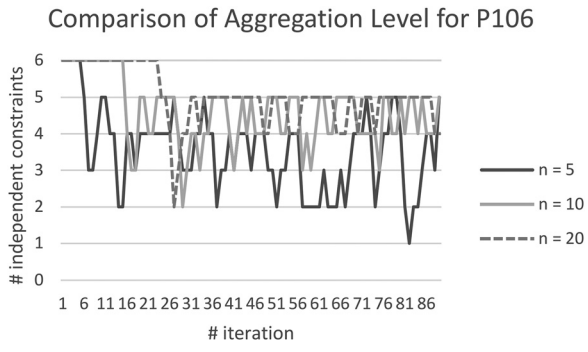

Fig. 3 Comparison of aggregation level across iterations for P106 given different *n* values

where $g^j_{\text{agg}}(x)$ is the output of the jth function in $\{g_{\text{agg}}\}_i$. $\{g_{\text{agg,norm}}(x)\}$ is the set of outputs for all $\{g_{\text{agg}}\}_i$, which is then aggregated into a single function using the KS method. Raspanti et al. [41] recommend that a different formulation of the KS function be used instead of Eq. (2) to avoid numerical errors associated with the use of large values of ρ

$$\text{KS}[g(x)] = g_{\max}(x) + \frac{1}{\rho}\ln\left[\sum_{i=1}^{n} e^{\rho(g_i(x)-g_{\max}(x))}\right] \tag{6}$$

where $g_{\max}(x)$ is the maximum constraint value at design point x. Substituting $g(x)$ for $\{g_{\text{agg,norm}}(x)\}$, Eq. (6) can then be modified to

$$\text{KS}\left[\{g_{\text{agg,norm}}(x)\}\right] = g_{\max}(x) + \frac{1}{\rho}\ln\left[\sum_{j=1}^{n} e^{\rho\left(g^j_{\text{agg,norm}}(x)-g_{\max}(x)\right)}\right]$$
$$g_{\max}(x) = \max\left(\{g_{\text{agg,norm}}(x)\}\right) \tag{7}$$

The resultant KS values, $\text{KS}\left[\{g_{\text{agg,norm}}(x)\}\right]$, represent the maximal normalized constraint violation, with values above 0 meaning the points are predicted to be in an infeasible region. $\text{KS}\left[\{g_{\text{agg,norm}}(x)\}\right]$ is then used to construct $\hat{c}^i_{\text{agg}}$ based on Eq. (7). Poon and Martins [25] recommend a value of 50 for ρ, but in this work ρ is situationally adapted at each iteration as follows:

$$\rho^{k+1} = \begin{cases} 2\cdot\rho^k \text{ if } \{g(x^k)\} \le 0 \text{ or } x^k = \emptyset \\ 0.5\cdot\rho^k \text{ if any } \{g(x^k)\} > 0 \end{cases} \tag{8}$$
$$1 \le \rho \le 8192$$

where k is the current iteration, and x^{k-1} is the previous iteration's candidate design. Because ρ is a parameter that controls the conservativeness of the KS aggregation, this control strategy allows the algorithm to become more conservative when repeated iterations fail to identify feasible designs. The range of 1–8192 gives the ρ term an adaptation range of 13 steps between the min and max values. The parameter will also become less conservative when the algorithm successfully finds feasible designs. One of the drawbacks of function aggregation is a potential reduction in predictive accuracy of the constraint space, because the output of constraint function aggregation is used for modeling with the RBF surrogate. The surrogate may not be able to predict feasibility as well with the aggregated functions compared to individual modeling of the constraint functions. This situational adaptive strategy mitigates this drawback by using conservative constraint boundary modeling to reduce the impact to modeling accuracy of function aggregation. One issue with the adaptation scheme presented in Eq. (8) is that while larger values of ρ allow the KS function to approach the constraint boundary, the KS function never coincides with the actual boundary. This can pose a problem with designs that are at or very close to the constraint boundary as even a large ρ value can cause the KS function to predict infeasibility for a promising design when it is feasible. To address this, the second term in Eq. (7) is set to zero when the value of ρ reaches the maximum value.

Another consideration with the KS function formulation is the variability of the second term in Eq. (7) with larger numbers of aggregated constraints. In theory, very large values are possible which can potentially make the KS function too conservative. Since the maximum and minimum values of $g^j_{\text{agg,norm}}(x)$ and $g_{\max}(x)$ are 1 and −1, respectively,

$$0 < e^{\rho\left(g^j_{\text{agg,norm}}(x)-g_{\max}(x)\right)} \le 1 \tag{9}$$

Taking the limit of the ln term to infinity we get

$$\lim_{n\to\infty}\text{KS}\left[\{g_{\text{agg,norm}}(x)\}\right]$$
$$= g_{\max}(x) + \frac{1}{\rho}\lim_{n\to\infty}\ln\left[\sum_{i=1}^{n} e^{\rho\left(g^j_{\text{agg,norm}}(x)-g_{\max}(x)\right)}\right] = \infty \tag{10}$$

However, from Eq. (9), we know that the inner term of the ln function can only be large if there are a large number of constraints. Specifically

$$\sum_{i=1}^{n} e^{\rho\left(g^j_{\text{agg,norm}}(x)-g_{\max}(x)\right)} = s \le \text{number of constraints} \tag{11}$$

Let s represent the expression inside the ln function, as shown in the equation above. Since ln increases sublinearly relative to the magnitude of s, the magnitude of the output of the ln expression decelerates as s increases in magnitude. For example, given two values of ρ and number of aggregated constraints, respectively, computation of the ln term results in the values shown in Table 1. The conservative behavior of KS exhibits significant variation by varying the two terms, which makes the adaptive behavior of ρ in this work more appropriate than defining a static value.

The advantages of the SAKS method are that it models constraints individually only when it needs to, which reduces unnecessary surrogate modeling of inactive or unimportant constraints, and situationally adapts the conservativeness of the KS aggregator to compensate for the drawbacks of function aggregation in MBDO.

Situational Adaptive Kreisselmeier and Steinhauser–Trust Region Optimizer

This work integrates the SAKS method with a modified TRMPS global optimizer to form the SAKS-TRO algorithm. TRMPS is a global optimization method for problems with expensive black-box functions that relies on stochastic sampling and surrogates to perform optimization. TRMPS [8] is shown to have very competitive performance on unconstrained global optimization problems, requiring comparably few objective function evaluations to obtain good solutions.

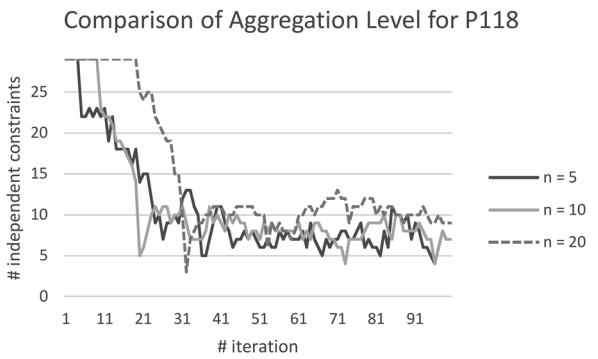

Fig. 4 Comparison of aggregation level across iterations for P118 given different *n* values

Table 1 Effect of varying ρ and number of aggregated constraints on ln(.) term

	Number of aggregated constraints	
ρ	*10*	*100*
1	2.302	4.605
10	0.2302	0.4605

Review of Trust Region-Based Mode Pursuing Sampling. At the beginning of the TRMPS algorithm, a small set of design points is randomly generated and evaluated using the black box. At each iteration, a linear spline RBF surrogate is constructed using expensive points contained within the regions exclusively covered by each of the trust regions. The linear spline RBF surrogate is formulated as follows:

$$\hat{f}(x) = \sum_{i=1}^{m} \alpha_i \|x - x_i\| \tag{12}$$

where m is the number of points in $\{y\}$; α_i is the weight of the function or the objective function value of point x_i; and the set of points x_i are the expensive points used to build the model. Sample points that are located within the trust regions are fitted onto the corresponding surrogate. At the end of each iteration, the trust regions are resized, and the expensive points are resorted into the appropriate trust region. The trust regions are also always centered at the expensive design with the lowest function value.

Trust region-based mode pursuing sampling maintains two trust regions S (TR_S) and B (TR_B) throughout the optimization, where in each iteration a surrogate modeling and discriminative sampling process (or, an MPS process [7]) occurs within the boundaries of each trust region. The two trust regions adapt in size to balance exploitation and exploration to more efficiently sample the problem space. If the search improves through the discovery of a better optimum point, then S expands to avoid being trapped into a local minimum region and B contracts to exploit the promising region. If the search does not improve in a certain number of iterations, S contracts to exploit the promising region and B expands to better explore the search space.

