algorithms

MDPI

Article

Test and Validation of the Surrogate-Based, Multi-Objective GOMORS Algorithm against the NSGA-II Algorithm in Structural Shape Optimization

Yannis Werner [1,*], Tim van Hout [2], Vijey Subramani Raja Gopalan [2] and Thomas Vietor [1]

[1] Institute for Engineering Design, Technische Universität Braunschweig, Hermann-Blenk-Str. 42, 38108 Brunswick, Germany; t.vietor@tu-braunschweig.de
[2] Technische Universität Braunschweig, 38106 Brunswick, Germany; tim.j.vanhout@gmail.com (T.v.H.); v.raja-gopalan@tu-braunschweig.de (V.S.R.G.)
* Correspondence: y.werner@tu-braunschweig.de; Tel.: +49-531-391-65013

Abstract: Nowadays, product development times are constantly decreasing, while the requirements for the products themselves increased significantly in the last decade. Hence, manufacturers use Computer-Aided Design (CAD) and Finite-Element (FE) Methods to develop better products in shorter times. Shape optimization offers great potential to improve many high-fidelity, numerical problems such as the crash performance of cars. Still, the proper selection of optimization algorithms provides a great potential to increase the speed of the optimization time. This article reviews the optimization performance of two different algorithms and frameworks for the structural behavior of a b-pillar. A b-pillar is the structural component between a car's front and rear door, loaded under static and crash requirements. Furthermore, the validation of the algorithm includes a feasibility constraint. Recently, an optimization routine was implemented and validated for a Non-dominated Sorting Genetic Algorithm (NSGA-II) implementation. Different multi-objective optimization algorithms are reviewed and methodically ranked in a comparative study by given criteria. In this case, the Gap Optimized Multi-Objective Optimization using Response Surfaces (GOMORS) framework is chosen and implemented into the existing Institut für Konstruktionstechnik Optimizes Shapes (IKOS) framework. Specifically, the article compares the NSGA-II and GOMORS directly for a linear, non-linear, and feasibility optimization scenario. The results show that the GOMORS outperforms the NSGA-II vastly regarding the number of function calls and Pareto-efficient results without the feasibility constraint. The problem is reformulated to an unconstrained, three-objective optimization problem to analyze the influence of the constraint. The constrained and unconstrained approaches show equal performance for the given scenarios. Accordingly, the authors provide a clear recommendation towards the surrogate-based GOMORS for costly and multi-objective evaluations. Furthermore, the algorithm can handle the feasibility constraint properly when formulated as an objective function and as a constraint.

Keywords: multi-objective optimization; NSGA-II; GOMORS; crash optimization; structural optimization; shape optimization; engineering optimization

Citation: Werner, Y.; van Hout, T.; Raja Gopalan, V.S.; Vietor, T. Test and Validation of the Surrogate-Based, Multi-Objective GOMORS Algorithm against the NSGA-II Algorithm in Structural Shape Optimization. *Algorithms* **2022**, *15*, 46. https://doi.org/10.3390/a15020046

Academic Editors: Szymon Łukasik and Piotr A. Kowalski

Received: 21 December 2021
Accepted: 26 January 2022
Published: 28 January 2022

1. Introduction

A common way to save weight in structural design is to use optimization approaches in the product development process (PDP) [1,2], such as topology, shape, and size optimization. Certainly, engineers apply the mentioned optimization types in different stages of the PDP, for instance, the preliminary or conceptual design. Engineers mostly use shape optimization in all design-relevant stages of the PDP. Primarily, Computer-aided Design (CAD) and Non-Uniform Rational B-Splines (NURBS)-based approaches vary the geometry of parts before the geometry is meshed and calculated [3,4]. Especially in automotive design, complex structures with multiple requirements exist, with crash safety being the

most challenging. Due to decreasing computational cost, crash calculations are state of the art in the automotive design process regarding the dimensioning of structural components. Notably, optimizing the Body-in-White (BIW) structures with shape optimization typically results in costly numerical evaluations. Each evaluated data point represents a linear and, often, also a non-linear calculation. Hence, the potential of automotive BIW crash optimization strongly depends on optimization algorithms that precisely find a good design by using a minimum of the expensive function evaluations. For instance, good designs in a crash are lightweight and fulfill all requirements towards crash safety, package, and structural stiffness [5–7].

