EMAS Engineering Modelling, Analysis & Simulation
THE NAFEMS JOURNAL

NAFEMS

Vol. 1, Issue 1, 2024

Model-based Design Optimization Taking into Account Design Viability via Classification

M. Niehoff[a,1], D. Bestle[a], P. Kupijai[b]

[a]Brandenburg University of Technology, 03046 Cottbus, Germany
[b]Rolls-Royce Deutschland Ltd & Co KG, 15827 Blankenfelde-Mahlow, Germany

Abstract

Design optimization of real-world industrial products is usually a challenging high-dimensional task with several multi-modal objectives. Therefore, the solution has to be found by global optimization algorithms which require fast surrogate models to realize a large number of design evaluations. However, approximating the original optimization criteria by surrogates may mislead the optimization by offering solutions in the entire design domain, even if designs are not viable in reality. Therefore, a classification model should be used as additional optimization constraint to guide the optimizer to viable results.

Keyword

aerospace, aero engines, design optimization, viability, surrogate models, ROM

1 Introduction

Aero engines (Figure 1) are complex technical systems with generally non-linear relations between many design parameters $\boldsymbol{x} = [x_1, \dots, x_D]^T$, $D \gg 1$, and some design criteria $\boldsymbol{y}(\boldsymbol{x})$, usually requiring costly numerical analyses in the aero engine design process. Therefore, optimization of aero engines w.r.t. criteria like specific fuel consumption (SFC) and mass is rather challenging.

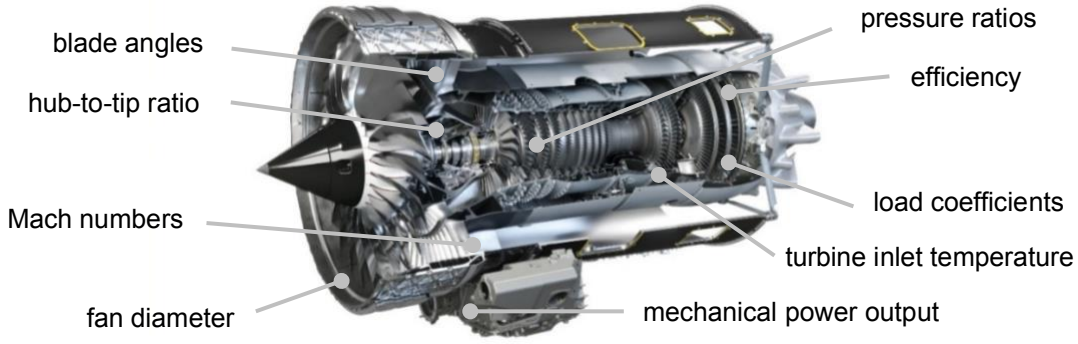

Figure 1. Cutaway of an aero engine and a small selection of potential design parameters.

[1]Corresponding author.
E-mail address: niehoff@b-tu.de (M. Niehoff)
https://doi.org/10.59972/c7b5hzx7

Fortunately, the challenges associated with optimization of complex and costly systems have been the subject of active research. The starting point is usually the substitution of the original costly-to-evaluate process by fast-to-evaluate surrogate models to perform surrogate-assisted optimization (SAO). These surrogates are trained with a relatively small set of originally evaluated support points to approximate input-output relations. The ongoing development using a plurality of different techniques is repeatedly summarized (see e.g. [1], [2]).

Using surrogates enables a variety of different approaches for optimization like evolutionary algorithms (EA), which usually need numerous evaluations to search for globally optimal designs. To reduce the amount of costly training samples, SAO is typically combined with adaptive re-sampling techniques, which sequentially evaluates promising new samples (e.g. [3]). Rather simple re-sampling methods are using the suggested points of the optimizer, while others assess suggested candidate designs w.r.t. dominance and convergence before costly re-evaluation [4]. Kriging-based SAO like Efficient Global Optimization (EGO) [5] deliver predictions of criterion values (exploitation) as well as uncertainties of these predictions which are utilized to determine promising new candidates (exploration). High dimensionality of the design space generally adversely affects the SAO process (*curse of dimensionality*). Therefore, different techniques to reduce the dimensionality have been investigated in this context (e.g. [6]-[8]). Additionally to optimization criteria, surrogates may also be used to identify boundaries of the feasible design space by approximating feasibility constraints (e.g. [9]).

Despite the numerous developments and investigations, an important aspect, which has not yet been addressed sufficiently in the literature, is the viability of designs when using SAO. Here, the term viability describes whether a certain design can (viable design) or cannot (non-viable design) be evaluated by the original process (physical experiment or numerical simulation). Within the context of optimization, viable designs will return values for optimization criteria, whereas non-viable designs cause the experiment or simulation to fail. It should be noted that the term viability is fundamentally different from the term feasibility, which only means that all known design constraints are satisfied and presumes computability (viability) of these functions.

Using SAO faces two challenges when dealing with viability: (i) since a trained surrogate model typically provides solutions for every design, even if it is not viable in reality, a usual SAO without incorporating an additional information about the viability may be misled to pseudo-optimal, non-viable designs; (ii) adaptive re-sampling techniques can be adversely affected or even stall if new designs are proposed regardless of their viability and cannot be evaluated to update the surrogate. Forrester et al. [10] treat non-viable designs as missing data and use data imputation in combination with Kriging-based surrogates to guide new samples towards viable designs. An additional Kriging-based surrogate for the EGO framework approximating the possibility of a design to be viable was investigated by He et al. for single- [11] and variable-fidelity [12] simulations. Sacher et al. [13] use a classification surrogate model to estimate viability which is incorporated as an inequality constraint into a single objective EGO framework. However, Kriging-based SAO has disadvantages when dealing with very high design dimensions, which is why approaches that are more general may be required.