The strategy of the MPS process in each trust region is to use uniform random sampling to generate a small set of expensive points which are then used to create a surrogate of the objective function. In a discriminative sampling process [42], many "cheap" points have their fitness values approximated using the surrogate. The cheap points are then used to selectively pick new expensive points, which are added to the old set of expensive points. The most promising subregions of the search space are identified and a quadratic model constructed in each subregion. Local optimization is performed in each subregion by using the quadratic model as the objective function. The quadratic approximation model is built for the region around the current optimum point (referred to as the current function mode) and is expressed as

$$q = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \beta_{ii} {x_i}^2 + \sum_{i<j} \sum_{j=1}^{n} \beta_{ij} x_i x_j, \tag{13}$$

where $\boldsymbol{\beta}$ are model coefficients. The model coefficients are calculated using the coordinate values of the points in the fitting region by solving the following equation for $\boldsymbol{\beta}$:

$$(\boldsymbol{x}^T \boldsymbol{x})\boldsymbol{\beta} = \boldsymbol{x}^T \boldsymbol{y} \tag{14}$$

The entire process iterates until certain stopping criteria are met.

Process of Situational Adaptive Kreisselmeier and Steinhauser–Trust Region Optimizer. The TRMPS algorithm is complementary to the SAKS method, because both aim to reduce the computational cost of surrogate modeling while maintaining efficient optimization. In TRMPS, the trust regions dynamically reduce the spatial coverage of RBF surrogates, which allows the RBF surrogates to be more accurate within the bounds of the trust regions while also limiting the number of design points used to construct the surrogate models. The trust regions help to limit the possibility of choosing new samples in infeasible space, and thus work in tandem with the situational adaptation strategy of SAKS to dynamically adjust conservativeness and maximize the chance of exploring within the feasible space.

Situational adaptive Kreisselmeier and Steinhauser-trust region optimizer retains the overall strategy of TRMPS with a few key modifications. First, the quadratic model construction and local optimization steps are removed. This was done to reduce the per-iteration cost of the algorithm from four to only two expensive evaluations sampled in two regions. While the local optimization process is useful for optimization without expensive constraints [7], it is much less effective for optimization with expensive constraints. With expensive constraints, additional quadratic approximation models must be constructed for each of the expensive constraints, which further lowers accuracy and reduces the probability of the local optimization process finding an improved design. Furthermore, eliminating the quadratic model construction and local optimization steps and instead dedicating expensive evaluations to the main optimization process with the SAKS constraint handling method proved to be a more efficient use of limited resources.

Second, the linear spline RBF surrogate is replaced with the chosen RBF variant of this work shown in Eq. (3). The sum of thin plate splines was used in the RBF-HDMR metamodeling method and demonstrated good generalized performance [40].

Third, a new mode is added for the case where a feasible design is not available after initial sampling. In this scenario, rather than constructing a surrogate of the objective function, the constraint surrogates are used to evaluate the fitness of candidate designs. Of the predicted constraint values, feasible entries are filtered, and the rest is summed to form a constraint-penalizing merit function, as follows:

$$\boldsymbol{c}_{\text{candidate}} = \begin{cases} \hat{\boldsymbol{c}}_j(\boldsymbol{x}) \text{ if } \hat{\boldsymbol{c}}_j(\boldsymbol{x}) \geq 0 \\ 0 \text{ if } \hat{\boldsymbol{c}}_j(\boldsymbol{x}) < 0 \end{cases} \qquad \hat{f}(\boldsymbol{c}_{\text{candidate}}) = \sum \boldsymbol{c}_{\text{candidate}} \tag{15}$$

where $\hat{\boldsymbol{c}}_j(\boldsymbol{x})$ is the predicted value of the jth constraint surrogate in the set $\{\hat{\boldsymbol{c}}\}_i$. This merit function is used to guide the sampling; once a feasible design has been found, the algorithm reverts to the original behavior.

Situational Adaptive Kreisselmeier and Steinhauser–Trust Region Optimizer Algorithm. We define sets of points $\{y\}$, $\{y_S\}$, and $\{y_B\}$, where $\{y_S\}, \{y_B\} \subseteq \{y\}$ and $\{y_S\} \cap \{y_B\} = \emptyset$. By adding elements to either $\{y_S\}$ or $\{y_B\}$ or any other sets of expensive points, we imply automatic addition of the same elements to $\{y\}$ unless $\{y\}$ already contains those elements. Furthermore, every set of points has a corresponding set of objective and constraint function values where actions on the set of points are mirrored on the sets of objective and constraint function values. Thus, $\{y\}$, $\{y_S\}$, and $\{y_B\}$ have corresponding sets of function values $\{f_y\}$, $\{f_{y_S}\}$, $\{f_{y_B}\}$, and $\{c_y\}$, $\{c_{y_S}\}$, and $\{c_{y_B}\}$. We also define the best design of the ith iteration as y_i, and ρ_S^i and ρ_B^i are the conservativeness parameters for TR_S and TR_B at the ith iteration, respectively.

Step 1: Initial Sampling.
Sample n_{ip} uniform random points $\{y\}$ in the design space, and evaluate $\{y\}$ with the black box function to obtain function values $\{f_y\}$.
For each pt in $\{y\}$,
If pt is within TR_S, place pt in $\{y_S\}$,
Else if pt is within TR_B, place pt in $\{y_B\}$.
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_i.

Step 3: Constraint Surrogates Construction.
Construct the RBF surrogates $\{\hat{\boldsymbol{c}}_{\text{ind}}\}_i$ according to Eq. (3) using expensive points $\{y_S\}$ in TR_S. Construct the aggregate RBF surrogate $\hat{c}^i_{\text{agg}}$ by first aggregating constraint values using Eqs. (5) and (7), then building the RBF model with Eq. (3) and expensive points $\{y_S\}$.

Step 4: Mode Selection.
If a feasible point is present, go to step 5, else, go to step 7.

Step 5: Surrogate Point Accumulation.
Sample a relatively large number of uniform random points, e.g., $n_{0s} = 5000$, in TR_S and fit the ones that satisfy the cheap constraints onto the $\{\hat{\boldsymbol{c}}_{\text{ind}}\}_i$ and $\hat{c}^i_{\text{agg}}$ surrogates. The points that satisfy the constraint surrogates $\{\hat{\boldsymbol{c}}_{\text{ind}}\}_i$ and $\hat{c}^i_{\text{agg}}$ are added to the point set $\{x_S\}$. This process is repeated until a good number of (e.g., 500) feasible samples have been accumulated in $\{x_S\}$.

Step 6: Objective Surrogate Construction.
Construct the RBF surrogate $\hat{f}$ of the objective function using Eq. (3) and expensive points $\{y_S\}$.
Go to step 8.

Step 7: Constraint-Penalizing Merit Function.
Sample $n_{0s} = 5000$ uniform random points in TR_S and fit the ones that satisfy the cheap constraints onto the $\{\hat{\boldsymbol{c}}_{\text{ind}}\}_i$ and $\hat{c}^i_{\text{agg}}$ surrogates. Use Eq. (3) to compute candidate point fitness values.

Step 8: Contour Selection.
Use the discriminative sampling procedure [8] to select a design x_{Si} from $\{x_S\}$. Evaluate x_{Si} with the black box to obtain results $f_{x_{Si}}$ and $\boldsymbol{c}_{\boldsymbol{x}_{Si}}$. Place x_{Si} in $\{y_S\}$.

Step 9: Update SAKS Conservativeness.
Using $\boldsymbol{c}_{\boldsymbol{x}_{Si}}$, update ${\rho_S}^{i+1}$ according to Eq. (8).

Step 10: B region.
Perform steps 2–9 for TR_B.

Step 11: Region Updates.
If the search is improved at this iteration, $R_B = R_B \cdot k_{\text{reduction}}$, $R_S = R_S / k_{\text{reduction}}$.
Else if no improvement for stall iterations, $R_S = R_S \cdot k_{\text{reduction}}$, $R_B = R_B / k_{\text{reduction}}$.

$$\{y_S\}, \{y_B\} = \emptyset$$

For each *pt* in $\{y\}$,
If *pt* is within TR_S, place *pt* in $\{y_S\}$,
Else if *pt* is within TR_B, place *pt* in $\{y_B\}$.

$$i = i + 1$$

Step 12: 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.
Figure 5 is an illustration of the algorithm.

