



Universiteit
Leiden
The Netherlands

Global sensitivity analysis is not always beneficial for evolutionary computation: a study in engineering design

Raponi, E.; Olarte Rodriguez, I.; Stein, N. van; Kononova, A.V.

Citation

Raponi, E., Olarte Rodriguez, I., & Stein, N. van. (2025). Global sensitivity analysis is not always beneficial for evolutionary computation: a study in engineering design. In A. V. Kononova (Ed.), *Natural Computing Series* (pp. 13-40). Singapore: Springer.
doi:10.1007/978-981-96-2540-6_2

Version: Publisher's Version

License: [Licensed under Article 25fa Copyright Act/Law \(Amendment Taverne\)](#)

Downloaded from: <https://hdl.handle.net/1887/4294891>

Note: To cite this publication please use the final published version (if applicable).

Global Sensitivity Analysis Is Not Always Beneficial for Evolutionary Computation: A Study in Engineering Design



Elena Raponi, Ivan Olarte Rodriguez, and Niki van Stein

Abstract In this chapter, we critically examine the application of global sensitivity analysis (GSA) techniques for variable screening in evolutionary optimization, particularly focusing on their effectiveness in reducing dimensionality for problems optimized via CMA-ES. We assess the use of the GSAreport tool, which generates sensitivity reports aimed at identifying what parameters are influential on a target of a given problem and their interactions. We analyze two test cases from engineering design: (1) a star-shaped crash box subjected to axial impact, evaluating crashworthiness, and (2) a sheet metal forming application, assessing the drawability of metal components. While GSA methods can identify variables with significant impact on model outputs, we find that relying solely on GSA for dimensionality reduction may limit optimization effectiveness when using adaptive algorithms like CMA-ES. In these case studies, we demonstrate that while GSA aids in understanding variable importance, the exclusion of “less influential” variables may oversimplify the problem space, potentially hindering CMA-ES’s capacity to explore and adapt to complex landscapes.

Keywords Evolutionary computation · Explainable AI · Variable selection · Global sensitivity analysis · Dimensionality reduction · Engineering design

E. Raponi (✉) · I. O. Rodriguez · N. van Stein
LIACS, Leiden University, Leiden, The Netherlands
e-mail: e.raponi@liacs.leidenuniv.nl

I. O. Rodriguez
e-mail: i.olarte.rodriguez@liacs.leidenuniv.nl

N. van Stein
e-mail: n.van.stein@liacs.leidenuniv.nl

1 Introduction

In the wider field of Explainable Artificial Intelligence (XAI), Global Sensitivity Analysis (GSA) [1, 2] has emerged as a fundamental tool for the analysis of complex systems in fields ranging from machine learning (ML) and, more broadly, artificial intelligence (AI), to engineering design [3, 4]. XAI seeks to provide transparency in ML and AI models, addressing one of the major challenges in AI deployment, i.e., trust in the prediction. As ML models grow in complexity, domain experts and decision-makers are increasingly concerned not just with the accuracy of a model's predictions but also with understanding the reasoning behind how the model arrives at those predictions (e.g., model intermediate behavior, decision rules, causal and counterfactual explanations, feature importance, global vs. local model prediction). Sensitivity analysis (SA) methods, particularly global approaches [5], aim to explain how input features influence a model's output by attributing the uncertainty in the system's output to variations in different input sources. As such, GSA is considered a model-agnostic approach to XAI, making it applicable across various fields where interpretability is critical.

When applied to optimization, GSA is able to reveal how input variables interact and contribute to the overall variability of the output. This provides crucial insights into where efforts should be focused to effectively define an optimization problem—such as selecting relevant variables, reducing dimensionality, or properly setting constraints and objectives—thereby accelerating algorithm convergence toward optimal solutions, increasing solution robustness, and enhancing overall algorithm performance.

In engineering design, GSA plays a dual role: first, as a tool for understanding model behavior, and second, as a method for optimizing designs through variable screening [6–8]. Variable screening [5] is the process of identifying the most influential parameters in a design problem and focusing computational resources on them, thereby reducing dimensionality and computational cost. In many engineering problems, this is particularly important because insights about the target that is optimized and the constraints that must be fulfilled are extracted from computationally expensive simulations, such as finite element (FE) analyses. To evaluate each parameter configuration of a mechanical design, one simulation must be performed, which takes hours or even days to complete. However, running such simulations across high-dimensional parameter spaces, where many parameter combinations must be evaluated, can be prohibitively expensive. This is where GSA provides significant value.

In the broader family of gradient-free optimization algorithms, also known as black-box algorithms, evolutionary algorithms (EAs) [9] have been widely used in engineering design over the past three decades [10]. This popularity stems from several key advantages: EAs do not require any specific conditions on the objective or fitness landscape characteristics (such as continuity, differentiability, or convexity), they can efficiently handle discrete and mixed variables, and they are capable of addressing multiple objectives simultaneously. Additionally, EAs allow for the

parallelization of evaluations, which is particularly beneficial in simulation-based optimization, where simulations can be distributed across multiple computing nodes to reduce time-to-solution. These advantages, coupled with the continuous advancement in computational power, have significantly expanded their use in both research and industry.

The focus of this chapter is on evaluating the role of GSA in variable screening for evolutionary optimization, with an emphasis on mechanical design applications. While GSA techniques are often employed to identify influential variables, we adopt a critical perspective on their utility when used for dimensionality reduction for optimization, especially when algorithm from evolutionary computation (EC) with efficient adaptation rules, like CMA-ES [11, 12], are used. Using the open-source tool GSAreport [2, 13], which automates the generation of sensitivity reports, we assess whether GSA can effectively streamline optimization by focusing on key variables without constraining the optimizer's flexibility. This critical evaluation is illustrated through two practical mechanical design case studies: a star-shaped crash box under axial impact and a sheet metal forming process. Through these examples, we question whether GSA-driven variable screening truly enhances optimization in complex design landscapes or if it risks oversimplifying problems in ways that may hinder adaptive search algorithms.

2 Background

The study presented in this chapter builds upon the foundational concepts of GSA and EC, exploring their interplay in the context of engineering optimization. GSA helps identify the most influential variables in complex models, while EC provides powerful tools for solving black-box optimization problems. To establish the necessary background, we introduce these key concepts and provide the reader with a brief overview of their theoretical foundations and practical relevance.

2.1 *Global Sensitivity Analysis and GSAreport*

SA methods can be divided into global and local approaches. GSA assesses the influence of input variations across their entire range to determine each variable's overall contribution to output variability, while local SA (LSA) examines the impact of small, localized changes in inputs around specific points. GSA considers variable interactions and is independent of any nominal point, whereas LSA is typically used to explain individual predictions. Additionally, SA methods can be classified into derivative-based, variance-based, and density-based approaches. In this section, we will focus on GSA, as it provides a better global understanding of the importance of different parameters, allowing for improved problem formulation through variable screening.

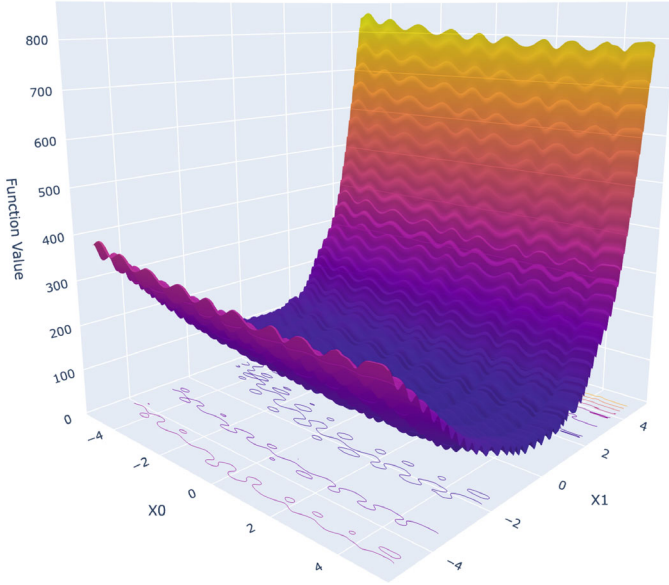


Fig. 1 Rastrigin function (f_3) of the BBOB benchmark with $n = 2$. The last parameter (in this case X_1) is the most influential

Several attempts have been made in the literature to provide an overview of available GSA methods [1, 2, 14, 15]. Among them, van Stein et al. [2] provide a comprehensive comparison of popular GSA methods, as well as pseudo-GSA methods (i.e., methods that can be applied for GSA even if it is not their primary purpose). They evaluate these methods both qualitatively and quantitatively, testing their robustness across different sample sizes and their accuracy on a large set of randomly generated linear functions with varying dimensions. Besides the review, the authors propose an open-source library GSAREport¹ [13] that allows users to create GSA reports for any model or dataset. As we will run GSAREport on selected test cases from computational mechanics, we here survey the GSA methods that are included in the report generated by this tool.

For demonstration purposes, to explain the different methods of SA, we use a 20-dimensional Rastrigin function (f_3) from the Black-Box Optimization Benchmarking (BBOB) test suite within the COCO framework [16] as the function we want to analyze, which is defined as follows:

$$f_3(x) = 10n + \sum_{i=1}^n [z_i^2 - 10 \cos(2\pi z_i)] + f_{\text{opt}}, \quad (1)$$

Figure 1 shows a surface plot of the function with $n = 2$ input parameters.

¹ <https://github.com/nikivanstein/GSAREport>.