There are different optimization algorithms that assist in minimizing the objective and constraint functions. Rayamajhi and Hunkeler [8] optimized the mass, energy absorbed, maximum section force, deflection, and intrusion of a bumper structure under crash load. Common engineering optimization problems such as crash optimization are black-box functions. They usually exhibit numerical noise, multimodal behavior, and uncertainties. In this case, deterministic search strategies cannot be used, as deterministic methods require the problem to exhibit specific mathematical characteristics. These characteristics are not given in crash optimization [9,10]. The most common algorithm type, which operates this optimization problem sufficiently and effectively, are heuristic and metaheuristic optimization approaches. Problems with these approaches arise when black-box evaluations are costly. In this case, surrogate models can significantly improve the amount of high-fidelity computer simulations. In particular, this is even more critical for multi-objective optimization tasks [9]. Indeed, a surrogate model is a mathematical model that provides cheaper responses to the input–output relationship for computationally expensive jobs [11]. Optimizing BIW components in automotive crashes combines multiple challenges, such as multimodality, costly functional evaluations for numerical analysis, numerical noise, and multi-objective target functions. For instance, suitable surrogates and optimization algorithms play an outstanding role in these optimization tasks [12–14]. Such processes use different optimization strategies such as Single-Objective Optimizations (SOOs) to handle one and Multi-Objective Optimizations (MOOs) to handle multiple objective functions [15]. In SOOs, all objectives and constraints are summed to a single objective function using approaches such as the weighted-sum-method. The algorithm optimizes only one objective function [16]. At the same time, MOO algorithms offer a more extensive solution space but lead to a higher dimensionality of the optimization problem by creating Pareto-efficient designs [15].

During the development process, the main obstacles in BIW design are the strict geometry and design requirements, which need to be considered [17]. Engineering optimization typically fails to handle these geometry-based constraints during optimization. For this reason, a new approach for the handling of geometrical constraints in package and design optimization was implemented in a Python-based framework called IKOS [18] by the authors. The proof of concept for the IKOS configuration used the well-established NSGA-II algorithm, following [19]. In [20], the authors applied the approach to a more complex and automotive-relevant preliminary design scenario of a b-pillar. The optimization showed that the NSGA-II tends to optimize slow and that the use of surrogate models could help to improve the speed of convergence. Consequently, this article involves selecting and implementing a surrogate-based optimization algorithm into the existing framework.

The Materials and Methods section gives a broad overview of black-box optimization algorithms and introduces the most crucial metamodeling and optimization strategies. The section summarizes criteria towards the performance of the metamodels, defines optimization strategies, and ranks the different optimization algorithms by their degree of criteria fulfillment. A surrogate-assisted framework called GOMORS is selected. Finally, the section briefly introduces the NSGA-II algorithm and the GOMORS framework to understand the underlying principles better. The Results section compares the performance of the NSGA-II to the implementation of the selected GOMORS framework in a comparative study [21]. The optimizations use the implemented b-pillar from [20] as a model. Furthermore, different load cases are analyzed for both algorithms: a torsional (linear), a

side-impact (non-linear), and the side-impact load case under the geometrical constraint. Finally, the constraint is changed to an objective function to test different configurations of the GOMORS. The Discussion section analyzes and classifies the results of the study. Finally, the Conclusion and Outlook closes the article and gives an outlook on further possible areas of application of the method.