This paper takes up some of the previously described concepts like the classification of viability and, compared to a usual SAO, its incorporation as an additional constraint into the optimization process. In order to focus this investigation to the viability issue, no other constraints are applied in the optimization process on purpose. Unlike other papers, it avoids Kriging and uses a regression-based bi-criterion optimization concept coupling an EA with a classification surrogate to address the viability issue without data imputation. The approach is applied to an industrial preliminary aero engine design process, which offers several hundred of design parameters. This enormous design degree of freedom could be limited to 117 design parameters by experience, which, however, is still a challenging amount of parameters. The original process combines rather basic numerical simulations and other calculations to combine several engine components to a simple holistic engine model which on average needs about 10 minutes to evaluate a design – or to fail. The intention of this paper is to demonstrate the workability of the introduced framework and methods for a real-world high-dimensional application. Here, the aero engine design is optimized w.r.t. the criteria $\boldsymbol{y} = [SFC, m]^T$, where SFC is the specific fuel consumption and m is the engine mass. The structure of the paper is as follows: Section 2 describes the general procedure and terms of using surrogate models for regression and classification. Section 3 finds surrogates for both optimization criteria (SFC, mass) and the only constraint (viability) using all 117 design parameters. Section 4 reduces the dimensionality based on global sensitivity analysis. Finally, Section 5 carries out a model-based design optimization using the reduced set of parameters and incorporating the additional information about viability as a constraint.

2 Substituting the original analysis by surrogate models

Fast-to-evaluate surrogate models are useful for approximating both continuous relations (e.g. regression) and categorical outcomes (classification) of expensive physical experiments or numerical simulations. This section will introduce the basic steps and terms. Two different types of surrogate models are used in this paper, which are taken from the Python library *Scikit-learn* [14]: (i) multi-layer perceptron and (ii) Extra-trees.

2.1 Design of Experiments and Scaling

The substitution usually starts with a design of experiments (DoE) for the design parameters $\boldsymbol{x}$ and their evaluation $\boldsymbol{y}(\boldsymbol{x})$ using the original costly analysis process, which results in samples

$$\left(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\right) \; where \; \boldsymbol{y}^{(n)} = \boldsymbol{y}\left(\boldsymbol{x}^{(n)}\right), n = 1 \dots N. \tag{1}$$

To address the viability of a design $\boldsymbol{x}$, an additional (binary) criterion $v(\boldsymbol{x})$ is introduced as

$$\left(\boldsymbol{x}^{(n)}, v^{(n)}\right) \; where \; v^{(n)} = v\left(\boldsymbol{x}^{(n)}\right) \in \{0,1\}, n = 1 \dots N, \tag{2}$$

where $v^{(n)} = 1$ represents a viable and $v^{(n)} = 0$ a non-viable design. To create a space-filling DoE, Latin Hypercube (LHC) sampling [15] is applied, generating two separate data sets, one for training with $N_{train} = 2000$ including the baseline design, and one for testing with $N_{test} = 1499$ sample points. For both data sets, about 53% of the designs $\boldsymbol{x}^{(n)} \in \mathcal{X}$ belong to the viable subset $\mathcal{X}_v \subseteq \mathcal{X}$, the remaining to its non-viable complement $\bar{\mathcal{X}}_v = \mathcal{X} \backslash \mathcal{X}_v$.

To improve training, it is advisable to normalize the input parameters and to standardize the optimization criteria. Given the minimum $\check{x}_d = min_n\left(x_d^{(n)}\right)$ and maximum $\hat{x}_d = max_n\left(x_d^{(n)}\right)$ of the samples within the training DoE of a design parameter x_d, $d \in \{1,2,\dots,\mathrm{D}\}$, the min-max normalization of each sample from both the training- and test-set is

$$x_d := (x_d - \check{x}_d)/(\hat{x}_d - \check{x}_d), d = 1, \dots, D. \tag{3}$$

Given the arithmetic mean $\bar{y}$ and standard deviation s of the evaluated samples within the training DoE, the standardization of each evaluated sample from both the training- and test-set is

$$y := (y - \bar{y})/s. \tag{4}$$

The normalized training-DoE as well as its evaluated and standardized (viable) outcomes are summarized in Figure 2a, and Figure 2b, respectively. Figure 2c shows the proportions of viable (True) and non-viable (False) samples from the training- and test-set.

2.2 Regression analysis

To substitute functional relations between inputs $\boldsymbol{x}$ and continuous outcomes $y(\boldsymbol{x})$ of a real experiment or simulation with an approximation error ε, regression analysis uses the model

$$\hat{y}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\beta}) = y(\boldsymbol{x}) + \varepsilon \tag{5}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are some model specific parameters; $\boldsymbol{\theta}$ are the parameters to be 'learned' during training, whereas $\boldsymbol{\beta}$ are hyper-parameters to be adjusted in a detached procedure called *model selection* [16]. To select proper hyper-parameters, *cross-validation* will be used and explained later.

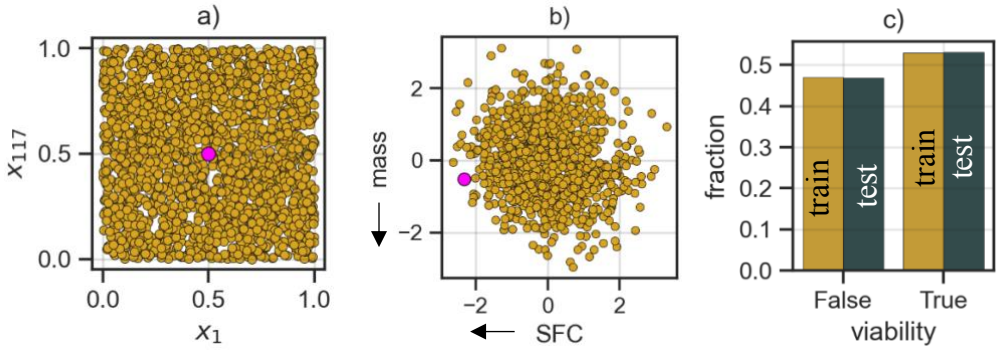

Figure 2. DoE for aero engine design: a) all designs as 2D-projection in normalized design space (yellow: LHC, magenta: baseline) and b) viable designs in standardized criterion space as well as c) proportions of viable designs in training- and test-samples.

The term training expresses the process that finds a priori unknown values $\boldsymbol{\theta}$ by minimizing a loss-function, e.g. the mean squared error (MSE)

$$L_{reg} = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} \left(y_{train}^{(n)} - \hat{y}_{train}^{(n)} \right)^2 \tag{6}$$

where $y_{train}^{(n)} = y\left(\boldsymbol{x}^{(n)} \in \mathcal{X}_{train}\right)$ and $\hat{y}_{train}^{(n)} = \hat{y}\left(\boldsymbol{x}^{(n)} \in \mathcal{X}_{train}\right)$ are the analysed outcomes (1) and predicted outcomes (5) of the training set, respectively.