Benchmark Process and Results

Test Methodology. The SAKS-TRO was compared to five different constrained optimization algorithms on a suite of nine different constrained benchmark problems. The five benchmark algorithms are CONMIN [43], SDPEN [44], KSOPT [26], COBYLA [45], and ALPSO [46], the implementations of which are obtained from the pyOpt project [47]. The constrained benchmark problems range from 3 to 30 variables with between 3 and 29 constraint functions and are commonly used in the research literature. 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. For the deterministic algorithms (CONMIN, SDPEN, COBYLA, and KSOPT), the starting point was randomly generated for each run. All deterministic algorithms have been set to use complex step sensitivity where gradients are needed. To test the constraint handling efficiency of the algorithms, all constraints are considered to be computationally expensive in addition to the objective function. In this work, NFE is the total number of function evaluations, with each function evaluation computing the results for all functions.

CONMIN. CONMIN implements the method of feasible directions for linear or nonlinear constrained optimization problems [43]. At each iteration, the algorithm determines a feasible direction and selects a step size that leads to improvement in the objective function.

SDPEN. SDPEN is a derivative-free sequential penalty approach for nonlinear constrained optimization problems. The algorithm extends the sequential penalty approach by leveraging a line search method to replace the use of derivatives and connects the updating of the penalization parameters to the line search. "The algorithm uses a sequence of approximate minimizations of a merit function where penalization of constraint violation is progressively increased" [44].

KSOPT. KSOPT uses the KS function to envelope all constraint functions and the objective, which converts the constrained problem into an unconstrained optimization problem [26]. At each iteration, the Davidon–Fletcher–Powell (DFP) algorithm is used to minimize the KS envelope function.

COBYLA. COBYLA is a derivative-free approach that uses a series of linear approximations to model the objective and constraint functions [45]. The method uses a trust region to manage algorithm convergence. At each iteration, the linear approximations are solved, and the candidate design is evaluated with the actual objective and constraint functions. The method has been limited to a maximum of 30,000 function evaluations during testing as COBYLA often exceeds this limit.

ALPSO. The augmented Lagrangian particle swarm optimizer (ALPSO) uses the augmented Lagrange multiplier approach to handle constraints, which is combined with the basic particle swarm optimizer for nonlinear constrained optimization [46]. The swarm size and hood size parameters have been set to 10, which has resulted in better performance for this algorithm in our testing.

Benchmark Results

Table 2 shows the success rate of each algorithm in achieving feasibility and reaching the target objective value for each problem, while Table 3 contains the results for efficiency of each algorithm for each problem for successful runs. In Table 3, blank fields indicate a failure to achieve any success in the given problems and criteria. This could be due to a programmatic failure, as is often the case with KSOPT, or performance issues. This is shown clearly in Table 2, where programmatic failures are indicated with a "-." In general, SAKS-TRO both achieves feasibility and reaches the target objective faster than CONMIN, COBYLA, KSOPT, SDPEN, and ALPSO. SAKS-TRO also can achieve feasibility and reach the target 100% of the time, while the other four algorithms are not always successful.

In the PV problem, SAKS-TRO achieves feasibility much faster than the other algorithms. It also quickly achieves the target objective. None of the other algorithms could reach the target objective, although all are successful in reaching feasibility. COBYLA consumed an average of 18,389 NFEs and reached the maximum NFE limit in half of all instances. CONMIN, KSOPT, SDPEN, and ALPSO all converged automatically without reaching the target objective. CONMIN consumed an average of 2419 NFEs, KSOPT used 143.6 NFEs, SDPEN used 218.4, and ALPSO used 8272. All four of these algorithms consumed more NFEs than SAKS-TRO without reaching the target. In the Spring

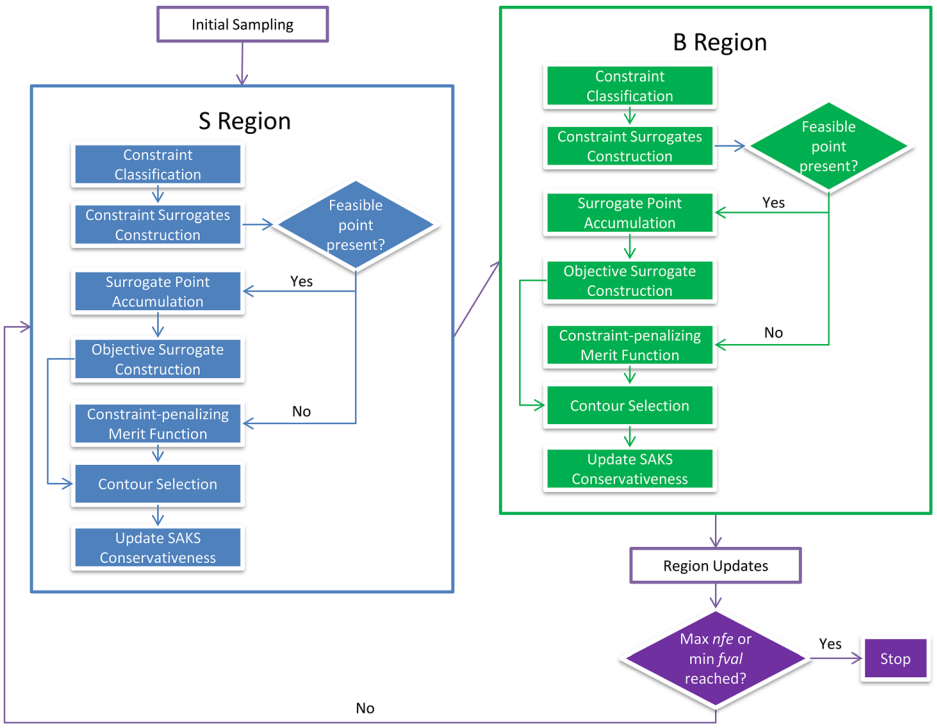

Fig. 5 Situational adaptive Kreisselmeier and Steinhauser-trust region optimizer flowchart

problem, CONMIN performed well and was almost as fast as SAKS-TRO in reaching feasibility. It also reached the objective target almost as fast and with much less variability than SAKS-TRO, although it failed to reach the target objective on a few runs. SDPEN performed well in the successful runs, but failed to reach the target objective most of the time. ALPSO and COBYLA performed poorly. KSOPT was not able to optimize the problem at all due to an exception we consistently encountered when running the compiled code.

In P106, SAKS-TRO was fastest followed by CONMIN, although CONMIN could achieve feasibility in less than half of runs and rarely reached the objective. COBYLA's reliability was similar to CONMIN's, although it was much less efficient at reaching the targets. ALPSO was very reliable, achieving over 90th percentile for both feasibility and objective targets, but consumed many function evaluations. KSOPT and SDPEN performed poorly, with the former achieving feasibility less than a quarter of the time and never reaching the objective target and the latter never achieving feasibility at all.

In P113, COBYLA was slightly faster than SAKS-TRO on average in reaching the objective target, but was slower to reach feasibility and failed to reach the objective target in a few runs. CONMIN was as fast as SAKS-TRO in reaching feasibility, but could only do so in half the runs and was much slower in reaching the objective target. KSOPT was reliable, but was slow in reaching the targets. SDPEN was efficient but only reached the objective target 1/10 of the time, and ALPSO was reliable but generally inefficient.

In P116, only the SAKS-TRO and CONMIN algorithms succeeded to find feasible solutions. CONMIN was only able to reach feasibility 13.3% of the time, and in these runs CONMIN was also able to reach the objective target. KSOPT consistently encountered fatal exceptions, and COBYLA, SDPEN, and ALPSO were unable to find feasible solutions.

In P117, COBYLA outperformed SAKS-TRO by a large margin in reaching the objective targets, but was slower to reach feasibility. KSOPT was much less efficient than both COBYLA and SAKS-TRO, while ALPSO struggled to reach the objective target. CONMIN was unable to reach the objective target at all.

In P118, SAKS-TRO was fastest followed at a distance by COBYLA. SDPEN also performed similarly to COBYLA except it was much less reliable at reaching the objective target. CONMIN and ALPSO were both an order of magnitude slower than SAKS-TRO, COBYLA, and SDPEN. CONMIN also struggled to reach feasibility, and KSOPT was again unable to run due to fatal exceptions.