Variance-Based Methods Variance-based methods operate under the assumption that output uncertainty can be effectively described by variance alone, as discussed by Saltelli et al. [17]. These methods determine the sensitivity of input parameters through an ANOVA-like decomposition of the function, partitioning the total variance into contributions from individual inputs and their interactions. This approach provides a straightforward means of quantifying the influence of each parameter and their combinations on the output variability.

Here, we focus on two widely used variance-based techniques: Sobol' indices and the Random Balance Designs Fourier Amplitude Sensitivity Test (RBD-FAST). Sobol' indices are well-regarded for their ability to capture both main and interaction effects comprehensively, making them suitable for high-dimensional problems where input interactions play a significant role. However, Sobol' methods are often computationally demanding. To balance accuracy and efficiency, we also consider RBD-FAST, a method that approximates sensitivity indices with fewer function evaluations by leveraging a Fourier decomposition approach.

Sobol'. The popular Sobol' method [18], also known as Sobol' indices, decomposes output variance into portions attributable to individual inputs or groups of inputs. This method allows for calculating first-order sensitivities (variance contribution of a single input) as well as higher-order sensitivities, often focusing on first, second, and total order effects. Sobol' indices are computed via Eq. (2), where $\text{Var}(Y)$ represents output variance, decomposed with respect to input sets X_i . This set-based approach enables Sobol' to assess sensitivity across individual or grouped inputs.

$$\text{Var}(Y) = \sum_{i=1}^n V_i + \sum_{i < j}^n V_{ij} + \dots + V_{12\dots n}, \quad (2)$$

where

$$V_i = \text{Var}_{X_i}(E_{X_{\sim i}}(Y|X_i)), \quad (3)$$

$$V_{ij} = \text{Var}_{X_{ij}}(E_{X_{\sim ij}}(Y|X_i, X_j)) - V_i - V_j, \quad (4)$$

and $X_{\sim i}$ denotes the set of all input variables except X_i .

The Sobol' indices plot output by *GSAreport* on the 20-dimensional Rastrigin function is shown in Fig. 2. Sobol' can calculate the first, second, and total order sensitivity of the parameters.

RBD-FAST. RBD-FAST [19] is an efficient variant of the classical Fourier Amplitude Sensitivity Test (FAST) [20], designed to reduce computational demands while maintaining comparable accuracy. FAST and RBD-FAST are based on conditional variances, quantifying the main and total effect contributions of each input factor to

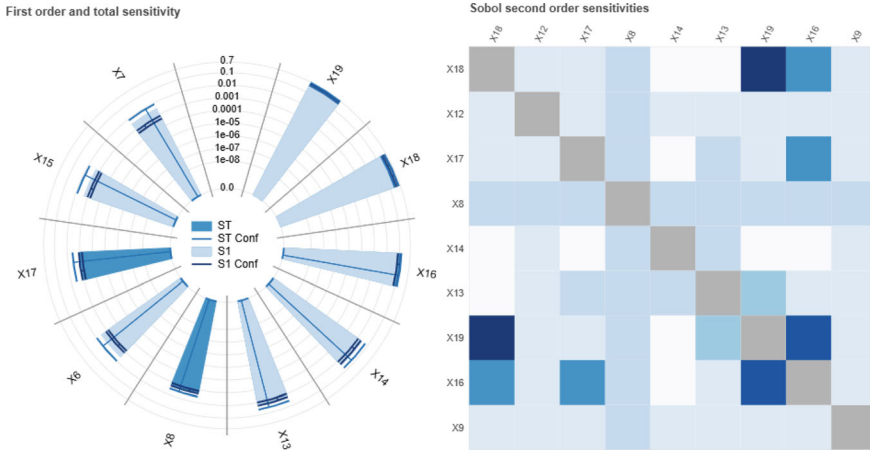


Fig. 2 Left plot shows Sobol’ sensitivity indices of f_3 as stacked bars for first-order (S1) and total sensitivity (ST) with 95% confidence intervals as dark blue error bars for both S1 and ST. The right plot is a heatmap illustrating second-order sensitivities between parameter pairs, where dark blue indicates high sensitivity and white indicates zero sensitivity [2]

output variance. FAST leverages Fourier series decomposition to isolate the influence of each input variable on the output variance. This is achieved by assigning each variable a unique frequency, hence generating oscillatory patterns that make it possible to separate and quantify the main and interaction effects of each variable based on its distinct frequency response.

While FAST applies a periodic sampling scheme, where the number of harmonics to sum in the Fourier series decomposition must be defined by the user, RDB-FAST relaxes this assumption and does not require a specific sampling scheme, making it a flexible choice for complex models.

An example of the first-order (S1) sensitivity calculated by using RBD-FAST for f_3 is given in Fig. 3.

Derivative-Based Methods. Derivative-based sensitivity analysis methods [21] assess the impact of small variations in input parameters on the output by evaluating the partial derivatives of the model with respect to each input. These methods are particularly useful for models where outputs are smooth and differentiable functions of the inputs, as they can capture the rate of change in the output resulting from changes in each variable. Unlike global methods, derivative-based approaches focus mainly on local sensitivity, often in the vicinity of a nominal input value. A famous exception to this is given by the Morris method, as it samples the nominal value within the entire input space. While precise, derivative-based GSA methods can become computationally intensive for models with many inputs or those requiring high-resolution gradients.

Morris method. Among derivative-based methods, the Morris method [22] stands out as a qualitative technique that offers several practical advantages. Designed to

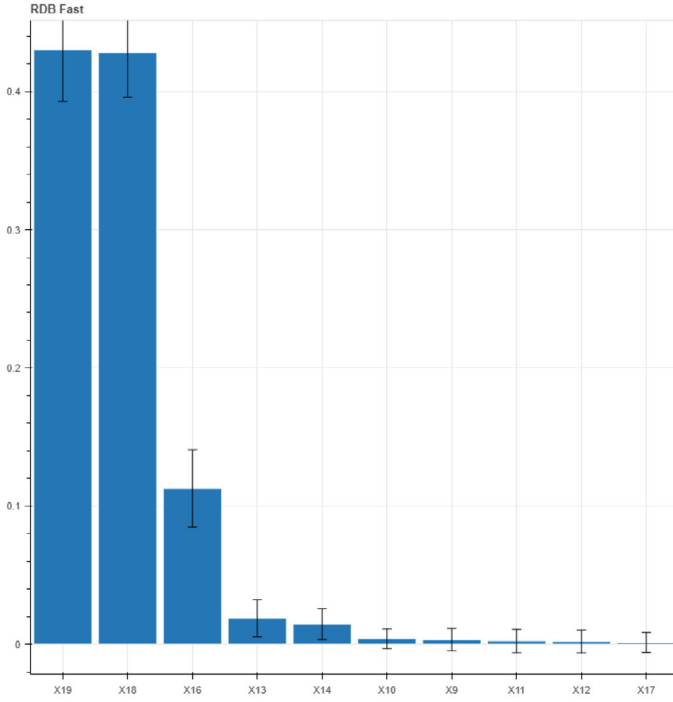


Fig. 3 RDB-FAST first-order sensitivity indices of f_3 with confidence as error bars

estimate the influence of factors in a computationally efficient way, the Morris method can operate directly on discrete domains, which makes it adaptable to a wide range of models. It utilizes multi-dimensional averaging, capturing the effect of a factor while other variables vary, providing a more realistic measure of sensitivity in complex systems. Additionally, it allows for factor grouping, making it effective for handling large datasets or systems with many inputs. A modified version of the Morris method, which calculates the absolute means of the elementary effects distribution, is also robust against type II errors, enhancing reliability in detecting significant factors.

Given a random sampling of an input vector $\mathbf{X} = (X_1, X_2, \dots, X_n)$ in an n -dimensional p -level grid, the Morris method calculates a distribution of elementary effects F_i for each input X_i . By generating r randomly sampled trajectories that adjust one input at a time by $\pm\Delta$, the elementary effect of the i th input on the j th trajectory is defined as

$$EE_i(\mathbf{X}^j) = \frac{[Y(X_1^j, \dots, X_i^j \pm \Delta, \dots, X_n^j) - Y(X_1^j, \dots, X_i^j, \dots, X_n^j)]}{\Delta}.$$

Based on this definition, two sensitivity indices are computed for each factor: the mean of the absolute value of the elementary effects

$$\mu_i^* = \frac{1}{r} \sum_{j=1}^r |\mathbb{E}E_i(\mathbf{X}^j)|, \quad (5)$$

and the standard deviation of the elementary effects

$$\sigma_i = \sqrt{\frac{1}{r} \sum_{j=1}^r \left(\mathbb{E}E_i(\mathbf{X}^j) - \frac{1}{r} \sum_{j=1}^r (\mathbb{E}E_i(\mathbf{X}^j)) \right)^2}. \quad (6)$$

The mean of the absolute elementary effects μ_i^* indicates the input's influence on output variance, while the standard deviation σ_i represents potential nonlinearity or interaction effects. Results are visualized on a σ vs. μ^* plot (Fig. 4). Notably, μ^* , introduced by Campolongo et al. [23], uses absolute values to prevent cancelation effects in non-monotonic models.

Density-Based Methods Density-based SA methods assess input sensitivity by examining the full probability density function (PDF) of the model output, allowing them to capture input interactions effectively. These methods are advantageous over variance-based approaches when input dependencies are present. However, their feasibility decreases with high-dimensional input spaces or when model evaluation times are lengthy. Among the other density-based SA methods, DELTA and PAWN are widely known and used.