To assess the model quality during the model selection process as well as to test a finally selected surrogate model, the coefficient of determination

$$R^2 = 1 - \sum_{n=1}^{N_{test}} \left(y_{test}^{(n)} - \hat{y}_{test}^{(n)} \right)^2 / \sum_{n=1}^{N_{test}} \left(y_{test}^{(n)} - \bar{y}_{test} \right)^2 \tag{7}$$

may be applied as metric, where $y_{test}^{(n)} = y\left(\boldsymbol{x}^{(n)} \in \mathcal{X}_{test}\right)$ are the true values and $\hat{y}_{test}^{(n)} = \hat{y}\left(\boldsymbol{x}^{(n)} \in \mathcal{X}_{test}\right)$ are the predicted values of the test set, respectively; $\bar{y}_{test}$ refers to the arithmetic mean of the evaluated test-data.

2.3 Classification

To determine the strictly binary property of a design $\boldsymbol{x}$ to be viable, a probabilistic approach is used here to estimate the membership of $\boldsymbol{x}$ belonging to the domain $\mathcal{X}_v$ of viable designs:

$$p_v(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\beta}) = \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_v] \in [0,1]. \tag{8}$$

A certain design $\boldsymbol{x}$ is then estimated to be viable, i.e., $\hat{v}(\boldsymbol{x}; p_{th}) = 1$, if p_v is above some cut-off threshold $p_{th} \in [0,1]$, i.e., $p_v(\boldsymbol{x}) \geq p_{th}$.

For classification, the training process differs from regression in terms of the loss-function. Here, the cross-entropy will be minimized:

$$L_{class} = -\frac{1}{N_{train}} \sum_{n=1}^{N_{train}} \left[p_{v,train}^{(n)} \, log \, p_{v,train}^{(n)} + \left(1 - p_{v,train}^{(n)}\right) log\left(1 - p_{v,train}^{(n)}\right) \right]. \tag{9}$$

To test or validate a classification model, the confusion matrix (see e.g. [17]) may be considered. Depending on a certain value for cut-off threshold p_{th}, the predictions $\hat{v}^{(n)} = 1$ ($P \triangleq$ positive) or $\hat{v}^{(n)} = 0$ ($N \triangleq$ negative) for test-data can be correct ($T \triangleq$ true) or wrong ($F \triangleq$ false) resulting in a confusion matrix of combinations TP, TN, FP, FN. The approximation quality may then be assessed, as a function of p_{th}, by the receiver operating characteristics (ROC) (see e.g. [17]) curve, which is a plot of the true positive rate $TPR = TP/(TP + FN)$ vs. the false positive rate $FPR = FP/(FP + TN)$. Especially the area under the ROC curve ($ROC\ AUC$) typically serves as a scalar metric to assess the classification model.

2.4 Cross-validation for model selection

Cross-validation (CV) is a frequently used re-sampling method in the process of model selection [16] to find the most appropriate set of hyper-parameters (see e.g. [18] [19]). Given a specific set of samples and hyper-parameters, CV re-samples the data into training and validation subsets. Here, repeated k-fold CV is applied for regression to find models with both low variance and bias [20]. The k-fold CV randomly splits the given data set into k disjoint and equally sized subsets. Then, the surrogates are trained on the union of $(k-1)$ subsets and applied to the remaining subset to get a performance measure ((7) in case of regression or $ROC\ AUC$ in case of classification). This procedure is repeated k-times, whereby a different subset is used for assessment in each repetition. The average of the k performance measures then serves as final quality measure. Repeated k-fold CV randomly repeats this procedure m times resulting in a final performance measure as the average over $k \times m$ training-validation repetitions. Throughout this paper, $k = 10$ and $m = 20$ is chosen, which results in $10 \times 20 = 200$ repetitions. For classification repeated stratified k-fold CV (see e.g. [19]) is applied which similarly splits the data sets into k subsets, but accounts for a consistent proportion between both classes in each subset.

2.5 Multi-layer Perceptron

A multi-layer perceptron model (MLP, see e.g. [21]) is a commonly used, fully connected, feed-forward artificial neural network that consist of $R \geq 2$ layers of neurons. The first layer is called *input layer*, the last one *output layer*. The remaining $(R-2)$ layers are called *hidden layers*. Each layer $r \in \{1, \dots, R\}$ consists of an individual number J_r of neurons, where the number of neurons in the input ($r = 1$) and output layer ($r = R$) are equal to the number of input variables and output targets, respectively. The inputs of a single neuron $j_r \in \{1, \dots, J_r\}$ within the layers $r \in \{2, \dots, R\}$ are the weighted outputs from all the neurons in the precursory layer $(r-1)$. The term *neuron* describes the process of two consecutive

mathematical operations: (1) calculate the sum $\bar{S}_{r,j_r}$ of all inputs and (2) apply a non-linear activation function $a(\bar{S}_{r,j_r})$ to get the output:

$$S_{r,j_r} = a\left(\theta_{r,0,j_r} + \sum_{j_{r-1}=1}^{J_{r-1}} S_{(r-1,j_{r-1})} \cdot \theta_{r,j_{r-1},j_r}\right) \tag{10}$$

where $\theta_{r,0,j_r}$ is a bias and θ_{r,j_{r-1},j_r} are the weights of the input of neuron j_r in layer r coming from neuron j_{r-1} in layer $(r-1)$. The number of neurons in the hidden layers is an important hyper-parameter and part of the model-selection process. To improve the training of the MLP model, the loss functions (6) and (9) are expanded by a L2-regularization term resulting in

$$L_{MLP} = L_l + \lambda \frac{1}{2N_{train}} \|\boldsymbol{\theta}\|_2^2, l \in \{reg, class\} \tag{11}$$

with λ as additional hyper-parameter for tuning the strength of the regularization. During CV, the following hyper-parameters are investigated in a full-factorial combination to find an appropriate MLP-model: $J_r \in \{2^0, 2^1, 2^2, 2^3, 2^4\}$ and $\lambda \in \{2^{-5}, 2^{-4}, \dots, 2^5\}$. Thus, $5 \times 11 = 55$ different MLP models are investigated during CV. The number of layers is fixed to the value $R = 3$, i.e., only a single hidden layer is used. For regression, hyperbolic tangent is used as activation function, for classification the sigmoid function is used.