Table 2 Reliability (success rate) of each algorithm

	SAKS-TRO		CONMIN		COBYLA		KSOPT		SDPEN		ALPSO	
Problem	% runs feasible	% runs got target	% runs feasible	% runs got target	% runs feasible	% runs got target	% runs feasible	% runs got target	% runs feasible	% runs got target	% runs feasible	% runs got target
PV	100	100	100	0.0	100	0.0	100	0.0	100	0.0	100	0.0
Spring	100	100	100	90	96.7	76.7	—	—	86.7	33.3	100	43.0
P106	100	100	48.1	11.1	50.0	7.69	24.1	0.0	0.0	0.0	96.7	93.3
P113	100	100	50.0	46.7	100	93.3	100	76.7	100	10.0	100	73.3
P116	100	100	13.3	13.3	0.0	0.0	—	—	0.0	0.0	0.0	0.0
P117	100	100	100	0.0	100	100	100	100	100	0.0	100	20.0
P118	100	100	26.7	13.3	100	100	—	—	100	63.3	100	23.3
Beam	100	100	50.0	0.0	16.7	6.7	6.7	0.0	100	100	100	63.3
CP15	100	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	63.3	13.3

Table 3 General efficiency benchmark results

Problem		SAKS-TRO		CONMIN		COBYLA		KSOPT		SDPEN		ALPSO	
		NFE to feasible	NFE to target	NFE to feasible	NFE to target	NFE to feasible	NFE to target	NFE to feasible	NFE to target	NFE to feasible	NFE to target	NFE to feasible	NFE to target
PV	Mean	5.8	45.7	36.9	—	51.7	—	15.4	—	39.8	—	11.6	—
	STD	0.9965	11.5345	29.232	—	41.8677	—	10.997	—	45.0526	—	10.875	—
Spring	Mean	6.6	92.8	9.1034	95.9615	41.0357	743.909	—	—	29.64	116.2	7.9655	535.385
	STD	1.6938	77.6137	7.5467	25.1531	136.0312	973.634	—	—	29.4164	50.607	7.1788	279.164
P106	Mean	44.8	781.333	51.857	2935	5466.69	19,312.5	266.57	—	—	—	2284	8358
	STD	15.5484	526.395	15.256	857.603	3609.24	10,116.6	176.98	—	—	—	1659.7	2814.3
P113	Mean	68.2	253.767	70.467	1531.3	80.448	194.37	253.069	1657.4	86.931	305	320.03	2085.2
	STD	38.7088	117.1606	23.814	888.18	53.9349	104.635	94.884	1174.5	23.463	282.501	216.88	649.98
P116	Mean	104.667	402.333	109.75	9009	—	—	—	—	—	—	—	—
	STD	52.8382	254.6607	39.508	10,774.9	—	—	—	—	—	—	—	—
P117	Mean	16.667	407.6	57.6	—	86.833	197.818	151.4	1688.1	61.9667	—	214.4	1665.1
	STD	1.2954	321.3382	37.206	—	19.099	57.0926	70.8435	1238.9	12.5711	—	142.03	1065.2
P118	Mean	35.857	167.821	2840.4	9823.5	250.933	392.875	—	—	359.933	390.684	2936.1	7442.9
	STD	23.4848	205.1056	4624.7	7704.4	74.8612	96.9549	—	—	166.246	129.80	1810.4	3366.4
Beam	Mean	20.467	269.9	67.0	—	130.2	200.5	18.5	—	43.1	49.125	56.6	440.42
	STD	7.9122	82.6594	36.553	—	131.696	226.98	24.749	—	10.9398	7.21	89.674	162.038
CP15	Mean	106.647	502.765	—	—	—	—	—	—	—	—	4960.2	6605.0
	STD	119.4262	440.4862	—	—	—	—	—	—	—	—	2512.3	2130.3

For the beam problem, SDPEN was the fastest by a large margin. SAKS-TRO came next, followed by ALPSO which failed to reach the objective target close to 40% of the time. COBYLA struggled to find feasible solutions, and KSOPT and CONMIN were unable to reach the objective target at all.

For the CP15 problem, the tight constraints caused CONMIN, COBYLA, KSOPT, and SDPEN to fail to find feasible solutions. Although ALPSO found feasible solutions in most runs, it only occasionally reached the target objective while taking over an order of magnitude more NFEs than SAKS-TRO.

Overall, SAKS-TRO demonstrated the highest reliability by always reaching feasibility and the objective targets. SAKS-TRO was also generally the fastest, except for the P117 and beam problems where COBYLA and SDPEN significantly outperformed it, respectively. This shows the potential of hybrid constraint aggregation methods to perform well and demonstrates that it is not always necessary to model all constraint functions individually to achieve good performance on constrained problems.

Design of an Industrial Recessed Impeller. As part of this work, SAKS-TRO is applied to optimize the design of an industrial recessed impeller for use in slurry pumping applications. Slurry pumps are a type of centrifugal pump that are commonly used in the oil and gas, mining, and power generation industries. Slurry is a semiliquid mixture where solid particles are suspended in liquid. Because of the presence of solid particles, slurry pump design is very different from the design of clear water pumps [48]. With slurry, there is more wear on the impeller and other pump components, which means that the blades need to be thicker to prolong life and prevent them from breaking. The blades also need to be fewer and spaced farther apart to allow larger solid particles to pass through. These and other challenges have made the design of slurry pumps a "science [that] is mixed also with a good sprinkling of art and a designer's know-how" [49]. Until recently, design of slurry pumps has relied primarily on engineers' knowledge and experience, and physical prototyping and testing [48].

In this work, we used CFD to simulate a full model of the impeller and volute [50]. ANSYS BLADEMODELER was used to build the impeller and volute geometry. BladeModeler allows specific turbomachinery parameters to be set and then modified, such as blade angles and thickness at varying radius values. ANSYS MESHING was used to generate a mesh of the fluid domain of the impeller and volute. To reduce the cost of computation, the mesh quality was set to achieve a balance of simulation time and accuracy and precision of the results. The mesh was found to give precise results although not as accurate as physical testing. However, the level of accuracy was acceptable for optimization and the overall trends of the simulation outputs were accurate. ANSYS CFX was used to setup and run the simulation. Only a single vane of the impeller was simulated to reduce the cost of computation. This is a common practice to leverage the symmetry of the impeller to reduce simulation complexity and significantly speed up computation. The single vane was contained in a rotating simulation domain with a steady-state interface to the volute which was contained in a stationary simulation domain Fig. 6. That is, the simulation was essentially a snapshot of the flow with the vane at a specific position relative to the volute. CFX simulations were run at three different volumetric flow rates of fluid through the pump to obtain a rough performance curve of the impeller and volute. One flow rate was at approximately the best efficiency point (BEP) where efficiency is at maximum. For the other flow rates, one was much lower and the other much higher than the BEP flow rate. Model analysis was performed on a machine configured with an Intel Core i7 6800K hexacore processor with 32 GB of RAM, and computation speed was further accelerated to an average of 25 min per run via parallel processing using Open MPI.

The objective of the optimization is to maximize the head-rise to shutoff (head drop) of the pump while keeping the head at shutoff and efficiency at different flow rates within certain bands. Head is the ability of the pump to raise water to a certain height while shutoff is the lowest pump flow rate. The head drop is the difference in head between the shutoff (lowest) and the highest flow rates. Head drop is the objective because with a flat head curve, the slurry pump can easily run at higher flow rates with lower efficiency without being noticed, because the desired head is still being achieved. Also, pump components can be damaged

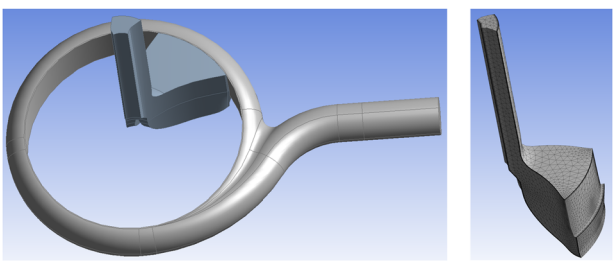

Fig. 6 Fluid domain geometry (left) and single vane mesh (right)

when the flow rate is too high. Thus, a steeper head curve helps ensure the pump is run as intended and reduces the wear damage on pump components. While maximizing the head drop, the efficiency of the pump and the head at shutoff cannot be too low so that the performance of the pump can be maintained. Inequality constraints were used to constrain the efficiency at the mid and high flow rates as well as the head at shutoff within acceptable performance bounds. Constraints are also placed on the output torque 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 flow rates were 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, such as vane shape and vane thicknesses [51]. Eight of the parameters define various aspects of the vane angle and shape, two parameters define the vane thickness, and the last two parameters control the blade height and number of blades. Due to confidentiality, the input parameters and operating velocities are generalized and values are expressed relative to nominal. The following shows the full optimization definition for this engineering problem:

$$
\begin{aligned}
&f(\boldsymbol{x}) = -(H_L - H_H)\\
&g_{1,2}(\boldsymbol{x}) \rightarrow 99\%\ \text{of BHS} \le H_L \le 101\%\ \text{of BHS}\\
&g_3(\boldsymbol{x}) = E_H \ge 83\%\ \text{of BBEP}\\
&g_{4,5}(\boldsymbol{x}) = 94\%\ \text{of BBEP} \le E_M \le 103\%\ \text{of BBEP}\\
&g_{6\rightarrow 11}(\boldsymbol{x}) = 0 \le H\ \text{at all flow rates} \le 2 \cdot \text{BHS}\\
&g_{12,13,14}(\boldsymbol{x}) = \text{torque at all flow rates} \ge 0\\
&g_{15\rightarrow 23}(\boldsymbol{x}) = \text{various mathematical constraints on input parameters}
\end{aligned}
\tag{16}
$$

where BHS is head at shutoff of the baseline design, BBEP is the BEP of the baseline design, H_L and H_H are the head at the low and high flow rates, and E_M and E_H are the efficiency at the mid and high flow rates. In total, there are 12 input parameters, one expensive objective, 14 expensive inequality constraints, and nine cheap constraints.