DELTA. The DELTA (δ) method [24] is a density-based sensitivity analysis approach that operates independently of the sampling method used. DELTA calculates both first-order sensitivities and a total sensitivity measure, δ , for each input. Unlike variance-based methods, DELTA evaluates the influence of the entire input distribution on the output distribution, focusing on the shape rather than specific moments (e.g., mean or variance) of the output. The moment-independent sensitivity indicator,

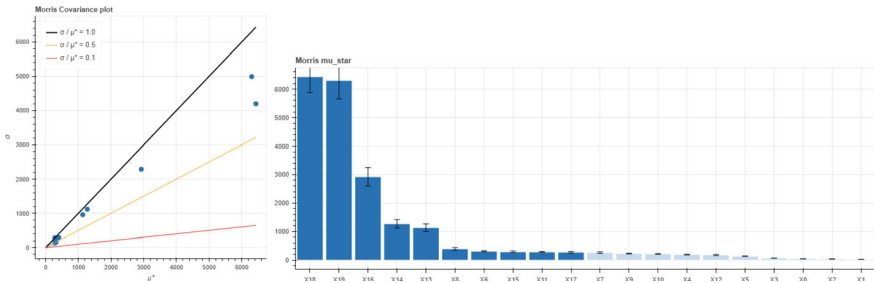


Fig. 4 Morris σ versus μ^* plot for SA on 20 factors for f_3 . This plot categorizes factors into three types: non-influential (low μ^* and σ), linear/additive (high μ^* , low σ), and nonlinear/interaction-driven (low μ^* , high σ)

δ , for each factor X_i , quantifies how changes in the distribution of X_i impact the overall output distribution, making it especially useful in capturing nonlinear and interaction effects across the entire output range. The moment-independent sensitivity indicator δ for a factor X_i is defined as

$$\delta_i = \frac{1}{2} \int f_{X_i}(x) \left[\int |f_Y(y) - f_{Y|X_i=x}(y)| dy \right] dx,$$

where X_i is a fixed input variable, $f_{X_i}(x)$ is its marginal probability density function, $f_Y(y)$ is the cumulative probability density function of the model output Y , and $f_{Y|X_i=x}(y)$ is the conditional density of Y given that one of the parameters, X_i , assumes a fixed value. Figure 5 shows the *GSAreport* output for the DELTA method on the 20-dimensional Rastrigin function.

PAWN. PAWN [25] is an efficient density-based GSA method that derives sensitivity metrics using cumulative distribution functions (CDFs), which are simpler to compute than probability density functions (PDFs). A key advantage of PAWN is its ability to calculate sensitivity indices over the entire output range or within specific

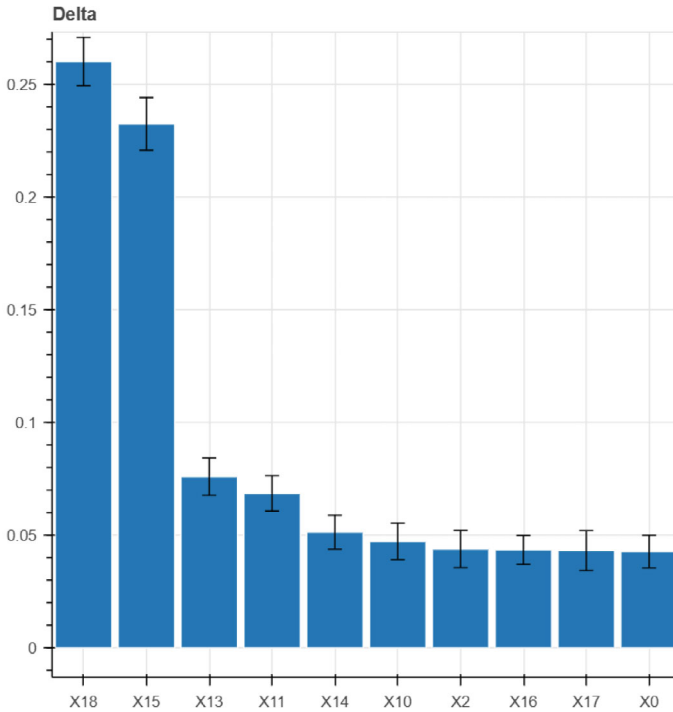


Fig. 5 δ sensitivities plot for f_3 , indicating X_{18} as most influential

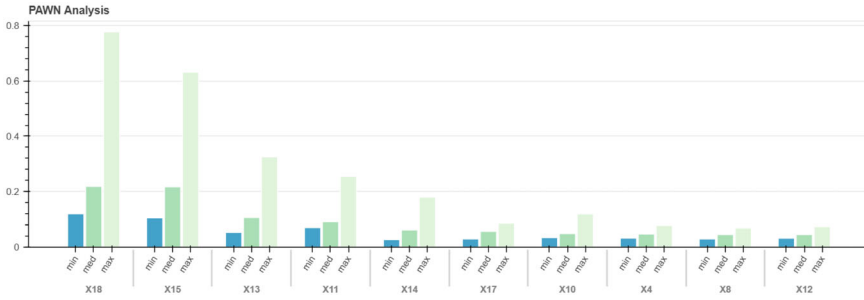


Fig. 6 PAWN minimum, median, and maximum sensitivity index on f_3

sub-ranges, making it useful for targeted applications. PAWN provides a comprehensive sensitivity profile for each parameter, including mean, minimum, median, maximum, and standard deviation. Figure 6 displays the minimum, median, and maximum sensitivity indices for the most influential parameters of the f_3 function.

Model-Based Methods Machine learning models can also provide insights into input feature importance. However, these insights are a by-product rather than the primary purpose of these models. Each model type has assumptions about the data—for instance, using linear regression on highly nonlinear functions is inappropriate for estimating feature sensitivity. Among model-based method, here we present random forests and Shapley values.

Random Forest. Random forest [26] is a machine learning technique that builds multiple decision trees on different random subsets of input features. By averaging the predictions from these diverse trees, the model improves robustness and accuracy, making it both powerful and flexible. As a by-product, random forests can provide feature importance scores, illustrated in Fig. 7 for the f_3 function. This is calculated by tracking how much each feature contributes to reducing the “impurity” (often using the Gini impurity) across all trees in the forest. Specifically, it measures the total reduction in the criterion (like Gini impurity or entropy) each feature achieves and normalizes this across trees, creating an importance score for each feature. This importance score, also called “Gini importance,” gives a sense of which features the model relies on most for its predictions. While random forests are powerful and flexible, they also have shown some disadvantages. First, their feature importance scores can sometimes be unreliable for high-cardinality features. Second, the model lacks the transparency of simpler models like single decision trees. Last, the prediction performance can vary depending on the values set for specific hyperparameters.

Shapley values. Shapley values, a cooperative game-theoretic approach, explain model outputs by treating input parameters as “players” contributing to the overall prediction. A core property of Shapley values is that their sum equals the difference between the model’s prediction with all features present and the prediction with none.

While typically used for local explanations (e.g., SHAP [27]), Shapley values can also provide global sensitivity by averaging across all data instances. However,

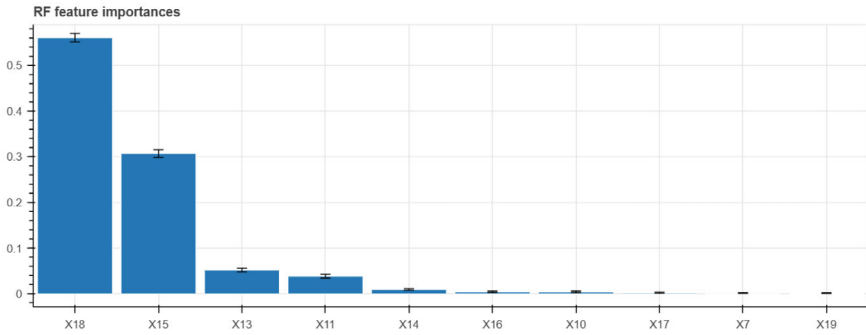


Fig. 7 Random forest feature importance on f_3

exact computation is often infeasible due to its exponential complexity, and even approximations remain costly with many features. SHAP's global feature analysis offers both importance and direction of feature influence, though it requires a full model and dataset, making it unsuitable for large datasets or fixed experimental designs. TreeSHAP [28], however, leverages tree-based model structures for efficient feature insights.

In Fig. 8, Shapley values for each feature on the f_3 function highlight x_{18} and x_{19} as the most influential parameters, evident from the wide spread of their points. Additionally, there is a positive correlation, as high feature values (red points) correspond to high SHAP values, indicating their strong contribution to increasing model output.

Evolutionary Computation Evolutionary computation (EC) [9, 29] refers to a family of optimization algorithms that are inspired by natural evolutionary processes, such as selection, mutation, and recombination. In a broader sense, EC algorithms belong to the class of black-box optimization algorithms, which aim at solving problems where the internal structure or analytical form of the objective function is unknown or inaccessible. In such cases, the optimization process only relies on evaluating input-output relationships without making assumptions about the underlying function's properties, such as smoothness, continuity, or differentiability. This makes black-box algorithms, like EC techniques, particularly well-suited for complex optimization tasks where the objective function may be noisy, discontinuous, or expensive to evaluate. Examples include optimizing physical experiments, engineering designs, or simulations, where evaluating the objective can be costly and/or derivative information is unavailable.