2.6 Extremely randomized trees

Extremely randomized trees (Extra-Trees, ET) [22] is a tree-based ensemble method that combines a number of randomly grown (decision- or regression-) trees. As with classical tree-based ensembles, both the variable as well as the point to split each variable at each node to minimize the loss-function are determined during training. In contrast to classical tree-based ensembles, the point to split each of the selected variables at each node is selected completely at random. The final output of the whole ensemble is calculated here by the average over all individual trees. Important hyper-parameters are the number of trees M within the ensemble, the relative amount K of randomly selected variables at each node to search for the best split as well as the minimum number of samples required to split a node n_{min}. During CV, the following hyper-parameters are investigated in a full-factorial combination to find an appropriate ET-model: $M \in \{200,400,800\}$, $K \in \{0.1, 0.2, \dots, 1.0\}$ and $n_{min} \in \{2, 4, 8, 16\}$. Thus, $3 \times 10 \times 4 = 120$ different ET models are investigated during CV.

3 Surrogate models using all parameters

At first, all 117 design parameters of the aero engine are used to find appropriate surrogate models for the three subtasks: (1) regression for SFC, (2) regression for mass and (3) classification of viability. The models are trained, validated and tested with respect to the metrics described in Section 2.

Figure 3 shows the results of the CV process using the training data. The blue circles represent MLP models for all combinations of hyper-parameters mentioned in subsection 2.5, whereas orange squares represent ET models for the hyper-parameter combinations given in subsection 2.6. The results are sorted with respect to metrics R^2 or $AUC\ ROC$, respectively. For the application here, the better results for regression are obtained by MLP models, whereas the better results for classification are achieved by ET models.

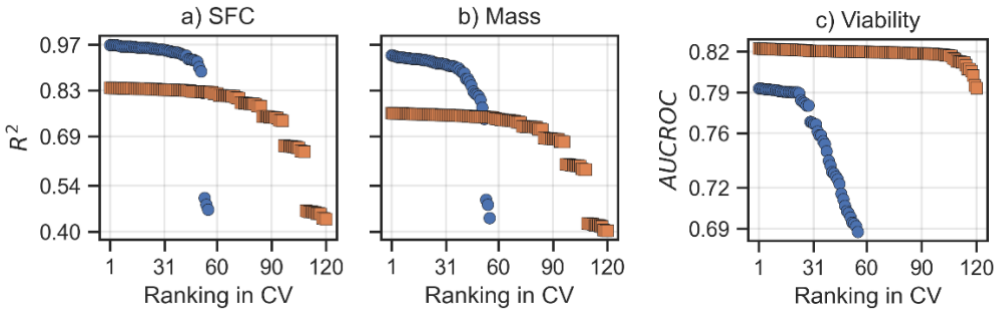

Figure 3. CV-results using all design parameters for MLP (●) and ET (■) surrogates: average scores for a) SFC, b) mass and c) viability.

In more detail, the overall best regression results are obtained by the MLPs with $J_2 = 16$ neurons and a moderate regularization strength of $\lambda = 4.0$ for both SFC and mass regression. The worst results

using MLPs are obtained with only one neuron and small regularization for both criteria. Overall, the ET models are less performant for regression. The sloping levels of the ETs are determined by the hyper-parameter K, where larger values for K lead to better results and vice versa. The best regression results using ETs are obtained with $K = 1.0$, $n = 4$ and $M = 800$ for both regression tasks.

For classification, the circumstances are, to a certain extent, kind of reversed. Here, ETs deliver the overall best results with a moderate $K = 0.4$, $M = 800$ and a less important value $n_{min} = 4$. The drop in ET's performance is mainly caused by too small values of both K and M. For MLP, it turned out that the most important hyper-parameter is the regularization strength λ, where the best results are achieved by $J_2 = 1$ and a regularization strength of $\lambda = 8$. However, the performance for more neurons remains at a similar level for $\lambda > 1$, whereas the performance decrease is mainly caused by too small regularization.

Based on these CV-results, the best performing models are selected with best settings of associated hyper-parameters and trained to the whole set of training data. They may now be applied to the test data, which are unknown new samples for the selected and trained models. Therefore, they serve as final estimate of the model quality. The results for test data are shown in Figure 4.

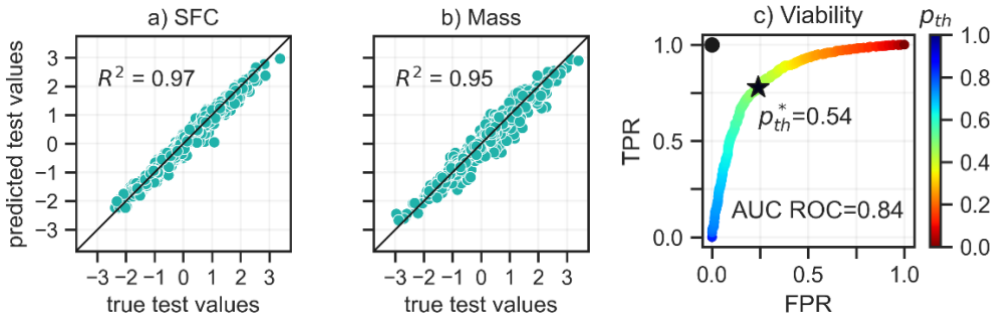

Figure 4. Results using all design parameters: test results for SFC (a) and mass (b) regression with MLP and c) ROC curve of ET classification model for viability.

In Figure 4.a and Figure 4.b the predicted values for SFC and mass are compared with the true test data. For an ideal prediction with $R^2 = 1$ (7), all points would lie exactly on the black line. In the real application here, a slight scatter about this ideal prediction can be seen, which is a bit smaller in the case of SFC regression with $R^2 = 0.97$ than in the case of mass regression with $R^2 = 0.95$. Nevertheless, both regression models can approximate the true data very well.

Figure 4.c shows the ROC curve for the classification of viability, where the color visualizes the cut-off threshold p_{th}. In the present case, the best classification model achieves an $AUC\ ROC = 0.84$ (see subsection 2.3). For the ideal value of $AUC\ ROC = 1$, the ROC curve would have to run through the ideal point $(FPR = 0, TPR = 1)$, which is marked by the black circle in Figure 4c. The black star marks the point on the ROC curve with the smallest Euclidean distance from this ideal point, achieved with $p_{th} = 0.54$. It should be noted here, that each point on the ROC curve is a Pareto-optimal solution of bi-criterion optimization minimizing FPR and maximizing the TPR. Thus, the choice of a particular cut-off threshold scalarizes the classification problem, which should be considered with care. If preference between FPR and TPR changes, another tradeoff may be more suitable.