2. Materials and Methods

For structural optimization in the conceptual and preliminary design, mainly two different types of optimizations are used: topology and shape optimization. Only shape and size optimization are applied [3]. Accordingly, engineers often perform shape and size optimizations in preliminary and detailed design stages. Both approaches rely on optimizing possible design parameters of the structure, mostly with heuristics or metaheuristics. Shape variation uses two main techniques: Mesh- and CAD-based [3,6] processes. CAD-based processes can be realized with the combination of CAD programs for the geometry and pre-processors for the mesh-generation and solver-deck setup or with CAD programs with integrated pre-processor, such as SFE CONCEPT [4,22,23]. In particular, SFE CONCEPT offers the ability to generate structures quickly and pre-process Finite Element (FE) decks easily to follow Finite Element Analysis (FEA). Hence, the SFE CONCEPT tool takes over the design and pre-processing to optimize structural components [18,24,25]. A vector of design variables (DVs) x describes the geometry variably. The upper x^{upper} and lower x^{lower} design variable boundaries define the geometrical limits of optimized parts. The response of the calculated FE simulation leads to the response function, which can be subdivided into the objective and constraint function, such as mass, intrusion, or stiffness [15]. Ryberg and Domeij [9] formulate the general form of an optimization problem as:

$$\begin{aligned} &\min_{x} \; f(x) \\ &\quad g(x) \;\leq\; 0 \\ \textit{subject to} \quad &\quad h(x) = 0 \\ &x^{lower} \leq x \leq x^{upper} \end{aligned} \tag{1}$$

Determining the design variables x that minimize the objective function f is the target of the optimization problem. The vectors g and h represent the inequality and equality constraints. Each objective and constraint function value depends on the design variables x. Feasible design points fulfill all constraints, while the design points which fail to comply with the constraints are infeasible. If the optimization problem has no constraints, it is called an unconstrained problem, while the contrary is called a constrained problem. If the problem has more than one objective function, it is called a multi-objective optimization. By the reformulation of the problem with m objective functions, the following condition results:

$$\begin{aligned} &\min_{x} \; f_1(x), \ldots, f_m(x) \\ &\textit{subject to } g(x) \;\leq\; 0 \end{aligned} \tag{2}$$

2.1. Surrogate Models and Optimization Methods

From the viewpoint of the optimizer, the system of geometry change, pre-processing, calculation, and evaluation is a black box. Surrogate models can predict the dependency of the constraints and objective functions on the input variables. Furthermore, different coupling methods of surrogates and optimization algorithms exist, classified as decoupled, surrogate-driven, and surrogate-assisted methods. Decoupled methods are the most straightforward approach as combining a surrogate and an optimization algorithm requires less effort. The surrogate models approximate the target functions, and the optimization algorithm searches the optima and predicts a good solution. The framework updates the surrogate model accordingly until it depletes the budget for expensive evaluations. The surrogate model is intrinsically built into the optimization procedure in surrogate-driven methods such as Bayesian optimization techniques, which use probabilistic surrogate mod-

els with sequential updates. A rather unpopular approach is surrogate-assisted methods, where surrogate models support the algorithm in a specific domain, e.g., local search [11].

An extensive range of optimization algorithms is applicable for this kind of problem. As the behavior response functions are typically highly multimodal and not present in their mathematical closed form, one needs to use heuristic or metaheuristic approaches for the optimization [9,26]. Metaheuristics often adopt biological behavior for the optimization of the target function. Hence, a broad spectrum of different optimization approaches exists, which allow the optimization of black-box problems in varying disciplines. Paas and van Dijk [5] compare a Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) to an NSGA-II and apply both algorithms to a BIW-optimization task. The NSGA-II converged faster to the Pareto-frontier and covered a more significant part of the solution space. The benchmark results of different automotive crash optimizations are summarized in [7]. The comparison of an Evolutionary Algorithm (EA), Genetic Algorithms (GAs), and Simulated Annealing (SA) for Noise, Vibration, and Harshness (NVH) in BIW optimization showed that for the GA and EA, the mutation and population size play an outstanding role, and so do the number of parents and children. Elitism improves multi-objective optimizations, while non-elitist approaches work well for single-objective optimizations. Furthermore, the EA outperformed the other algorithms under high noise and non-linearity. Due to the diversification and good exploration, EAs tend to be very successful, as they consider the whole solution space in early conceptual design. Indeed, one should consider the more exploitive algorithm should. All of these metaheuristics differ in their intensification and diversification [27]. The Bacterial Foraging Algorithm (BFO) tends to be less performant than the GA, Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) in the scenario of an aeroelastic wing shape optimization. Therefore, modified Cooperative Bacterial Foraging Optimization (CFBO) tends to perform like the mentioned algorithm regarding the optimization performance [28]. Single algorithms have characteristics of either good exploitation (intensification) or exploration (diversification). Thus, new approaches arise that combine the benefits of intensification and diversification by combining different metaheuristics, e.g., presented in [29]. Due to its simplicity and effectiveness, recent research implemented the NSGA-II into the IKOS framework. Furthermore, it was validated using two different generic b-pillar structures under a geometrical constraint [19,30]. Due to the high computational demand of crash simulations, surrogate models should improve optimization speed [9,12].