Evolution Strategies Evolution strategies (ESs) are a subclass of EC. Unlike other evolutionary algorithms, which focus on combinatorial problems, ES is particularly suited for continuous optimization tasks. The key idea is to iteratively evolve a population of candidate solutions by applying selection, mutation, and recombination to find an optimal or near-optimal solution to a given problem.

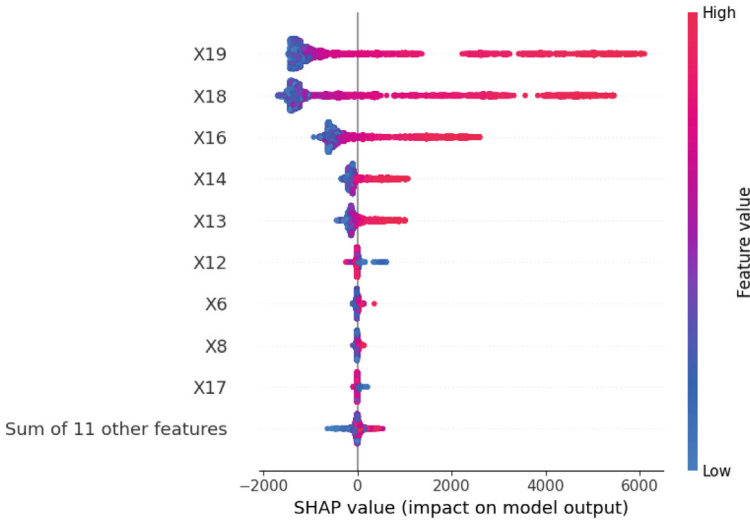


Fig. 8 SHAP summary plot of feature importances of f_3 . Red values correspond to high feature values and blue to low feature values. Each dot shows the SHAP values for an instance in the training dataset

Each candidate solution is represented as a vector in a continuous search space, along with a set of strategy parameters (often mutation step sizes) that control the extent of variations applied during mutation. ESs work according to few conceptual steps. They start with some *Initialization*, where a population of candidate solutions is generated with random parameters and initial step sizes. *Mutation* follows, in which each candidate is perturbed by adding random noise, scaled by the current step sizes, to explore the search space. Then comes a *Selection* step, where a subset of the population is selected based on a fitness function that evaluates the performance of each candidate. With *Recombination*, the parameters from multiple individuals are combined to create offspring with mixed attributes. In the end, we have *Adaptation* of step sizes: ESs include a self-adaptive mechanism where mutation step sizes are adjusted over generations to improve search efficiency, allowing finer or coarser exploration as needed.

Popular ES variants include the (μ, λ) -ES and $(\mu + \lambda)$ -ES strategies, where μ represents the number of parents and λ is the number of offspring. In (μ, λ) -ES, only offspring are eligible for selection, while in $(\mu + \lambda)$ -ES, parents and offspring compete together for the next generation. ESs are robust for high-dimensional and noisy objective functions and are widely applied in fields like engineering design, machine learning (e.g., neural architecture search), and control systems. Its adaptive step-size feature allows it to efficiently navigate rugged or multimodal landscapes, making it an attractive choice for complex optimization problems. In the broad scenario of ESs, the covariance matrix adaptation-evolution strategy (CMA-ES) [11] stands out

for its good performance and is considered to be the state-of-the-art algorithm in the field.

Covariance Matrix Adaptation Evolution Strategy CMA-ES [11] is a stochastic, or randomized, method for real-parameter (continuous) optimization of nonlinear, non-convex functions.

CMA-ES adapts its search strategy based on the covariance matrix of the distribution used to sample candidate solutions. The core of the CMA-ES algorithm revolves around updating a multivariate normal distribution that models locally the search space, allowing it to focus on promising regions over time. The algorithm works according to five key iterative steps:

1. Initialization of a mean vector $\mathbf{m}^{(0)}$ and covariance matrix $\mathbf{C}^{(0)}$ to define a multivariate normal distribution from which candidate solutions are samples and of a step-size $\sigma^{(0)}$ that controls the scale of the search.
2. Sampling of λ candidate solutions from the multivariate distribution and their evaluation. Hence the k -th candidate solution sampled at iteration t is $\mathbf{x}_k^{(t)} \sim \mathcal{N}(\mathbf{m}^{(t)}, (\sigma^{(t)})^2 \mathbf{C}^{(t)})$ for $k = 1, 2, \dots, \lambda$ and $t = 0, 1, 2, \dots$.
3. Selection of the μ best points from the λ offspring individuals to generate the parent population and update of the mean vector.
4. Update of the mean vector $\mathbf{m}^{(t+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^{(t)}$, where $\mathbf{x}_{i:\lambda}^{(t)}$ are the top μ solutions ranked by objective value, and w_i are positive recombination weights such that $\sum_{i=1}^{\mu} w_i = 1$.
5. Update of the covariance matrix to reflect the new distribution of the population: $\mathbf{C}^{(t+1)} = (1 - c_1 - c_{\mu})\mathbf{C}^{(t)} + c_1 \mathbf{p}_c^{(t+1)}(\mathbf{p}_c^{(t+1)})^T + c_{\mu} \sum_{i=1}^{\mu} w_i (\mathbf{x}_{i:\lambda}^{(t)} - \mathbf{m}^{(t)})(\mathbf{x}_{i:\lambda}^{(t)} - \mathbf{m}^{(t)})^T$, where c_1 and c_{μ} are learning rates for rank-one and rank- μ updates, respectively, and $\mathbf{p}_c^{(t+1)}$ is the evolution path that captures the momentum of successful moves over multiple generations.
6. Step-size adaptation to update the exploration/exploitation attitude. This is done by using a *path length control* mechanism, where the evolution path for the step-size is updated as $\mathbf{p}_{\sigma}^{(t+1)} = (1 - c_{\sigma})\mathbf{p}_{\sigma}^{(t)} + \sqrt{c_{\sigma}(2 - c_{\sigma})\mu_{\text{eff}} \frac{\mathbf{m}^{(t+1)} - \mathbf{m}^{(t)}}{\sigma^{(t)}}}$, with c_{σ} being the learning rate for step-size control, and μ_{eff} the effective number of parents (a measure of the population diversity). Hence, the new step-size is defined as $\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_{\sigma}}{d_{\sigma}} \left(\frac{\|\mathbf{p}_{\sigma}^{(t+1)}\|}{\mathbb{E}[\|\mathcal{N}(0, I)\|]} - 1\right)\right)$, where d_{σ} is the damping parameter for step-size adaptation, and $\mathbb{E}[\|\mathcal{N}(0, I)\|]$ is the expected length of a random vector sampled from a standard normal distribution.

Steps (2) to (5) are repeated iteratively until a stop criterion is met, e.g., a predefined number of objective function evaluations, convergence of the population, or improvement stagnation.

Compared to other ESs, CMA-ES is particularly effective for nonlinear optimization because it can adapt its search strategy to the landscape of the objective function. Specifically, for any convex-quadratic (ellipsoid) objective function, CMA-ES adjusts its sampling distribution to match the contour lines of the function, which is equivalent to locally transforming the function into a spherical (isotropic) one [11].

This capability is especially valuable for accelerating convergence on severely ill-conditioned or non-separable problems. Additionally, the covariance learning rates in CMA-ES prevent the population from collapsing into lower-dimensional subspaces, allowing for smaller population sizes than those typically required in standard ES, which contributes to faster convergence. Finally, the step-size control in CMA-ES helps prevent premature convergence to local optima [11].

3 Problem Definition

3.1 Study Description

In this chapter, we focus on a scenario where a dataset containing relevant input-output information is already available from previous analyses. This dataset serves as a valuable resource for extracting insights about the underlying system or model from which it was generated. Our objective is to explore how this information can be used to set up a new optimization problem on the same model, with an emphasis on improving the problem formulation. Specifically, we aim to refine the optimization process by identifying and prioritizing the most influential parameters, thereby reducing the dimensionality of the search space and enhancing the overall efficiency of the optimization.

Hence, we pose the following key question:

Can an existing dataset be analyzed to identify the most influential parameters affecting a target variable, enabling dimensionality reduction and creating a more focused and efficient optimization problem?

To answer, we designed an experimental study with the main steps summarized in Fig. 9. We analyze the performance of the optimizer, CMA-ES in our case, both

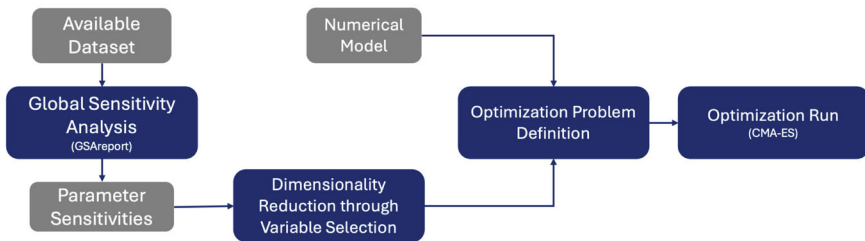


Fig. 9 Overview of the experimental study exploring GSA for dimensionality reduction and optimization efficiency. Using an existing dataset, GSA is applied to identify the most influential parameters, defining a reduced search space for the optimization

with and without preliminary dimensionality reduction through variable selection based on GSA indexes extracted from GSAreport. We assume that a large dataset from a prior experimental study is available, and we aim to extract as much useful information as possible.