Due to the high computational cost of the CFD simulation, the optimization was limited to 300 simulation calls which took 150 h. SAKS-TRO took 25 simulation calls before reaching feasibility. Figure 7 shows the head curve comparison between the SAKS-TRO result and the baseline design. It shows that the optimized design has a 75% larger head drop than the baseline while maintaining head values that are close to the baseline for the low and medium flow rates. Figure 8 shows the efficiency curve comparison and shows that the optimized design tracks very closely to

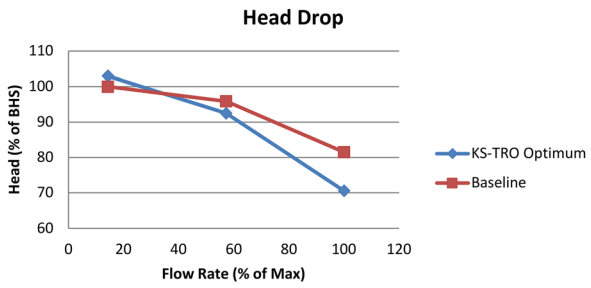

Fig. 7 Head drop comparison between KS-TRO optimum and baseline

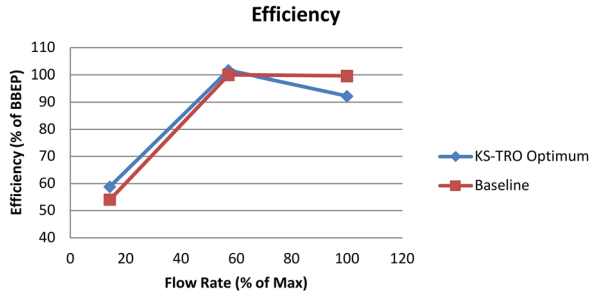

Fig. 8 Efficiency comparison between KS-TRO optimum and baseline

the efficiency curve of the baseline design aside from a small drop in efficiency at the high flow rate.

A modified version of the optimized design was prototyped by Hevvy Pumps and the performance improvement was validated via physical testing using a full-size pump and a clearwater tank. Figure 9 shows the constraint aggregation performance for SAKS-TRO during the impeller optimization for the 14 expensive inequality constraints. At the beginning of the optimization all constraints were aggregated. From 10 iterations to 100 iterations, the first five constraints are primarily independently modeled, which shows their importance to the optimization. The other nine expensive constraints are always aggregated, which indicates that they are rarely violated. The pattern of aggregation demonstrates the usefulness of the SAKS adaptive aggregation strategy for practical expensive constrained optimization.

Final Remarks

This work proposed a SAKS method as a novel expensive constraint handling strategy. The method combines the KS envelope function with RBF modeling to form a hybrid adaptive constraint aggregator that can effectively explore expensively constrained spaces without needing to individually model each expensive constraint function all the time. The trust region-based mode pursuing sampling (TRMPS) algorithm was modified to improve constrained exploration and was combined with the SAKS method to form a new SAKS-TRO single-objective constrained optimizer for expensively constrained black box problems. The SAKS-TRO method represents a significant departure from earlier MBDO methods [31] for expensive constrained optimization with the novel hybrid constraint aggregation approach. The method was benchmarked against five other constrained optimizers and was also used to optimize the design of a slurry pump impeller.

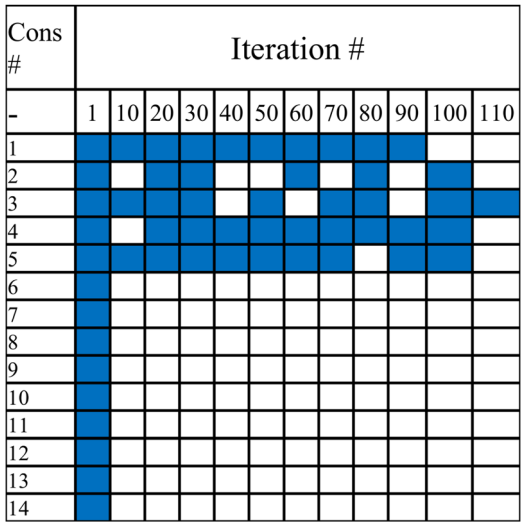

Fig. 9 Constraint classification for impeller optimization (blue = independent, white = aggregated)

The benchmarks show that SAKS-TRO performs well compared to CONMIN, COBYLA, KSOPT, SDPEN, and ALPSO. SAKS-TRO demonstrates a high level of reliability and is generally faster in the benchmarks. The results also show that a constraint aggregation approach when properly applied can be effective in reducing the need to model all expensive constraints individually using metamodels. Such aggregation is useful in cases where there are expensive constraints that are not as tight or as significant in determining feasibility. The results indicate that SAKS-TRO is a promising algorithm for high-dimensional expensive constrained optimization.

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 (Kingston, ON, Canada).

Appendix: Benchmark Problems

Pressure Vessel. The pressure vessel design problem is a four-variable problem with three constraints [7,52]. The cylindrical pressure vessel is constructed from rolled steel plate (carbon steel ASME SA 203 grade B) that are joined welding two forged end caps onto a cylindrical shell. All welds are single-welded butt joints with a backing strip. The pressure vessel should store 750 ft^3 of compressed air at a pressure of 3000 psi. There are four design variables: radius (R) and length (L) of the cylindrical shell, shell thickness (Ts), and spherical head thickness (Th), all of which are in inches

$$\begin{aligned}
&\min F = 0.6224T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R\\
&\text{Subject to } \mathrm{g1} = T_s - 0.0193R \geq 0\\
&\qquad g_2 = T_h - 0.00954R \geq 0\\
&\qquad g_3 = \pi R^2L + (4/3)\pi R^3 - 1.296E6 \geq 0\\
&\qquad R \in [25,\ 150],\ T_s \in [1.0,\ 1.375],\\
&\qquad L \in [25,\ 240],\ T_h \in [0.625,\ 1.0]
\end{aligned} \tag{A1}$$

The target objective value is 7200.

Spring. The spring minimization problem, obtained from Ref. [53], involves specifying the dimensions of a coil spring to withstand axial loading. The objective is to minimize the weight of the spring subject to deflection, stress, surge wave frequency, and diameter constraints. The tension/compression coil spring design problem is a three-variable problem with four constraints [53], where a load is applied axially

$$\begin{aligned}
f(\boldsymbol{x}) &= (x_3+2)x_2x_1^2\\
g_1(\boldsymbol{x}) &= 1 - \frac{x_2^3x_3}{71875x_1^4} \leq 0\\
g_2(\boldsymbol{x}) &= \frac{x_2(4x_2-x_1)}{12566x_1^3(x_2-x_1)} + \frac{2.46}{12566x_1^2} - 1 \leq 0\\
g_3(\boldsymbol{x}) &= 1 - \frac{140.54x_1}{x_2^2x_3} \leq 0\\
g_4(\boldsymbol{x}) &= \frac{x_2+x_1}{1.5} - 1 \leq 0\\
x_1 &\in [0.05, 0.2], x_2 \in [0.25, 1.3], x_3 \in [2, 15]
\end{aligned} \tag{A2}$$

$x_1 = d$, the wire diameter in inches, $x_1 = D$, the mean coil diameter in inches, $x_3 = N$, the number of active coils, g_1 is the deflection constraint, g_2 is the stress constraint, g_3 is the surge wave frequency constraint, and g_4 is the outer diameter constraint. The target objective value is 0.013.