The following section reviews frameworks and Python libraries, which use optimization algorithms combined with surrogate models. Wang and Chai combine the PSO with a BFO to form the PSO-BFO algorithm [31]. The algorithm uses a Radial Basis Function (RBF) surrogate model to optimize the behavior of a full-frontal impact and side impact of an SFE CONCEPT BIW model. While the PSO tends to have excellent explorative behavior, the BFO is better for exploitation. Hence, combining both algorithms with a surrogate improves the structural performance in both crash scenarios. The combined algorithm outperformed the PSO-GA and the simple PSO and BFO.

Besides the structural applications, different black-box optimization approaches exist. The following paragraph mentions notable exemplary implementations. Based on the MOCE+ algorithm, Haber and Beruvides propose the Simple Multi-Objective Cross-Entropy (SMOCE) algorithm [32]. The algorithm uses elitism clustering by histograms of the objective functions. One of the main benefits of this algorithm is its robustness and the low amount of algorithm parameters that makes the algorithm very consistent. The authors performed a comparative study for commonly used test problems with two objective functions. The study found that the SMOCE outperforms the MOCE+, NSGA-II, MOEA/D, MOPSO, SPEA-II, and PESA-II for hyper area exploration. The other algorithm may benefit from a smaller sampling size and generations (epochs); the SMOCE appears suitable for inexpensive function calls, e.g., analytical functions, as demonstrated in the article. Belakaria and Deshwal describe the Max-value Entropy Search for Multi-Objective Optimization (MESMO) with Bayesian Optimization for Network on Chip optimizations in [33]. The

MESMO uses an output-space, entropy-based acquisition function to detect evaluation points, which leads to the efficient detection of the Pareto-frontier points. The authors compare MESMO to different Bayes approaches, which are Pareto-efficient global optimization (ParEGO), the Predictive entropy search for Multi-objective Bayesian Optimization (PESMO), S-Metric-Selection-based Efficient Global Optimization (SMSego) expected improvement in Pareto hypervolume (EHI), and the probability of improvement in stepwise uncertainty reduction (SUR). The paper compares all algorithms for the same two synthetic benchmark problems, which combine classical, single-objective benchmark problems. MESMO outperforms the other algorithms in terms of robustness and convergence.

Another approach is the GOMORS framework, a multi-objective EA that uses RBF-surrogate models [21]. The EA predicts new function points based on the surrogate model and generates possible candidates. A few points are selected based on rules, which balance exploration, exploitation, and frontier diversification. The selected candidates are then evaluated based on the costly function evaluation. The expensive black-box evaluations then update the RBF-surrogates, which serve as the foundation for further optimizations with the EA. Indeed, different embedded EAs have been evaluated, such as the NSGA-II, MOEA/D, and AMALGAM. The AMALGAM outperformed the other algorithms slightly. The GOMORS was then directly compared to the ParEGO and NSGA-II on a groundwater remediation problem and synthetic test functions. The GOMORS topped the ParEGO [34] and the NSGA-II regarding the effectivity. The authors state GOMORS to find suitable solutions under a low amount of costly function evaluations.