4 Dimension reduction of design space using parameter ranking

As mentioned above, reducing the dimensionality of the aero engine design problem helps to simplify SAO. For this purpose, the design parameters need to be sorted w.r.t. to their individual influence on SFC, mass and viability. Then, based on this sorting, the best possible set of design parameters may be estimated using CV.

4.1 Global sensitivity analysis

The three outcomes of the original analysis model can be expressed as functions $y_{SFC}(\boldsymbol{x})$, $y_m(\boldsymbol{x})$ and $v(\boldsymbol{x})$ with $\boldsymbol{x} = [x_1, x_2, \dots, x_D]^T$. Quantitative global sensitivity analysis (SA) can be used to rank the design variables x_d w.r.t. their influence on these functions (see e.g. [23], [24]). Here, the model independent

variance-based SA-method called Sobol method [25] is applied. To measure the 'total' effect of a specific variable x_d, i.e., its first order (x_d alone) and higher order effects (interactions with the remaining variables), the total sensitivity index S_{Td} ([24],[26]) is defined as

$$S_{Td} = 1 - \frac{V_{\boldsymbol{x}_{\sim d}}\left(E_{x_d}[y|\boldsymbol{x}_{\sim d}]\right)}{V(y)} \tag{12}$$

where $\boldsymbol{x}_{\sim d}$ includes all variables except x_d and $V_{\boldsymbol{x}_{\sim d}}\left(E_{x_d}[y|\boldsymbol{x}_{\sim d}]\right)$ determines the expected reduction in variance caused by fixed values of $\boldsymbol{x}_{\sim d}$ while x_d is varied over all possible values. In practise, S_{Td} is estimated by quasi Monte Carlo (MC) sampling using $2N_{MC} \times (D+1)$ samples, where N_{MC} should be sufficiently large (at least 500) [26]. Here, $N_{MC} = 2^{12} = 4096$ is chosen, which leads to $2 \times 4096 \times 118 = 966.656$ samples in total. The Sobol method is carried out using the Python library *SALib* [27]. Since the original analysis function would be too expensive to evaluate the large sample, the best performing surrogates found in Section 3 are used instead, which is a matter of a few minutes even on low-cost computers.

The obtained S_{Td} results for each x_d are sorted with respect to their value, ranking the estimated influence of the design variables from 1 to 117. Figure 5 shows the 40 highest ranked values of S_{Td}. Note that the abscissa does not show the variables x_1 up to x_{40}, but only the ranking with descending importance, where the order of corresponding design parameters is different for SFC (●), mass (■) and viability (♦).

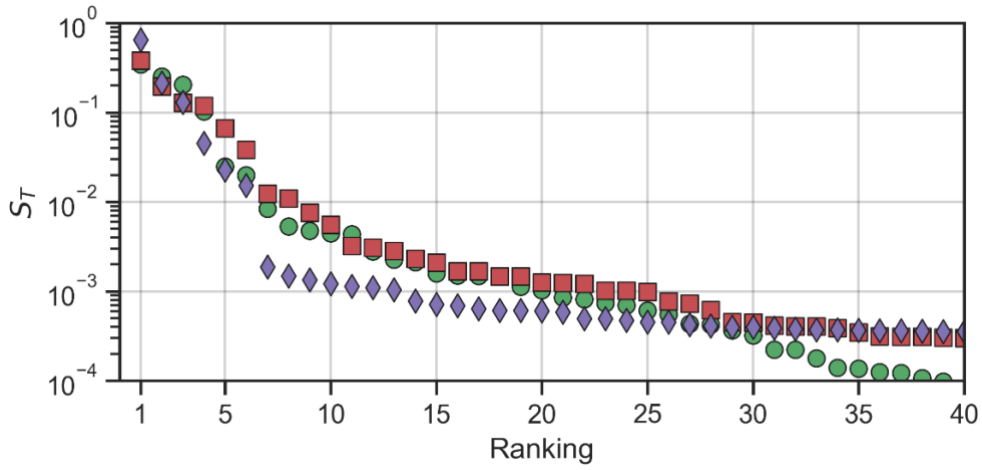

Figure 5. Total sensitivity index S_{Td} of the design variables for SFC (●), mass (■) and viability (♦) with descending variable influence associated with a higher ranking.

4.2 Finding reduced subsets of variables

From the original set of indices $\mathcal{I} = \{1,2,\dots,D\}$ representing all design parameters and the previously estimated ranking of the individual variables based on S_{Td}, three reduced sets of indices $\mathcal{I}_k^{red} \subseteq \mathcal{I}$, $k \in \{SFC, m, v\}$, of variables x_d, $d \in \mathcal{I}_k^{red}$, have now to be found for each of the optimization criteria and the viability constraint. For this purpose, a procedure based on the prediction quality is used, which has already been used in a similar form by Most and Will [28]. It finds the most optimal set of parameters by cross-validating the prediction performance of different sets of parameters.

Starting with a set containing only the most important parameter (being ranked as one in Figure 5), several MLPs with only this single input are cross-validated based on the training set used above and the various hyper-parameter combinations in Section 2.5. The resulting best average performance serves as reference. Then the two best ranked parameters are used as inputs for the MLPs and processed as before to provide a cross-validated performance measure. This is continued with three, four and more parameters up to 40, resulting in the performance measures in Figure 6 depending on the number of input parameters for the MLPs in the order of the ranking provided in Figure 5.

As can be seen, all three curves first increase with the number of parameters up to a certain and individual point. This number of parameters, for which the highest cross-validated performance is found, is marked in green. For the regression of SFC, the highest performance of $R^2 \approx 0.98$ is obtained using $\left|\mathcal{I}_{SFC}^{red}\right| = 32$ parameters. For mass, the best performance of $R^2 \approx 0.95$ is obtained with $|\mathcal{I}_m^{red}| = 28$ variables. For both regression problems, the performance stays almost constant beyond this point with

only minor performance changes. For classification, the situation is a little different. Here, the highest performance of $AUC\ ROC \approx 0.85$ is achieved with a number of $|\mathcal{J}_v^{red}| = 15$ highest ranked design parameters. Adding more parameters deteriorates the model performance significantly. Note that if all 117 variables would be added to the model, each performance would correspond to the results of the cross-validated model selection procedure in Section 3.