Heat Exchanger Design (P106). The P106 is an eight-variable test problem with six constraints [54] and takes the following form:

$$\begin{aligned}
f(\boldsymbol{x}) &= x_1 + x_2 + x_3\\
g_1(\boldsymbol{x}) &= 0.0025(x_4 + x_6) - 1 \leq 0\\
g_2(\boldsymbol{x}) &= 0.0025(x_5 + x_7 - x_4) - 1 \leq 0\\
g_3(\boldsymbol{x}) &= 0.01(x_8 - x_5) - 1 \leq 0\\
g_4(\boldsymbol{x}) &= 100x_1 + 833.33252x_4 - x_1x_6 - 83333.333 \leq 0\\
g_5(\boldsymbol{x}) &= x_2x_4 + 1250x_5 - x_2x_7 - 1250x_4 \leq 0\\
g_6(\boldsymbol{x}) &= 1,250,000 + x_3x_5 - x_3x_8 - 2500x_5 \leq 0\\
x_1 &\in [1e2, 1e4], \{x_2, x_3\} \in [1e3, 1e4], \{x_4, x_5, x_6, x_7, x_8\} \in [10, 1e3]
\end{aligned} \tag{A3}$$

The target objective value is 8000.

Wong No. 2 (P113). The P113 is a ten-variable test problem with eight constraints from Hock and Schittkowski [54] and takes the following form:

$$
\begin{aligned}
f(\boldsymbol{x}) &= x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\
&\quad + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 \\
&\quad + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \\
g_1(\boldsymbol{x}) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\
g_2(\boldsymbol{x}) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\
g_3(\boldsymbol{x}) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\
g_4(\boldsymbol{x}) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
g_5(\boldsymbol{x}) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
g_6(\boldsymbol{x}) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\
g_7(\boldsymbol{x}) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\
g_8(\boldsymbol{x}) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0 \\
& x \in [-10, 10]
\end{aligned}
\tag{A4}
$$

The target objective value is 40.

Three-Stage Membrane Separation (P116). The P106 is a 13-variable test problem with 15 constraints [54] and takes the following form:

$$
\begin{aligned}
f(\boldsymbol{x}) &= x_{11} + x_{12} + x_{13} \\
g_1(\boldsymbol{x}) &= x_2 - x_3 \leq 0 \\
g_2(\boldsymbol{x}) &= x_1 - x_2 \leq 0 \\
g_3(\boldsymbol{x}) &= 0.002x_7 - 0.002x_8 - 1 \leq 0 \\
g_4(\boldsymbol{x}) &= 50 - x_{11} - x_{12} - x_{13} \leq 0 \\
g_5(\boldsymbol{x}) &= x_{11} + x_{12} + x_{13} - 250 \leq 0 \\
g_6(\boldsymbol{x}) &= 1.262626x_{10} - 1.231059x_3x_{10} - x_{13} \leq 0 \\
g_7(\boldsymbol{x}) &= 0.03475x_2 + 0.975x_2x_5 - 0.00975x_2^2 - x_5 \leq 0 \\
g_8(\boldsymbol{x}) &= 0.03475x_3 + 0.975x_3x_6 - 0.00975x_3^2 - x_6 \leq 0 \\
g_9(\boldsymbol{x}) &= -x_5x_7 + x_1x_8 + x_4x_7 - x_4x_8 \leq 0 \\
g_{10}(\boldsymbol{x}) &= -1 + 0.002(x_2x_9 + x_5x_8 - x_1x_8 - x_6x_9) + x_5 + x_6 \leq 0 \\
g_{11}(\boldsymbol{x}) &= -x_2x_9 + x_3x_{10} + x_6x_9 + 500x_2 - 500x_6 - x_2x_{10} \leq 0 \\
g_{12}(\boldsymbol{x}) &= -x_2 + 0.9 + 0.002(x_2x_{10} - x_3x_{10}) \leq 0 \\
g_{13}(\boldsymbol{x}) &= -x_4 + 0.03475x_1 + 0.975x_1x_4 - 0.00975x_1^2 \leq 0 \\
g_{14}(\boldsymbol{x}) &= -x_{11} + 1.262626x_8 - 1.231059x_1x_8 \leq 0 \\
g_{15}(\boldsymbol{x}) &= -x_{12} + 1.262626x_9 - 1.231059x_2x_9 \leq 0 \\
& \{x_1, x_2, x_3\} \in [0.1, 1], x_4 \in [1e-4, 0.1], \{x_5, x_6\} \\
& \in [0.1, 0.9], \{x_7, x_8\} \in [0.1, 1e3], x_9 \in [500, 1000], \\
& x_{10} \in [0.1, 500], x_{11} \in [1, 150], \{x_{12}, x_{13}\} \in [0.0001, 150]
\end{aligned}
\tag{A5}
$$

The target objective value is 130.

Colville No. 2 (P117). The P117 is a 15-variable test problem with five constraints [54] and takes the following form:

$$
\begin{aligned}
f(\boldsymbol{x}) &= -\sum_{j=1}^{10} b_j x_j + \sum_{j=1}^{5}\sum_{k=1}^{5} c_{kj} x_{10+k} x_{10+j} + 2\sum_{j=1}^{5} d_j x_{10+j}^3 \\
g_j(\boldsymbol{x}) &= 2\sum_{k=1}^{5} c_{kj} x_{10+k} + 3d_j x_{10+j}^2 + e_j - \sum_{k=1}^{10} a_{kj} x_k \geq 0; j = 1, \ldots, 5
\end{aligned}
$$

$$a=\begin{bmatrix}-16&2&0&1&0\\0&-2&0&0.4&2\\-3.5&0&2&0&0\\0&-2&0&-4&-1\\0&-9&-2&1&-2.8\\2&0&-4&0&0\\-1&-1&-1&-1&-1\\-1&-2&-3&-2&-1\\1&2&3&4&5\\1&1&1&1&1\end{bmatrix},\quad b=\begin{bmatrix}-40\\-2\\-0.25\\-4\\-4\\-1\\-40\\-60\\5\\1\end{bmatrix}$$

$$c=\begin{bmatrix}30&-20&-10&32&-10\\-20&39&-6&-31&32\\-10&-6&10&-6&-10\\32&-31&-6&39&-20\\-10&32&-10&-20&30\end{bmatrix},\quad d=\begin{bmatrix}4\\8\\10\\6\\2\end{bmatrix}$$

$$x\in[0,10] \tag{A6}$$

The target objective value is 100.

QLR-P1-2 (P118). The P118 is a 15-variable test problem with 29 constraints [54] and takes the following form:

$$\begin{aligned}f(\boldsymbol{x})&=\sum_{k=0}^{4}\Big(2.3x_{3k+1}+0.0001x_{3k+1}^2+1.7x_{3k+2}\\&\quad+0.0001x_{3k+2}^2+2.2x_{3k+3}+0.00015x_{3k+3}^2\Big)\\&g_{1\text{ to }8}(\boldsymbol{x})\rightarrow 0\le x_{3j+1}-x_{3j-2}+7\le 13\\&g_{9\text{ to }16}(\boldsymbol{x})\rightarrow 0\le x_{3j+2}-x_{3j-1}+7\le 14\\&g_{17\text{ to }24}(\boldsymbol{x})\rightarrow 0\le x_{3j+3}-x_{3j}+7\le 13\\&j=1,\ldots,4\\g_{25}(\boldsymbol{x})&=x_1+x_2+x_3-60\ge 0\\g_{26}(\boldsymbol{x})&=x_4+x_5+x_6-50\ge 0\\g_{27}(\boldsymbol{x})&=x_7+x_8+x_9-70\ge 0\\g_{28}(\boldsymbol{x})&=x_{10}+x_{11}+x_{12}-85\ge 0\\g_{29}(\boldsymbol{x})&=x_{13}+x_{14}+x_{15}-100\ge 0\\&x_1\in[8,21],x_2\in[43,57],x_3\in[3,16],x_{3k+1}\\&\in[0,90],x_{3k+2}\in[0,120],x_{3k+3}\in[0,60]\\&k=1,\ldots,4\end{aligned} \tag{A7}$$

The target objective value is 730.

Cantilever Beam Design (Beam). The beam problem is a 30-variable test problem with 21 constraints and involves minimizing the tip deflection of a stepped cantilever beam subject to constraints. This problem is a modified version of the problem included in Refs. [55] and [56], which was first proposed in Ref. [57].