3.2 Benchmarks from Engineering Design

Optimization of a star-shaped crash box Crashworthiness refers to a vehicle's ability to protect its occupants during an impact by absorbing and dissipating energy. Optimizing crashworthiness involves the usage of heavy FE simulations, which are costly and time-consuming. By using GSA to identify the most influential design variables, we reduce the dimensionality of the optimization problem and focus on the most critical aspects of the design. To verify the benefits of using GSA for variable screening in evolutionary computation, we test it on a benchmark from computational mechanics consisting of the maximization of the absorbed energy by a star-shaped crashbox while constraining the allowed maximum deformation. Figure 10a, b illustrates the model simulated to conduct our experiments before and after a frontal impact by a rigid wall that is axially loading the structure. The crashbox is fixed on a support on the other extreme.

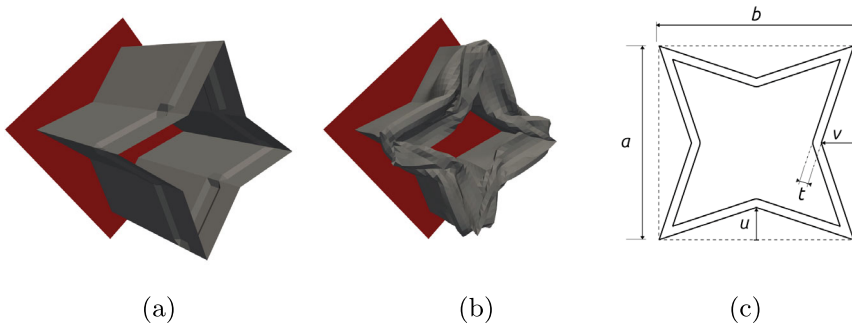


Fig. 10 Case Study 1. Various views and conditions of the crash box: **a** before impact, **b** after impact, and **c** shape parameters

The intrinsic nature of the optimization problem that we address is constrained. We aim to optimize the performance of a crashbox subjected to longitudinal loading, typical in frontal collision scenarios. The goal is to maximize the specific energy absorption (SEA), which measures how effectively the crashbox absorbs energy relative to its mass, while ensuring that the crashbox does not deform beyond 70% of its original length in the longitudinal direction. The problem is defined as follows:

$$\begin{aligned}
\max_{\mathbf{x}} \text{SEA}(\mathbf{x}) &= \frac{E_{\text{abs}}(\mathbf{x})}{m(\mathbf{x})} \\
\text{s.t. } L_{\text{deform}}(\mathbf{x}) &\leq 0.7L_0 \\
\mathbf{x} &\in [\mathbf{x}_l, \mathbf{x}_u].
\end{aligned} \tag{7}$$

The energy absorbed by the crashbox, E_{abs} , is defined as the difference between the initial kinetic energy and the residual kinetic energy of the system during impact: $E_{\text{abs}} = E_{\text{kin, initial}} - E_{\text{kin, residual}} = \frac{1}{2}m_{\text{impactor}}v_0^2 - \frac{1}{2}m_{\text{impactor}}v_{\text{res}}^2$. Here, m_{impactor} is the mass of the external impactor, v_0 is the initial velocity of the impactor at the moment of contact, and v_{res} is the residual velocity after impact. As we run the numerical simulation until we observe rebound of the impactor, we assume $E_{\text{kin, residual}} = 0$. In Eq. (7), $m(\mathbf{x})$ is the mass of the crashbox, which depends on the design variables $\mathbf{x}$ (e.g., material properties, geometry, etc.), $L_{\text{deform}}(\mathbf{x})$ is the deformation of the crashbox in the longitudinal direction as a function of $\mathbf{x}$, and L_0 is the initial length of the crashbox before deformation. As depicted in Fig. 10c, the vector of design variables is five-dimensional and defined as $\mathbf{x} = [a, b, u, v, t]$, where $a \in [60, 120]$ mm and $b \in [60, 120]$ mm are the height and width of the crash box section, $u \in [0, 30]$ mm and $v \in [0, 30]$ mm are intrusions in the x- and y-directions to shape the star-like section, and $t \in [0.7, 3]$ mm is the thickness of the component, perpendicular to the plane of projection.

As we want to limit our study to unconstrained optimization scenarios, we integrate the constraint in the definition of the objective as a penalty. Therefore, the actual optimization problem we address is:

$$\begin{aligned}
\max_{\mathbf{x}} \text{SEA}(\mathbf{x}) - p|L_{\text{deform}}(\mathbf{x}) - 0.7L_0| \\
\text{s.t. } \mathbf{x} \in [\mathbf{x}_l, \mathbf{x}_u].
\end{aligned} \tag{8}$$

In order to guarantee that the optimum of the problem satisfies the constraint, we set $p = 763$.

Optimization in sheet metal forming As second benchmark inspired by real-world engineering design, we consider the sheet metal forming test case introduced by Lehrer et al. [4]. In sheet metal forming, the drawability of the metal—its ability to be shaped without failure—is a key performance metric. Optimizing this process involves evaluating the influence of numerous factors, such as material properties, die geometry, and process parameters. Similar to crashworthiness optimization, FE simulations are commonly used to predict the performance of different configurations, but these simulations can be prohibitively expensive for high-dimensional problems. GSA allows us to screen variables and identify the most important factors affecting drawability. The numerical model used to generate data for GSAreport is shown in Fig. 11.

In this problem, we aim to optimize the deep drawing process, by minimizing the average thickness reduction (thinning) of the sheet metal during the deep drawing process, ensuring that the component maintains structural integrity and quality. By reducing thinning, the process will produce higher-quality components with better

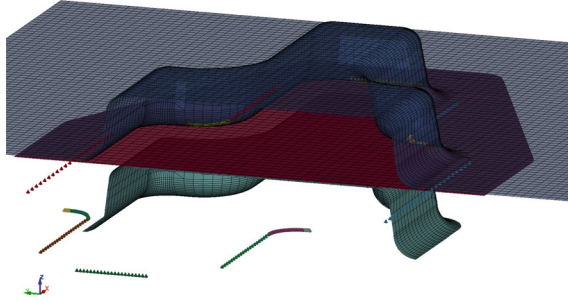


Fig. 11 Case Study 2. Arrangement of the geometries in the sheet metal forming process. From top to bottom, the figure shows the die (reference position), a gap-dependent blank, the blankholder, and the punch. The die moves toward the punch until the maximum forming length is reached, with drawbeads restraining material flow

structural properties. We consider the measure of thinning as defined by Guo et al. [30] and define an unconstrained optimization problem as follows:

$$\begin{aligned} \min_{\mathbf{x}} \text{Thinning}(\mathbf{x}) &= 1 - \frac{1}{N} \sum_{e=1}^N \frac{h'_e(\mathbf{x})}{h_0} \\ \text{s.t. } \mathbf{x} &\in [\mathbf{x}_l, \mathbf{x}_u], \end{aligned} \quad (9)$$

where N is the number of elements in the simulation model, h_0 is the initial thickness of the blank sheet, and h'_e represents the thickness of element e at the end of the drawing process. The parameter vector is again five-dimensional and defined as $\mathbf{x} = [t, r, \sigma_y, f_{\text{blkh}}, \mu_d]$, where $t \in [1.2, 2.0]$ mm is the sheet thickness, $r \in [0.8, 1.6]$ is the Lankford coefficient, $\sigma_y \in [140.0, 180.0]$ MPa is the yield strength of the blank, $f_{\text{blkh}} \in [130.0, 190.0]$ kN is the blankholder force, and $\mu_d \in [0.05, 0.12]$ is the dynamic coefficient of friction. This problem is treated as unconstrained because the variable bounds ensure that any solution resulting from the optimization will be feasible.

4 Experimental Setup

This section describes the settings used for GSA and optimization with the CMA-ES algorithm, including details on the benchmarks, sampling methods, and evaluation budgets across case studies. Details about our computational resources are also provided.

4.1 GSA Settings

For Benchmark 1, crashworthiness optimization of a star-shaped crashbox, we used GSAREport to compute sensitivity indexes based on multiple GSA methods described in Sect. 2.1. The dataset contained 2482 samples generated with three sampling schemes: 876 samples with Morris, 876 with Sobol', and 730 with Latin Hypercube Sampling. All parameter ranges were rescaled to fall within -5 to 5, covering a total of five parameters.

For Benchmark 2, deep drawing in sheet metal forming, we started from an existing dataset with 500 samples. Since this dataset was smaller, we used GSAREport to first fit a random forest model on the data (provided as x.csv and y.csv). The random forest model was then used to generate new samples to meet the requirements for each GSA method, which is needed to extract sensitivities when direct access to extensive data is limited.

4.2 CMA-ES Settings

For optimization, we used the CMA-ES algorithm to test the impact of GSA-based dimensionality reduction on convergence speed and solution quality.

We performed 250 evaluations to assess whether optimizing on a reduced dimensionality, according to the GSA results, could accelerate convergence on the high-fidelity model. Four-dimensional cases were compared: $D=1$, $D=3$, $D=5$ (original dimensions), and $D=50$ (adding 45 ineffective dimensions). Although we tested CMA-ES with four different step-size initializations (0.2, 1.0, 2.5 as default for CMA-ES on our search space, and 5.0), we will present results for $\sigma_0 = \{0.2, 5\}$, as these configurations produced the most diverse scenarios. We run 40 repetitions using different random seeds for statistical reliability. The settings used for CMA-ES are summarized in Table 1.

Table 1 CMA-ES parameter settings for case study

Parameter	Values
Evaluation budget	250
Dimensionality D	1, 3, 5, 50
Initial step-size σ_0	0.2, 1.0, 2.5, 5.0
Repetitions	40 (different seeds)

4.3 Software and Hardware

All sensitivity analysis and optimization procedures were carried out using the *GSAreport*² reporting tool for sensitivity extraction and a default version of CMA-ES from the *modcma*³ Python package for optimization. Numerical simulations were performed using OpenRadioss (Altair Engineering Inc.) for the crashbox benchmark and LS-Dyna (ANSYS Inc.) for the sheet metal forming test case.