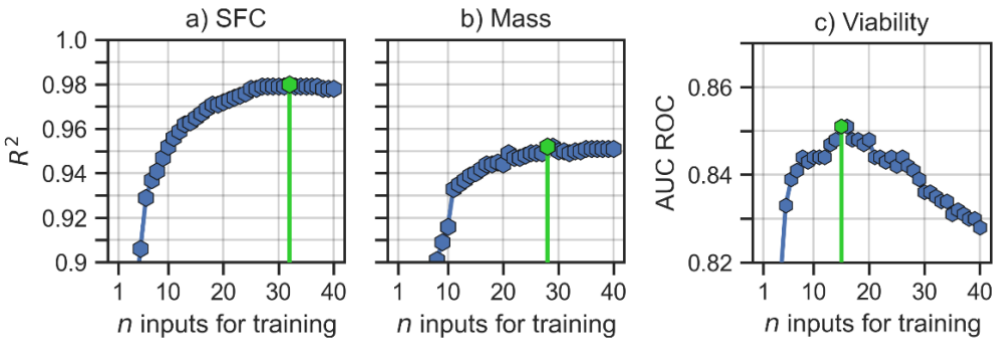

Figure 6. Mean cross-validation scores R^2 (regression) and $ROC\ AUC$ (classification) using different subsets of design variables in descending importance order.

4.3 Model selection using the reduced sets of design parameters

Analogous to the model selection procedure in Section 3, the three reduced sets of design parameters are now used to find the most appropriate model out of the MLPs and ETs described in subsections 2.5 and 2.6. Figure 7 shows the results in analogous form to Figure 3. As can be seen, the curves behave rather similar, where surprisingly models with the reduced dimension generally show a (slightly) better performance. This is particularly evident in Figure 7c for classification, where certain MLP topologies now achieve even better performance than the ET models. The MLP approach is now able to detect relations with rather small to moderate regularizations of $\lambda = 0.5$, but with significantly more neurons of $J_2 = 16$ in contrast to the results from Section 3. The worst performing MLP models are mainly characterized by very small regularizations. The ET approach behaves similar to Section 3, where best results are obtained by the models using $M = 800$ trees and $K = 0.3$. The parameter n_{min} plays a minor role, but should be around $n_{min} = 8$.

The highest regression performance is dominated by the MLPs, just as before in Section 3. Different to Section 3 is now, that the respective topologies of the best MLPs differ between SFC and mass regression. For SFC, the best results are generally obtained with a high number of neurons $J_2 = 16$ and a moderate regularization strength $\lambda = 0.5$, whereas for mass regression a rather small number of neurons $J_2 = 4$ and moderate to small regularizations $\lambda < 1$ performs best. Overall, the ETs perform worse than the MLPs, where the best ET results are obtained, as before in Section 3, for both criteria for high values of K and M as well as small values for n_{min}.

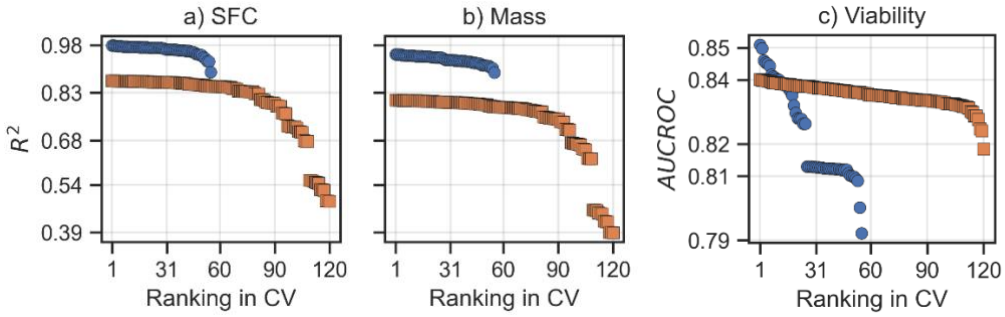

Figure 7. CV-results using reduced sets of parameters for surrogate models MLP (●) and ET (■): average scores for a) SFC, b) mass and c) viability.

4.4 Testing the best found surrogates using reduced sets of parameters

Similar to Section 3, the best models are again applied to the test data where now, based on the results from the model selection procedure, MLPs are used for all three criteria. The predictions of the test data regarding SFC, mass and viability, as well as the corresponding metrics are shown in Figure 8.

Compared to Figure 4, no severe degradation in the prediction of the unknown test data occurred due to the reduced dimensionality. Only the prediction of the mass now results in larger deviations for some individual samples. The R^2 remains roughly the same due to the overall slight decrease in variance about the ideal black line.

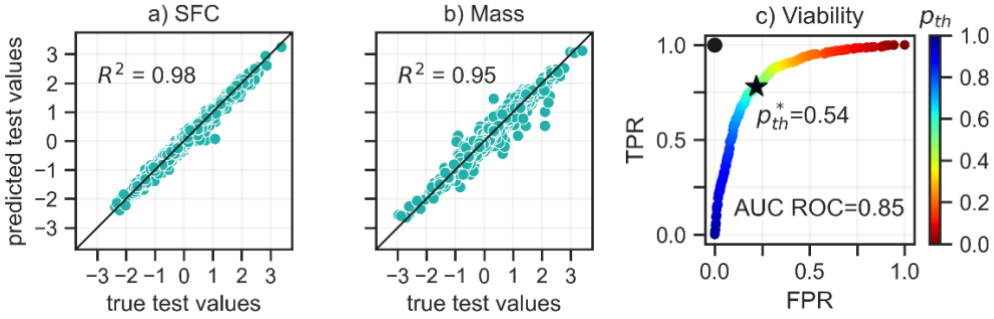

Figure 8. Test-results using reduced subsets of design parameters and MLPs for both regression and classification.

5 Design optimization including the assessment of viability

Based on the models from subsection 4.4, now a multi-objective aero engine optimization with reduced subsets of design parameters can be carried out, where the viability of a design can be incorporated as, in contrast to a usual SAO, an additional design constraint. Here, the viability is intentionally the only constraint. For the purpose of optimization, the reduced sets of parameters are combined to a union set of design parameters. Figure 9 illustrates the three individual sets of parameter indices $\mathcal{I}_k^{red}$ corresponding to the design parameters used by the three different surrogates for $k \in \{SFC, m, v\}$. The design vector $\boldsymbol{x}_{red}$ of parameters to be considered for optimization has to combine these parameters x_d, where $d \in \mathcal{I}^{red} = \mathcal{I}_{SFC}^{red} \cup \mathcal{I}_m^{red} \cup \mathcal{I}_v^{red}$. Since the three different parameter sets have some intersections, the cardinality of the union set is $|\mathcal{I}^{red}| = 44$.