For testing the proposed algorithm, a cantilever beam with $d=10$ steps is chosen with a $P=50kN$ force on the tip. $E=200$GPa and $\sigma_{\text{allow}}=35\times10^7$Pa are selected as the properties for the used material. For each beam step, there exist three variables for the optimization: width (b_i), height (h_i), and length (l_i) of the beam step. Therefore, 30 input variables exist in this minimization problem and are arrayed in the following order:

$$X=[b_1,h_1,l_1,b_2,h_2,l_2,\ldots,b_{10},h_{10},l_{10}] \tag{A8}$$

The objective is to minimize the tip deflection (δ), expressed as the summation of all the deflections [58]

$$\begin{aligned}\delta&=\int_0^{l_d}\frac{Px_d^2}{EI_d}dx_d+\int_0^{l_{d-1}}\frac{P(x_{d-1}+l_d)^2}{EI_{d-1}}dx_{d-1}+\cdots\\&\quad+\int_0^{l_1}\frac{P(x_1+l_2+l_3+\cdots+l_d)^2}{EI_1}\\&=\frac{Pl_d^3}{3E\left(\frac{b_dh_d^3}{12}\right)}+\frac{P}{3E\left(\frac{b_{d-1}h_{d-1}^3}{12}\right)}\left[(l_{d-1}+l_d)^3-l_d{}^3\right]\\&\quad+\cdots+\frac{P}{3E\left(\frac{b_1d_1^3}{12}\right)}\left[(l_1+l_2+\cdots+l_d)^3-(l_2+l_3+\cdots+l_d)^3\right]\\&=\frac{P}{3E}\sum_{i=1}^{d}\left[\frac{12}{b_ih_i^3}\left(\left(\sum_{j=i}^{d}l_j\right)^3-\left(\sum_{j=i+1}^{d}l_j\right)^3\right)\right]\end{aligned} \tag{A9}$$

where P is the concentrated load, E is the material elasticity modulus, and I_i is the moment of inertia about the neutral axis. There are "$2\times d+1$" constraints in the problem, where d is the number of steps. First, the bending stress in all d steps should be less than the allowable bending stress (σ_{allow}). The constraints for each step beam can be described as

$$\frac{6P\sum_{j=i}^{d}l_j}{b_ih_i^2}\le\sigma_{\text{allow}}\quad i=1,2,\ldots,d \tag{A10}$$

In addition, the aspect ratios (AR) of all cross sections make another set of d constraints that can be shown as

$$\frac{h_i}{b_i}\le\text{AR}\quad i=1,2,\ldots,d \tag{A11}$$

where AR is the aspect ratio. The last constraint is that the total length of the cantilever beam should be more than a specified value

$$\sum_{i=1}^{d}l_i\ge L_{\min} \tag{A12}$$

$L_{\min}$ is the minimum required length that is expected for the beam. In brief, the minimization problem can be rewritten as

$$\delta=f(X)=\frac{P}{3E}\sum_{i=1}^{10}\left[\frac{12}{x_{3i-2}x_{3i-1}^2}\left(\left(\sum_{j=i}^{10}x_{3j}\right)^3-\left(\sum_{j=i+1}^{10}x_{3j}\right)^3\right)\right] \tag{A13}$$

Subject to

$$\begin{aligned}&\frac{6P}{x_{3i-2}x_{3i-1}^2}\sum_{j=i}^{10}x_{3j}\le\sigma_{\text{allow}}\\&\frac{x_{3i-1}}{x_{3i-2}}\le\text{AR}\;i=1,2,\ldots,10\\&\sum_{j=1}^{10}x_{3j}\ge L_{\min}\\&i=1,2,\ldots,10\end{aligned} \tag{A14}$$

AR = 25 and $L_{\min} = 6m$ are selected as the aspect ratio and minimum length for the beam. The following bounds are selected for the variables:

$$\begin{aligned} 0.01\,\text{m} < b_i < 0.05\,\text{m} \\ 0.30\,\text{m} < h_i < 0.65\,\text{m} \\ 0.50\,\text{m} < l_i < 1.00\,\text{m} \\ i = 1, 2, \ldots, 10 \end{aligned} \tag{A15}$$

The target objective value is 0.0120.

CP15. A new test problem with 15 variables and 11 constraints [59] is introduced in this work with the following form:

$$\begin{aligned} f(\boldsymbol{x}) &= x_1(x_2 + x_3) \\ g_1(\boldsymbol{x}) &= 5000x_{10} + 25,000x_{11} - 500x_4 - 2000x_5 \le 0 \\ g_2(\boldsymbol{x}) &= 5000x_{12} + 25,000x_{13} - 500x_6 - 2000x_7 \le 0 \\ g_3(\boldsymbol{x}) &= 5000x_{14} + 25,000x_{15} - 500x_8 - 2000x_9 \le 0 \\ g_4(\boldsymbol{x}) &= x_{12} + x_{13} - x_{14} - x_{15} + 1000 \le 0 \\ g_5(\boldsymbol{x}) &= x_{10} + x_{11} - x_{12} - x_{13} + 1000 \le 0 \\ g_6(\boldsymbol{x}) &= -x_2 + x_4 + x_6 + x_8 \le 0 \\ g_7(\boldsymbol{x}) &= -x_3 + x_5 + x_7 + x_9 \le 0 \\ g_8(\boldsymbol{x}) &= x_1(5000x_{10} + 25,000x_{11} + 5000x_{12} \\ &\quad + 25,000x_{13} + 5000x_{14} + 25,000x_{15}) - 10 \times 10^{10} \le 0 \\ g_9(\boldsymbol{x}) &= -x_1(5000x_{10} + 25,000x_{11} + 5000x_{12} \\ &\quad + 25,000x_{13} + 5000x_{14} + 25,000x_{15}) + 8 \times 10^8 \le 0 \\ g_{10}(\boldsymbol{x}) &= -x_1 \sum_{i=0}^{5} x_{10+i} + 450,000 \le 0 \\ g_{11}(\boldsymbol{x}) &= x_1 \sum_{i=0}^{5} x_{10+i} - 500,000 \le 0 \\ & x_1 \in [10, 50], x_2 \in [21600, 144000], \\ & x_3 \in [3600, 24000], \{x_4, x_6, x_8\} \in [1e3, 144000], \\ & \{x_5, x_7, x_9\} \in [1e3, 24000], \\ & x_{10} \in [500, 3000], x_{11} \in [1e3, 5000], \\ & x_{12} \in [500, 4000], x_{13} \in [1e3, 6000], \\ & x_{14} \in [500, 5000], x_{15} \in [1e3, 7000] \end{aligned} \tag{A16}$$

The target objective value is 6.8×10^6.