All numerical experiments (full results are available on Zenodo [31]), except for the sheet metal drawing simulations (refer to [4]), were conducted on the Academic Leiden Interdisciplinary Cluster Environment (ALICE), a hybrid high-performing computing cluster with a capacity of 1234 TFlops, 1712 cores (3424 threads), and 17.9 TB of RAM. For all the experiments performed, only CPU nodes were used. The simulation of the 2482 configurations of the star-shaped crash box took 432 h of total CPU time. All of these simulations were run on a single processor with 4 threads and single precision. On the other hand, optimization experiments took a total of 46 h of total CPU time (including ablation).

5 Results

5.1 *GSAreport*

The first output generated by *GSAreport* for each of the considered case studies is a surface plot defined as a slice of a RF model fitted to the input data. Figure 12 presents the approximated landscapes for the objectives in the two optimization problems. Specifically, Fig. 12a depicts a RF approximation of the composite objective for the crashbox geometry optimization problem as formulated in Eq. (8), utilizing 2482 samples. Meanwhile, Fig. 12b illustrates the RF model of the thinning objective for the optimization problem defined in Eq. (9), using an available dataset of 500 samples. As both optimization problems operate in a five-dimensional parameter space, the figures display only two-dimensional slices of these landscapes. The axes in these plots represent the two most influential parameters identified through the GSA.

These surface plots illustrate how dataset size and the characteristics of each test case influence the model's response surface. The RF approximation for the crashbox test case, shown in Fig. 12a, reaches a mean R^2 score over three folds equal to $R^2 = 0.85996$. We observe that the surface plot shows a sharp change in the response across the slice defined by the thickness parameter X_4 . For low values, the response is relatively low, but as we move along the most influential direction, the surface rises steeply, following by a flatter area. As we are maximizing the target, this trend suggests an optimal region where also less influential parameters can play

² <https://github.com/nikivanstein/GSAreport>.

³ <https://pypi.org/project/modcma/>.

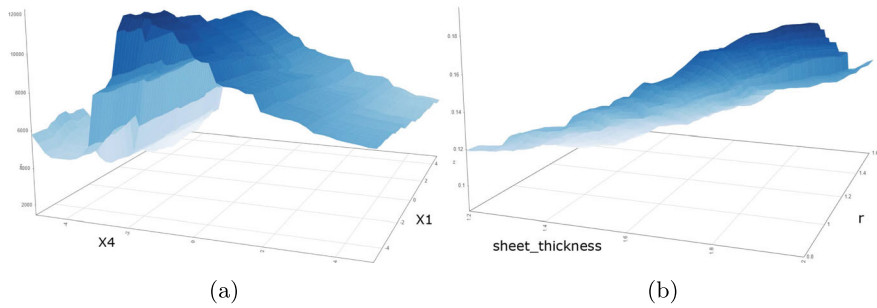


Fig. 12 GSAREport surface plots of a slice of the Random Forest model fitted to **a** a dataset of 2482 samples for the crashbox test case, and **b** an existing dataset of 500 samples for the sheet metal forming test case. The slice is defined by considering only the two most influential directions identified by GSA, with all other parameters fixed at the center of their range

a role in the refinement of the optimal solution. In the sheet metal forming test case (Fig. 12b), the surface plot shows a gradual, steady increase in response along both the most influential directions (t and r), with no intermediate peaks as observed in the crashbox test case. The response appears smoother, monotonic with a more moderate slope, which suggests that the 500 samples provide sufficient coverage for the model to capture the main trends in the main directions. This is also confirmed by a mean R^2 score of 0.96646, obtained using a standard threefold cross-validation approach.

Out of all GSA plots generated by GSAREport, we show in Fig. 13 the Morris covariance plots for each test case. All of them show five points, corresponding to the five design variables characterizing both Problem (8) and Problem (9). In the crash test case shown in Fig. 13a we see that, despite only one factor (thickness) has $\mu^* > 6000$, most points lie along or above the line $\sigma/\mu^* = 1$ and far away from the origin, suggesting that many parameters have substantial elementary effects with significant variability, indicating both high sensitivity and interaction effects. Both the fact that t has not only high mean, but also high variance, and that none of the variables seem to provide null contribution, show that reducing the dimensionality of the optimization problem to only a few parameters can restrict the optimization potential. However, the parameter sensitivities predicted by the reporting tool are confirmed by visually inspecting the output of simulations run on models where the most (t) and least (u, v) influential parameters are set at the extremes of their ranges. One can note that the main difference in longitudinal deformation is obtained when modifying t from 0.7 (Fig. 14a) to 3.0 (Fig. 14b, c). However, despite u and v are classified as the least influential by GSAREport, they also contribute in the deformation modes, as shown by Fig. 14b, c.

In the sheet metal forming test case (Fig. 13b), we observe a distinct pattern where many points are concentrated close to the origin, with relatively low values of both μ^* and σ . This suggests that the parameters corresponding to such points exhibit negligible sensitivity and interaction effects compared to the crash test cases. The only points showing higher μ^* values, hence having higher main effects, lie well below

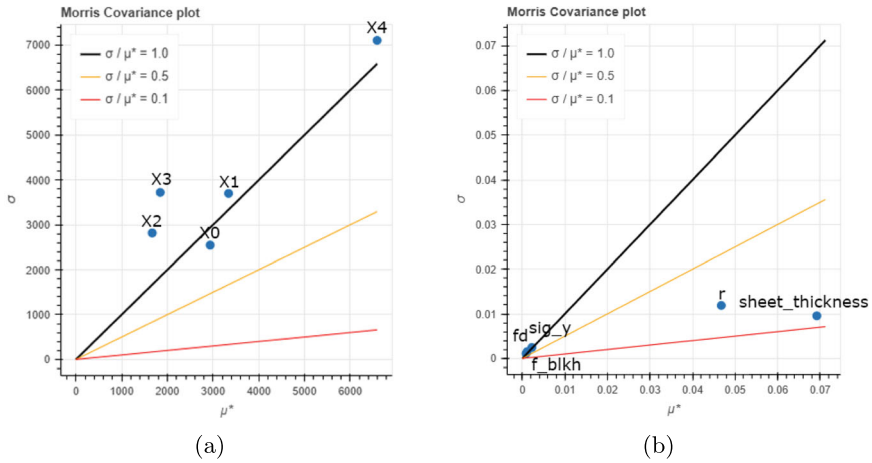


Fig. 13 Morris covariance plot generated via GSAreport for the **a** star-shaped crash box and **b** sheet metal forming test cases. μ^* , the absolute of the mean elementary effect, against σ , the standard deviation of the elementary effect. The lines represent different ratios of σ/μ^* , which indicates the relative importance and interaction of parameters

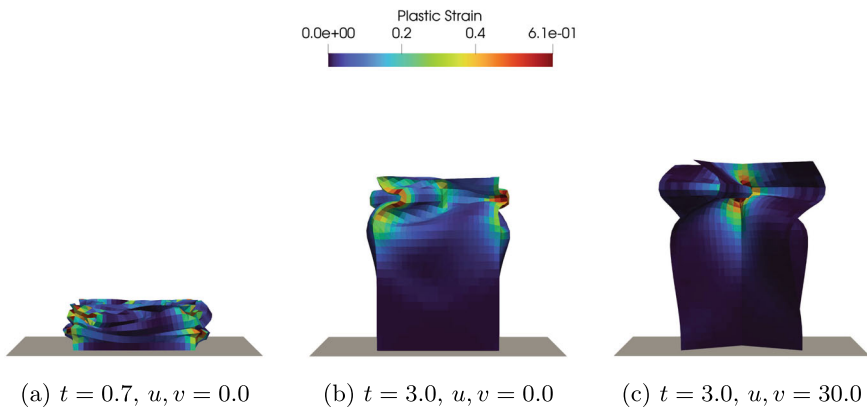


Fig. 14 Snapshots of extreme configurations of star-shaped crash box at 1.45 s of simulation time for fixed $a, b = 60.0$. In this figure, it is observed that the thickness has the most influence on the longitudinal deformation. Despite the parameters u and v have small influence on the optimization problem, these influence the deformation pattern of the crashbox. **b** shows a pattern comprising plastic hinges in a mixed axisymmetric (concertina) and non-axisymmetric (diamond) pattern, whereas in **c** the collapse mode is of Euler (buckling) type, with no plastic hinges and increased plastic deformation toward the middle axis of the crash box. For practical considerations, just axisymmetrical collapse patterns are preferred as these ensure the most energy absorption, and therefore u and v should be tuned and not disregarded during an optimization process