The bi-criterion optimization using criteria surrogates $\hat{\boldsymbol{y}} = [\hat{y}_{SFC}, \hat{y}_m]^{\mathrm{T}}$ and viability constraint $p_v \geq p_{th}$ then reads as

$$\min_{\boldsymbol{x}_{red} \in \mathcal{X}_{red}} \hat{\boldsymbol{y}}(\boldsymbol{x}_{red}) \; where \, \mathcal{X}_{red} \coloneqq \{\boldsymbol{x}_{red} \in \mathbb{R}^{44} | p_v(\boldsymbol{x}_{red}) \geq p_{th}\} \tag{13}$$

The optimization task is conducted by the genetic algorithm NSGA-II [29] implemented in the Python library pymoo [30] using a population size of 100 and 200 generations.

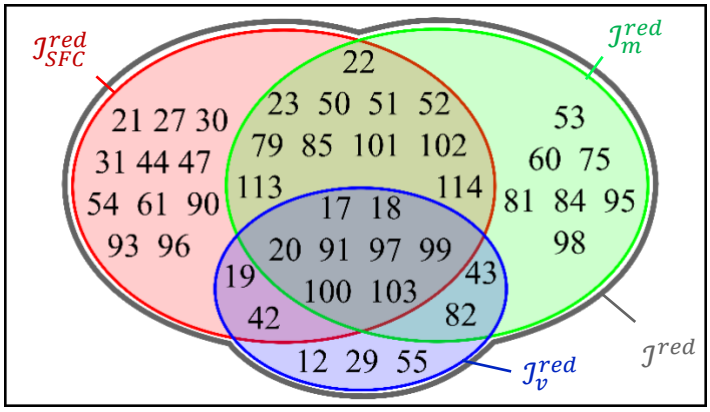

Figure 9. Individual index-subsets $\mathcal{I}_k^{red}$, $k \in \{SFC, mass, v\}$, and union $\mathcal{I}^{red}$ of design parameters for optimization

Design evaluations supposed by the optimizer are carried out with the three best found surrogates in subsection 4.4 where variation of the reduced set of design parameters $\boldsymbol{x}_{red} \in \mathcal{X}_{red}$ is sufficient. In total, three distinct optimization runs are conducted with different values $p_{th} \in \{0, 0.54, 0.8\}$ for the cut-off threshold. Since the estimated probability $p_v(\boldsymbol{x}_{red})$ of $\boldsymbol{x}_{red}$ to be viable is non-negative $(p_v(\boldsymbol{x}_{red}) \geq 0 \; \forall \, \boldsymbol{x}_{red})$, the value $p_{th} = 0$ represents an optimization without taking into account the viability

constraint. The value $p_{th} = 0.54$ represents the point on the ROC with the smallest Euclidean distance to the ideal point in Figure 8.c and $p_{th} = 0.8$ is set to an arbitrary more conservative value to reduce the FPR. At this point, it should be noted again that there is no best point on the ROC, but all are optimal trade-offs. Depending on the use case, setting any other value greater than zero may also be justified and appropriate.

Each of the three optimizations yields a set of Pareto-optimal solutions, as shown in Figure 10, where Figure 10.a shows the Pareto-optimal solutions as well as the outcomes of the training DoE and Figure 10.b shows a magnified section of the Pareto-optimal solutions. The color of each solution represents the outcomes of the classification p_v and thus the estimated probability that the design belongs to the set of viable designs. While the optimization without incorporating viability $(p_{th} = 0)$ yields the best results with respect to SFC and mass, the classification of all these designs would consistently suggest them as non-viable ($p_v \to 0$), hence unrealistic designs. When viability is taken into account (by a constraint using $p_{th} > 0$), the set of Pareto-optimal solutions shifts slightly toward worse designs. Here, more strict viability constraints mainly affect the upper left part of the Pareto-front for small SFC-values and relatively large masses. However, the difference is rather small and all of the designs now have a significantly higher estimated probability of being viable. Even the results for the strictest constraint are only slightly worse w.r.t. SFC and mass.

The relatively small differences w.r.t. the optimization criteria may, despite the considerable differences w.r.t. the constraint, caused by the fact that separate sets of design parameters were assigned to each of the criteria and the constraint (see Figure 9). Although these sets have intersections, they also contain some design parameters that are exclusive to the respective criteria and constraint. Thus, the individual criteria as well as the constraint can, at least to a certain extent, be fine-tuned separately of each other.

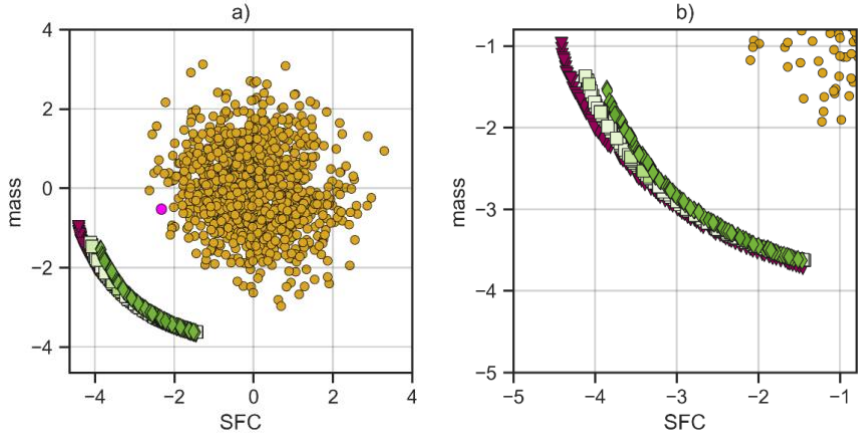

Figure 10. Resulting Pareto-optimal solutions for different cut-off thresholds p_{th} (▼: 0, ■: 0.54, ♦: 0.8) relative to the training DoE outcomes (●) in the a) overall view and b) magnified section.

6 Conclusions

In the present work, surrogate-assisted optimization of a high-dimensional aero engine design problem incorporating the additional information about the viability of designs into the optimization via constraint is investigated on the basis of an industrial application example, i.e., preliminary aero engine design. The following conclusions can be drawn from this research: (i) the type and hyper-parameters of the most appropriate surrogate model depends on the problem as well as the dimensionality and should be chosen carefully; (ii) a cross-validated search along the ranking of parameters based on the results of variance-based sensitivity analysis can reduce the dimension significantly without decreasing the prediction performance; (iii) the incorporation of viability into a surrogate-assisted optimization is reasonable or even necessary to avoid pseudo-optimal non-viable results; (iv) identifying individual subsets of design parameters for the individual optimization criteria and constraint may help to find good results while satisfying the constraint.