References

[1] Gu, L., 2001, "A Comparison of Polynomial Based Regression Models in Vehicle Safety Analysis," ASME Paper No. DAC-21063.
[2] Duddeck, F., 2008, "Multidisciplinary Optimization of Car Bodies," Struct. Multidiscip. Optim., **35**(4), pp. 375–389.
[3] Mitchell, M., 1996, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.
[4] Bertsimas, D., and Tsitsiklis, J., 1993, "Simulated Annealing," Stat. Sci., **8**(1), pp. 10–15.
[5] Poli, R., Kennedy, J., and Blackwell, T., 2007, "Particle Swarm Optimization: An Overview," Swarm Intell., **1**(1), pp. 33–57.
[6] Jones, D., Perttunen, C., and Stuckman, B., 1993, "Lipschitzian Optimization Without the Lipschitz Constant," J. Optim. Theory Appl., **79**(1), pp. 157–181.
[7] Wang, L., Shan, S., and Wang, G., 2004, "Mode-Pursuing Sampling Method for Global Optimization on Expensive Black-Box Functions," J. Eng. Optim., **36**(4), pp. 419–438.
[8] Cheng, G. H., Younis, A., Haji Hajikolaei, K., and Wang, G. G., 2015, "Trust Region Based MPS Method for Global Optimization of High Dimensional Design Problems," ASME. J. Mech. Des., **137**(2), p. 021407.
[9] Jones, D. R., Schonlau, M., and Welch, W. J., 1998, "Efficient Global Optimization of Expensive Black-Box Functions," J. Global Optim., **13**(4), pp. 455–492.
[10] Wang, G. G., and Shan, S., 2006, "Review of Metamodeling Techniques in Support of Engineering Design Optimization," ASME J. Mech. Des., **129**(4), pp. 370–380.
[11] Kazemi, M., Wang, G. G., Rahnamayan, S., and Gupta, K., 2011, "Global Optimization for Problems With Expensive Objective and Constraint Functions," ASME J. Mech. Des., **133**(1) , p. 014505.
[12] Homaifar, A., Qi, C., and Lai, S., 1994, "Constrained Optimization Via Genetic Algorithms," Simulation, **62**(4), pp. 242–254.
[13] Joines, J., and Houck, C., 1994, "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems With GAs," First IEEE Conference on Evolutionary Computation, Orlando, FL, June 27–29, pp. 579–584.
[14] Michalewicz, Z., and Attia, N., 1994, "Evolutionary Optimization of Constrained Problems," Third Annual Conference on Evolutionary Programming, San Diego, CA, Feb. 24–26.
[15] Hadj-Alouane, A., and Bean, J., 1997, "A Genetic Algorithm for the Multiple-Choice Integer Program," Oper. Res., **45**(1), pp. 92–101.
[16] Tessema, B., and Yen, G., 2006, "A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization," IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16–21, pp. 246–253.
[17] Coello, C., 2000, "Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems," Comput. Ind., **41**(2), pp. 113–127.
[18] Holmström, K., Quttineh, N., and Edvall, M., 2008, "An Adaptive Radial Basis Algorithm (ARBF) for Expensive Black-Box Mixed-Integer Constrained Global Optimization," Optim. Eng., **9**(4), pp. 311–339.
[19] Riche, R., and Haftka, R., 1995, "Improved Genetic Algorithm for Minimum Thickness Composite Laminate Design," Compos. Eng., **5**(2), pp. 143–161.
[20] Ullah, A., Sarker, R., and Lokan, C., 2012, "Handling Equality Constraints in Evolutionary Optimization," Eur. J. Oper. Res., **221**(3), pp. 480–490.
[21] Paredis, J., 1994, "Co-Evolutionary Constraint Satisfaction," Third Conference on Parallel Problem Solving From Nature, Jerusalem, Israel, Oct. 9–14.
[22] Parmee, I., and Purchase, G., 1994, "The Development of a Directed Genetic Search Technique for Heavily Constrained Design Spaces," Adaptive Computing in Engineering Design and Control '94, Plymouth, UK, pp. 97–102.
[23] Camponogara, E., and Talukdar, S., 1997, "A Genetic Algorithm for Constrained and Multiobjective Optimization," Third Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA), Vaasa, Finland, pp. 49–62.
[24] Parr, J., Keane, A., Forrester, A., and Holden, C., 2012, "Infill Sampling Criteria for Surrogate-Based Optimization With Constraint Handling," Eng. Optim., **44**(10), pp. 1147–1166.
[25] Poon, N., and Martins, J., 2007, "An Adaptive Approach to Constraint Aggregation Using Adjoint Sensitivity Analysis," J. Struct. Multidiscip. Optim., **34**(1), pp. 61–73.
[26] Wrenn, G., 1989, "An Indirect Method for Numerical Optimization Using the Kreisselmeier–Steinhauser Function," NASA Langley Research Center, Hampton, VA, Contractor Report No. 4220.
[27] Bloss, K. F., Biegler, L. T., and Schiesser, W. E., 1999, "Dynamic Process Optimization Through Adjoint Formulations and Constraint Aggregation," Ind. Eng. Chem. Res., **38**(2), pp. 421–432.
[28] Jiang, Z., Cheng, G. H., and Wang, G. G., 2013, "Mixed Discrete and Continuous Variable Optimization Based on Constraint Aggregation and Relative Sensitivity," ASME Paper No. DETC2013-12668.
[29] Kleijnen, J., van Beers, W., and van Nieuwenhuyse, I., 2010, "Constrained Optimization in Expensive Simulation: A Novel Approach," Eur. J. Oper. Res., **202**(1), pp. 164–174.
[30] Regis, R., 2011, "Stochastic Radial Basis Function Algorithms for Large Scale Optimization Involving Expensive Black-Box Objective and Constraint Functions," J. Comput. Oper. Res., **38**(5), pp. 837–853.
[31] Rashid, K., Ambani, S., and Cetinkaya, E., 2013, "An Adaptive Multiquadric Radial Basis Function Method for Expensive Black-Box Mixed-Integer Nonlinear Constrained Optimization," Eng. Optim., **45**(2), pp. 185–206.
[32] Basudhar, A., Dribusch, C., Lacaze, S., and Missoum, S., 2012, "Constrained Efficient Global Optimization With Support Vector Machines," Struct. Multidiscip. Optim., **46**(2), pp. 201–221.
[33] Kramer, O., 2010, "A Review of Constraint-Handling Techniques for Evolution Strategies," Appl. Comput. Intell. Soft Comput., **2010**, pp. 1–11.
[34] Coello, C., 2002, "Theoretical and Numerical Constraint-Handling Techniques Used With Evolutionary Algorithms: A Survey of the State of the Art," Comput. Methods Appl. Mech. Eng., **191**(11–12), pp. 1245–1287.
[35] Lucken, C., Baran, B., and Brizuela, C., 2014, "A Survey on Multi-Objective Evolutionary Algorithms for Many-Objective Problems," Comput. Optim. Appl., **58**(3), pp. 707–756.
[36] Kreisselmeier, G., and Steinhauser, R., 1979, "Systematic Control Design by Optimizing a Vector Performance Index," IFAC Proceedings Volumes, **12**(7), pp. 113–117.
[37] Powell, M., 1987, *Radial Basis Functions for Multivariable Interpolation: A Review of Algorithms for Approximation*, 3rd ed., Clarendon Press, Oxford, UK.
[38] Jin, R., Chen, W., and Simpson, T., 2001, "Comparative Studies of Metamodeling Techniques Under Multiple Modeling Criteria," Struct. Multidiscip. Optim., **23**(1), pp. 1–13.
[39] Chen, V., Tsui, K. L., Barton, R. R., and Meckesheimer, M., 2006, "A Review on Design, Modeling and Applications of Computer Experiments," IIE Trans., **38**(4), pp. 273–291.
[40] Shan, S., and Wang, G. G., 2010, "Metamodeling for High Dimensional Simulation-Based Design Problems," ASME J. Mech. Des., **132**(5), p. 051009.

[41] 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.
[42] Fu, J., and Wang, L., 2002, "A Random-Discretization Based Monte Carlo Sampling Method and Its Applications," Methodol. Comput. Appl. Probab., **4**(1), pp. 5–25.
[43] Vanderplaats, G., 1973, "CONMIN—A Fortran Program for Constrained Function Minimization," NASA Ames Research Center, Moffett Field, CA, Technical Memorandum No. TM X-62282.
[44] Liuzzi, G., Lucidi, S., and Sciandrone, M., 2010, "Sequential Penalty Derivative-Free Methods for Nonlinear Constrained Optimization," SIAM J. Optim., **20**(5), pp. 2614–2635.
[45] Powell, M., 1994, "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation," *Advances in Optimization and Numerical Analysis*, Springer, Dordrecht, The Netherlands, pp. 51–67.
[46] Jansen, P., and Perez, R., 2011, "Constrained Structural Design Optimization Via a Parallel Augmented Lagrangian Particle Swarm Optimization Approach," Int. J. Comput. Struct., **89**(13–14), pp. 1352–1366.
[47] Perez, R., Jansen, P., and Martins, J., 2012, "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," Struct. Multidiscip. Optim., **45**(1), pp. 101–118.
[48] Gjernes, T., 2014, "Optimization of Centrifugal Slurry Pumps Through Computational Fluid Dynamics," Masters thesis, Simon Fraser University, Burnaby, BC, Canada.
[49] Grzina, A., Roudnev, A., and Burgess, K. E., 2002, *Slurry Pumping Manual*, Warman International, Glasgow, Scotland.
[50] Shah, S., Jain, S., Patel, R., and Lakhera, V., 2012, "CFD for Centrifugal Pumps: A Review of the State-of-the-Art," Chemical, Civil, and Mechanical Engineering Tracks of Third Nirma University International Conference, Ahmedabad, India, Dec. 6–8, pp. 715–720.
[51] Roudnev, A., 1999, "Some Aspects of Slurry Pump Design," World Pumps, **1999**(389), pp. 58–61.
[52] Wilde, D., 1978, *Globally Optimal Design*, Wiley, New York.
[53] Arora, J., 2004, *Introduction to Optimum Design*, Elsevier Academic Press, San Diego, CA.
[54] Hock, W., and Schittkowski, K., 1981, *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, Secaucus, NJ.
[55] Hsu, Y., Wang, S., and Yu, C., 2003, "A Sequential Approximation Method Using Neural Networks for Engineering Design Optimization Problems," Eng. Optim., **35**(5), pp. 489–511.
[56] Thanedar, P., 1995, "Survey of Discrete Variable Optimization for Structural Design," J. Struct. Eng., **121**(2), p. 301.
[57] Hajikolaei, K. H., Pirmoradi, Z., Cheng, G. H., and Wang, G. G., 2014, "Decomposition Based on Quantified Variable Correlations Uncovered by Metamodeling for Large Scale Global Optimization," Engineering Optimization, **47**(4), pp. 429–452.
[58] Schutte, J., and Haftka, R., 2010, "Global Structural Optimization of a Stepped Cantilever Beam Using Quasi-Separable Decomposition," Eng. Optim., **42**(4), pp. 347–367.
[59] Sagar, A., and Gadhvi, B., 2017, *Constrained Nonlinear Optimization Problems: Formulation and Solution*, Simon Fraser University, Surrey, BC, Canada.