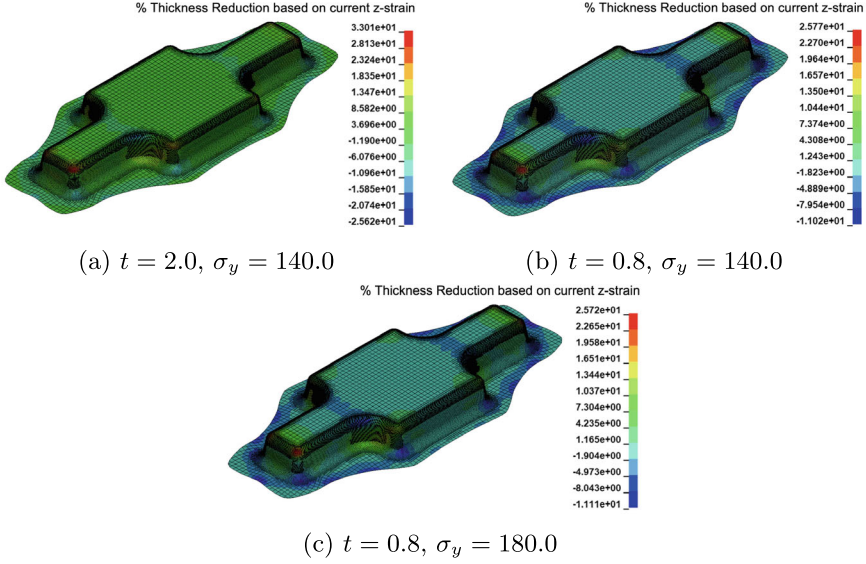


Fig. 15 Contour lines of the percentage thickness reduction based on the current z-strain. The yield strength σ_y is uninfluential on the thinning; by comparing (b) and (c), the alteration of the yield strength does not alter significantly on the overall thickness variation, whereas in a the range of thickness reduction is enlarged throughout the geometry, thus showing that the sheet thickness t plays a more significant role

the $\sigma/\mu^* = 0.5$ line, implying that interactions are minimal. In an optimization context, this suggests that a simplified optimization using only the two most influential parameters could effectively capture the main trends of the response while minimizing computational complexity. The contour lines of percentage thickness reduction in Fig. 15 provide a visual validation for the GSA analysis results. Specifically, the simulation outputs demonstrate that the yield strength σ_y has a negligible influence on the thinning process. This conclusion is supported by the comparison between cases in Fig. 15b, c, where variations in yield strength do not lead to significant changes in the overall thickness distribution. In contrast, Figure case 15a reveals that the sheet thickness t has a much greater impact, as indicated by the enlarged range of thickness reduction throughout the geometry. This confirms that t is a more critical parameter in determining the thinning behavior, aligning with the insights derived from the GSA analysis.

5.2 CMA-ES

To answer the key question in Sect. 3, after extracting parameter sensitivities with GSareport, we run CMA-ES on the original, reduced, and augmented space, for both

mechanical benchmarks. We assume to perform GSA on pre-existing datasets and use high-fidelity simulations for evaluating the target in the optimizations, which explains the limited evaluation budget of 200 evaluations for all the compared baselines.

Figure 16 shows the cumulative convergence plots for the star-shaped crashbox. Figure 16a shows the results for CMA-ES initialized with a small step-size of 0.2, while figure compares the same baselines, but CMA-ES starts from the default value of the initial step-size for a search space where each variable variation range is $[-5, 5]$, which is 2.5. Each line represents the objective function's progression for CMA-ES running on search spaces with dimensionality set to 1D, 3D, 5D (the original problem), and an augmented 50D where 45 additional ineffective axes are included. In the 1D and 3D baselines, the parameters that are considered uninfluential, and hence are not optimized, are randomly initialized and held constant. The target is always evaluated on a 5D array, which is an extension of the points CMA-ES queries in the 1D and 3D cases, and a reduction of the candidate solutions to only the effective dimensions in the 50D case. For each case, we run the original version of CMA-ES, with covariance matrix adaptation and denoted as “Anisotropic,” and a constrained version of CMA-ES, where only the step-size is adapted during the search, which is denoted as “Isotropic.” The parent and offspring population sizes are chosen based on the default values for the problem dimensionality seen by CMA-ES, i.e., 1, 3, 5, and 50 dimensions. We performed 40 repetitions for statistical significance.

With a smaller initial step-size (Fig. 16a), the reduced-dimensionality setups (1D and 3D) converge faster and achieve higher objective values within a limited evaluation budget (until evaluation 75) compared to the full-dimensional problem (5D) and the augmented problem (50D). This indicates that, with a small initial step-size, focusing the optimization on fewer, more influential parameters (as identified by

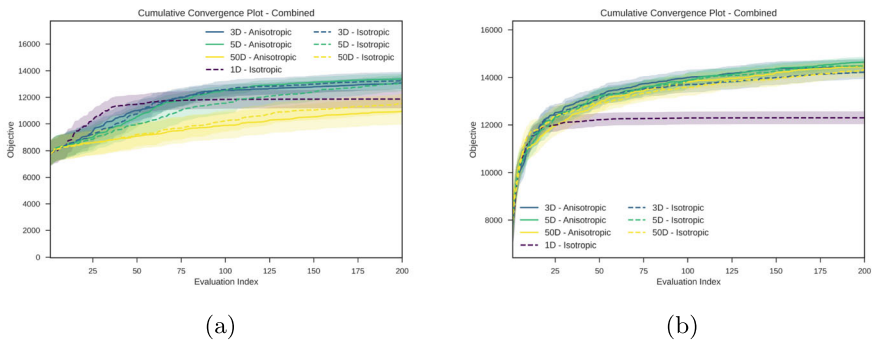


Fig. 16 Cumulative convergence plots for the star-shaped crashbox optimization problem with varying dimensionality, with two different initial step-sizes: **a** small initial step-size and **b** larger, default initial step-size set by CMA-ES. Each line represents the objective function's progression over 200 evaluations, with the dimensionality set to 1D, 3D, 5D (the original problem), and an augmented 50D where 45 additional ineffective axes are included. The “Anisotropic” and “Isotropic” tags refer to the default CMA-ES version, with covariance matrix adaptation, and the CMA-ES version without covariance matrix adaptation, respectively

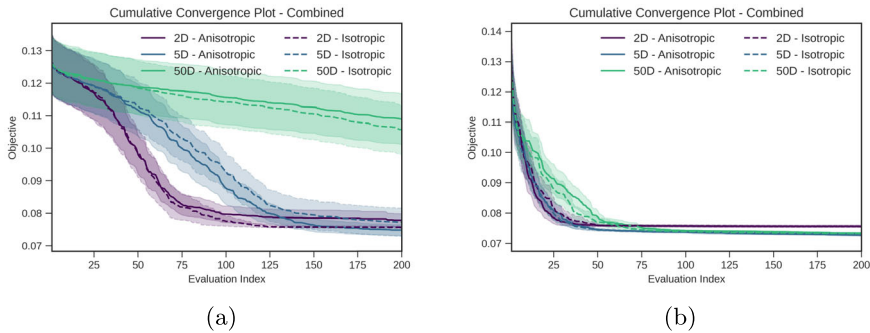


Fig. 17 Cumulative convergence plots for the sheet metal forming optimization problem with varying dimensionality, with two different initial step-sizes: **a** a small initial step-size and **b** a larger, default initial step-size set by CMA-ES. Each line represents the objective function’s progression over 200 evaluations, with the dimensionality set to 2D, 5D (the original problem), and an augmented 50D where 45 additional ineffective axes are included. The “Anisotropic” and “Isotropic” tags refer to the default CMA-ES version, with covariance matrix adaptation, and the CMA-ES version without covariance matrix adaptation, respectively

GSA) allows for better short-term convergence. However, this is likely because the small step-size limits the search’s exploration radius, making it beneficial to initially focus on the most influential parameters and using a smaller population size with more frequent updates. The adaptation of both the step-size and the covariance matrix during the run changes the respective ranking of the baselines at the end of the evaluation budget, where CMA-ES on the 3D and 5D scenarios perform best.

When the initial step-size is larger (see Fig. 16b), the advantage of dimensionality reduction diminishes. In this case, the 1D curve stabilizes after only a few iterations, while the other setups show statistically comparable performance. The similarity between the 3D and 5D curves indicates that excluding the two least influential parameters (u and v) does not result in a loss of significant contributions toward the target. However, reducing the problem’s dimensionality for CMA-ES does not accelerate the convergence trend, even for smaller evaluation budgets. This observation is further supported by the 50D case, where CMA-ES successfully identifies and focuses on the five truly influential parameters, adapting the search accordingly. Additionally, a larger population size (the higher the dimension the larger the population size by default) benefits the process, especially when the objective function has a well-defined global structure, as is the case here (see Fig. 10a). The default larger step-size enables broad initial exploration, allowing CMA-ES to effectively adapt to the search space and dynamically identify influential directions within the high-dimensional setting. This adaptability underscores CMA-ES’s ability to treat the full dimensionality of the problem without sacrificing convergence speed, especially in problems with a clear global structure.

Figure 17 presents cumulative convergence plots for CMA-ES applied to the sheet metal forming benchmark. In this analysis, we consider three setups: a reduced 2D space (from the original 5D problem), the full 5D problem, and an augmented 50D

problem with 45 ineffective dimensions added. As observed in the crash benchmark, when using a small initial step-size, CMA-ES on the reduced 2D problem achieves faster initial convergence, benefiting from the focused search in a lower-dimensional space. The 5D setup follows a similar pattern, reaching comparable objective values slightly slower than the 2D setup, reflecting the added complexity of the higher-dimensional space in a case where the additional three variables have indeed minimal influence on the target, as shown in Fig. 13b. In contrast, the 50D setup, which includes 45 ineffective dimensions, converges more slowly, and the overall improvement across the entire evaluation budget remains minimal. This outcome indicates that the added dimensions hinder the convergence process, with the optimizer struggling to ignore the irrelevant directions within the limited budget and search radius.