The conducted investigation shows the workability of the presented framework and emphasizes that viability, whenever it might be an issue in the context of surrogate-assisted optimization, should be considered in the process.

Nevertheless, this investigation has its limitations too and can only be the start for answering further questions. These include, among others, the inclusion of usual optimization constraints that narrow the feasible design space, a more imbalanced viability in the data as well as a smaller initial sample size and the incorporation of adaptive re-sampling strategies for this high-dimensional application.

7 Acknowledgements

The investigations within this paper were carried out in collaboration with Rolls-Royce Deutschland Ltd & Co KG as part of the joint research program VIT-V (Verfahren der Industire 4.0 für die Triebwerks-Vorentwicklung) funded by the ERDF (European Regional Development Fund) and the federal state State of Brandenburg. The second author thanks the German Research Foundation (DFG, grant no. 50184085) for supporting his AI-studies. Greatly acknowledged is the permission by Rolls-Royce Deutschland Ltd & Co KG to publish this work.

8 References

[1] T. Simpson, V. Toropov, V. Balabanov, and F. Viana, “Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come-or not,” in 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008, p. 5802.

[2] K. Cheng, Z. Lu, C. Ling, and S. Zhou, “Surrogate-assisted global sensitivity analysis: an overview,” Struct Multidisc Optim, vol. 61, pp. 1187–1213, 2020.

[3] C. Audet, J. Denni, D. Moore, A. Booker, and P. Frank, “A surrogate-model-based method for constrained optimization,” in 8th Symposium on Multidisciplinary Analysis and Optimization, 2000, p. 4891.

[4] J. Li, P. Wang, H. Dong, J. Shen, and C. Chen, “A classification surrogate-assisted multi-objective evolutionary algorithm for expensive optimization,” Knowledge-Based Systems, vol. 242, p. 108416, 2022.

[5] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” Journal of Global Optimization, vol. 13, no. 4, p. 455, 1998.

[6] T. Most and J. Will, “Metamodel of Optimal Prognosis - an automatic approach for variable reduction and optimal metamodel selection,” Proc. Weimarer Optimierungs-und Stochastiktage, vol. 5, pp. 20–21, 2008.

[7] M. Lockan, P. Amtsfeld, D. Bestle, and M. Meyer, “Non-Linear Sensitivity Analysis Based on Regression Trees with Application to 3D Aerodynamic Optimization of Gas Turbine Blades,” in International Conference on Engineering and Applied Sciences Optimization (OPT-i), Kos, Griechenland, 2014.

[8] L. Hartwig and D. Bestle, “Compressor blade design for stationary gas turbines using dimension reduced surrogate modeling,” in 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1595–1602.

[9] Z. Wang and M. Ierapetritou, “A novel feasibility analysis method for black-box processes using a radial basis function adaptive sampling approach,” AIChE J., vol. 63, no. 2, pp. 532–550, 2017.

[10] A. I. J. Forrester, A. Sóbester, and A. J. Keane, “Optimization with missing data,” Proc. R. Soc. A., vol. 462, no. 2067, pp. 935–945, 2006.

[11] Y. He, J. Sun, P. Song, and X. Wang, “Dual Kriging assisted efficient global optimization of expensive problems with evaluation failures,” Aerospace Science and Technology, vol. 105, p. 106006, 2020.

[12] Y. He, J. Sun, P. Song, and X. Wang, “Variable-fidelity expected improvement based efficient global optimization of expensive problems in presence of simulation failures and its parallelization,” Aerospace Science and Technology, vol. 111, p. 106572, 2021.

[13] M. Sacher et al., “A classification approach to efficient global optimization in presence of non-computable domains,” Struct. Multidisc Optim., vol. 58, pp. 1537–1557, 2018.

[14] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” The Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[15] M. D. McKay, R. J. Beckman, and W. J. Conover, “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” Technometrics, vol. 42, no. 1, pp. 55–61, 2000.

[16] G. C. Cawley and N. L. C. Talbot, “On over-fitting in model selection and subsequent selection bias in performance evaluation,” The Journal of Machine Learning Research, vol. 11, pp. 2079–2107, 2010.

[17] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.

[18] M. W. Browne, “Cross-Validation methods,” Journal of Mathematical Psychology, vol. 44, no. 1, pp. 108–132, 2000.

[19] Daniel Berrar, “Cross-Validation,” in Reference Module in Life Sciences, B. D. Roitberg, Ed., Place of publication not identified: Elsevier, 2016.

[20] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, “Prediction error estimation: a comparison of resampling methods,” Bioinformatics (Oxford, England), vol. 21, no. 15, pp. 3301–3307, 2005.

[21] R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, and M. Steinbrecher, Computational Intelligence. Cham: Springer International Publishing, 2022.

[22] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” Mach Learn, vol. 63, pp. 3–42, 2006.

[23] I. M. Sobol, “Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates,” Mathematics and Computers in Simulation, vol. 55, 1-3, pp. 271–280, 2001.

[24] T. Homma and A. Saltelli, “Importance measures in global sensitivity analysis of nonlinear models,” Reliability Engineering & System Safety, vol. 52, no. 1, pp. 1–17, 1996.

[25] I. M. Sobol', “On sensitivity estimation for nonlinear mathematical models,” Matematicheskoe modelirovanie, vol. 2, no. 1, pp. 112–118, 1990.

[26] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola, “Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index,” Computer Physics Communications, vol. 181, no. 2, pp. 259–270, 2010.

[27] J. Herman and W. Usher, “SALib: An open-source Python library for sensitivity analysis,” Journal of Open Source Software, vol. 2, no. 9, p. 97, 2017.

[28] T. Most and J. Will, “Sensitivity analysis using the Metamodel of Optimal Prognosis,” Weimar Optimization and Stochastic Days, vol. 8, no. 0, 2011.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Trans. Evol. Computat., vol. 6, no. 2, pp. 182–197, 2002.

[30] J. Blank and K. Deb, “Pymoo: Multi-Objective Optimization in Python,” IEEE Access, vol. 8, pp. 89497–89509, 2020.