When examining the performance with the default (larger) initial step-size (shown in Fig. 17a), CMA-ES in the 50D case shows a notably slower initial convergence but eventually benefits from the larger step-size, enabling a more efficient search as the budget increases. Despite this improvement, the 50D setup still lags behind the 2D and 5D configurations, indicating the limitations of considering unnecessary dimensions if the available budget is small.

Compared to the crash benchmark, the sheet metal forming problem exhibits more pronounced differences in algorithm performance across different dimensionalities, regardless of the initial step-size. In both cases (small and large step-sizes), reducing the dimensionality to 2D provides an advantage for smaller evaluation budgets, though this effect is less evident with the larger step-size. The results emphasize that dimensionality reduction remains beneficial for CMA-ES in the early stages of optimization in the presence of variables that are indeed uninfluential on the target. However, with an appropriately tuned initial step-size and a sufficient evaluation budget, CMA-ES can effectively handle higher-dimensional problems by adaptively aligning the search direction based on information gathered about the function landscape.

6 Conclusions

The results presented in this chapter demonstrate that while GSA can effectively identify the most and least influential parameters within a problem, relying solely on GSA to define an optimization problem may inadvertently restrict the potential of the optimization process. In Sect. 5.1, by examining simulation outputs at the extremes of influential and non-influential parameter ranges, we observed that GSA allows us to understand the impact of different parameters on the target. However, a purely GSA-driven problem dimensionality reduction may overlook parameters that could become significant within the optimization search (Sect. 5.2). Similarly, limiting dimensionality based solely on domain expertise can also impose unnecessary constraints on the optimization landscape. For instance, in the star-shaped crash box test case, focusing exclusively on the thickness parameter disregards the

potential contributions of other parameters that might interact in unexpected ways during optimization.

A more effective approach, therefore, involves maintaining a broader parameter space that also includes parameters initially deemed less influential. Utilizing black-box optimization algorithms that allow the search to dynamically identify and prioritize relevant directions based on the collected evaluations, like CMA-ES, results to be a winning strategy. This approach capitalizes on the optimizer's ability to self-adjust and focus on impactful parameter combinations, rather than relying exclusively on predefined parameter rankings.

This message is particularly relevant for applied researchers who may prioritize achieving a precise problem definition at the outset, potentially underestimating the adaptive capabilities of advanced optimizers. While an ill-conceived problem definition that neglects key parameters cannot be compensated for by the optimizer, an over-dimensioned parameter space provides flexibility for the optimizer to refine the search efficiently. In summary, retaining a higher dimensionality and leveraging the optimizer's capabilities can lead to more robust solutions than coming up with an over-precise, and potentially wrong, optimization problem formulation.

Acknowledgements We would like to thank Nikolaus Hansen for insightful discussions on CMA-ES, which deepened our understanding of its complexities, particularly regarding ineffective dimensions. We also thank Fabian Duddeck and his research group, especially Tobias Lehrer and Arne Kaps, for providing mechanical test cases that enriched our analysis with practical scenarios from engineering design.

References

1. Iooss, B., Lemaître, P.: A review on global sensitivity analysis methods (2014). <http://arxiv.org/abs/1404.2405> [arXiv:1404.2405](https://arxiv.org/abs/1404.2405) [math, stat]
2. van Stein, B., Raponi, E., Sadeghi, Z., Bouman, N., van Ham, R.C., Bäck, T.: A comparison of global sensitivity analysis methods for explainable AI with an application in genomic prediction. *IEEE Access*, 1 (2022). <https://doi.org/10.1109/ACCESS.2022.3210175>
3. Büttner, J., Schwarz, S., Schumacher, A., Bäck, T.: Global sensitivity matrix for vehicle development. *ATZ World*. **123**(3), 26–31 (2021). <https://doi.org/10.1007/s38311-020-0630-1>
4. Lehrer, T., Kaps, A., Lepenies, I., Raponi, E., Wagner, M., Duddeck, F.: Complementing drawability assessment of deep-drawn components with surrogate-based global sensitivity analysis. *ASCE-ASME J. Risk Uncertainty Eng. Syst., Part B: Mech. Eng.* **10**(3), 031204 (2024). <https://doi.org/10.1115/1.4065143>
5. Iooss, B., Lemaître, P.: A review on global sensitivity analysis methods (2014)
6. Chang, Q., Zhou, C., Faes, M.G.R., Valdebenito, M.A.: Design optimization with variable screening by interval-based sensitivity analysis. *ASCE-ASME J. Risk Uncertainty Eng. Syst., Part A: Civil Eng.* **10**(3), 04024039 (2024). <https://doi.org/10.1061/AJRUA6.RUENG-1266>
7. Giammaria, V., Del Bianco, G., Raponi, E., Fiumarella, D., Ciardiello, R., Boria, S., Duddeck, F., Belingardi, G.: Material parameter optimization of flax/epoxy composite laminates under low-velocity impact. *Comp. Struct.*, 117303 (2023). <https://doi.org/10.1016/j.compstruct.2023.117303>
8. Raponi, E., Fiumarella, D., Boria, S., Scattina, A., Belingardi, G.: Methodology for parameter identification on a thermoplastic composite crash absorber by the Sequential Response Surface

- Method and Efficient Global Optimization. *Comp. Struct.*, 114646 (2021). <https://doi.org/10.1016/j.compstruct.2021.114646>
9. Back, T.: *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press (1996)
 10. Greiner, D., Gaspar-Cunha, A., Hernández-Sosa, D., Minisci, E., Zamuda, A., Hernández-Sosa, D., Minisci, E., Zamuda, A. (eds.): *Evolutionary Algorithms in Engineering Design Optimization*. MDPI—Multidisciplinary Digital Publishing Institute (2022). <https://doi.org/10.3390/books978-3-0365-2715-4>
 11. Hansen, N.: The CMA Evolution Strategy: A Tutorial (2023). <https://arxiv.org/abs/1604.00772>
 12. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comp.* **11**(1), 1–18 (2003)
 13. van Stein, B., Raponi, E.: GSAreport: easy to use global sensitivity reporting. *J. Open Source Softw.* **7**(78), 4721 (2022). <https://doi.org/10.21105/joss.04721>
 14. Becker, W.: Metafunctions for benchmarking in sensitivity analysis. *Reliability Eng. Syst. Saf.* **204**, 107189 (2020)
 15. Cheng, K., Lu, Z., Ling, C., Zhou, S.: Surrogate-assisted global sensitivity analysis: an overview. *Structural Multidiscipl. Optim.* **61**(3), 1187–1213 (2020)
 16. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. *Optim. Methods Softw.* **36**(1), 114–144 (2021). <https://doi.org/10.1080/10556788.2020.1808977>
 17. Andrea, S., Marco, R., Terry, A., Francesca, C., Jessica, C., Debora, G., Michaela, S., Stefano, T.: *Global Sensitivity Analysis: The Primer*. Ist John Wiley & Sons, The Atrium, Southern Gate, Chichester, England (2008)
 18. Sobol, I.M.: Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Math. Comp. Simul.* **55**(1–3), 271–280 (2001)
 19. Tarantola, S., Gatelli, D., Mara, T.A.: Random balance designs for the estimation of first order global sensitivity indices. *Reliability Eng. Syst. Saf.* **91**(6), 717–727 (2006)
 20. Cukier, R., Fortuin, C., Shuler, K.E., Petschek, A., Schaibly, J.H.: Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. *J. Chem. Phys.* **59**(8), 3873–3878 (1973)
 21. Kucherenko, S., Iooss, B.: Derivative-based global sensitivity measures. In: *Handbook of Uncertainty Quantification*, pp. 1–24 (2015). https://doi.org/10.1007/978-3-319-11259-6_36-1
 22. Morris, M.D.: Factorial sampling plans for preliminary computational experiments. *Technomet.* **33**(2), 161–174 (1991). <https://doi.org/10.1080/00401706.1991.10484804>
 23. Campolongo, F., Cariboni, J., Saltelli, A.: An effective screening design for sensitivity analysis of large models. *Environmental Modell. Softw.* **22**(10), 1509–1518 (2007). <https://doi.org/10.1016/j.envsoft.2006.10.004>
 24. Plischke, E., Borgonovo, E., Smith, C.L.: Global sensitivity measures from given data. *European J. Operat. Res.* **226**(3), 536–550 (2013). <https://EconPapers.repec.org/RePEc:eee:ejores:v:226:y:2013:i:3:p:536-550>
 25. Pianosi, F., Wagener, T.: A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. *Environ. Model. Softw.* **67**, 1–11 (2015). <https://doi.org/10.1016/j.envsoft.2015.01.004>
 26. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
 27. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc. (2017). <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>
 28. Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I.: From local explanations to global understanding with explainable AI for trees. *Nature Mach. Intell.* **2**(1), 2522–5839 (2020)

29. Bäck, T.H.W., Kononova, A.V., van Stein, B., Wang, H., Antonov, K.A., Kalkreuth, R.T., de Nobel, J., Vermetten, D., de Winter, R., Ye, F.: Evolutionary algorithms for parameter optimization-thirty years later. *Evol. Comp.* **31**(2), 81–122 (2023). https://doi.org/10.1162/evco_a_00325
30. Guo, Y.Q., Batoz, J.L., Naceur, H., Bouabdallah, S., Mercier, F., Barlet, O.: Recent developments on the analysis and optimum design of sheet metal forming parts using a simplified inverse approach. *Comp. Struct.* (2000)
31. Raponi, E.: Global Sensitivity Analysis is Not Always Beneficial for Evolutionary Computation: Additional Results (2024)