2.2. IKOS: The Framework

As this article deals with implementing a new optimization algorithm into an existing optimization framework, the following paragraph briefly introduces the present implementation. The basic framework and approach are described in [18]. In addition, an extension of the method and more detailed optimizations are performed in [20]. The general framework is a Python 3-based implementation for the multi-objective structural shape and size optimization of linear and non-linear load cases for structural components. In particular, the framework currently uses SFE CONCEPT as a CAD-modeling tool and as a pre-processor. The solver OptiStruct calculates the linear load cases, and RADIOSS deals with the non-linear calculation of the structural components. Both load cases are evaluated individually or as multidisciplinary optimization. The framework uses the multi-objective NSGA-II as an optimization algorithm [19]. Again, one can formulate the optimization problem as a black-box function, wherein the calculation time for the non-linear evaluations is high and hence costly. The NSGA-II optimized the parts but showed slow convergence ratios and many function evaluations. Still, the NSGA-II is a robust algorithm for optimizing a b-pillar under the given geometrical constraint [20]. Furthermore, the framework allows for the parallelization of the calculations and server support. Hence, the user can surveil the optimization routine locally while the algorithm performs costly calculations on a server of choice, as shown in Figure 1.

Figure 2 shows the basic feasibility detection method. The geometrical intersection is evaluated in ANSA and processed to the framework. The point "Assess Feasibility" summarizes the unique novelty in the framework, which is the optimization of structural components under the geometrical constraint.

The infeasible designs are penalized following the general approaches presented in the articles mentioned above, which the following rule can summarize:

$$f(x)_m = \begin{cases} f(x)_m, & penalty = 0 \\ f(x)_m \times penaltyfactor(penalty), & 1 > penalty > 0 \end{cases} \quad (3)$$

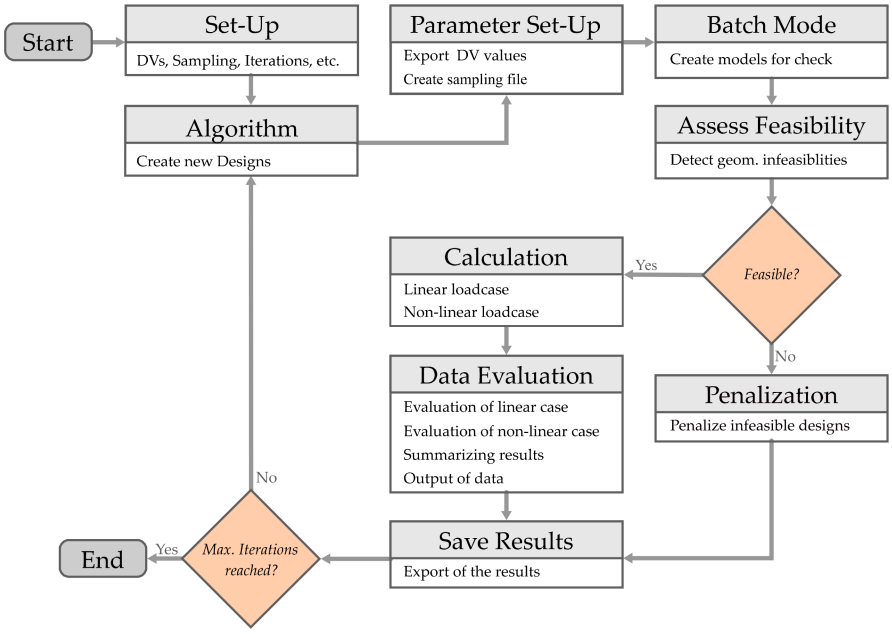

Figure 1. The flowchart of the IKOS framework describes the single process steps of the optimization process.

(**a**) (**b**)

Figure 2. (**a**) shows the b-pillar and hull surface, while (**b**) shows the b-pillar enveloped by the hull surface.

If the design x intersects with the boundary surface, the intersecting surface area is calculated to the penalty factor. Accordingly, the infeasibility factor penalty varies between zero and one. A design is fully infeasible ($penalty = 1$) if the hull surface penetrates every element of the proposed design. The factor $penalty$ turns 0 if the check finds no penetrations. For every mth objective function, the objective function value is penalized.

2.3. Requirements towards Optimization Algorithms

The missing availability of criteria for choosing a proper optimization algorithm is one of the fundamental problems for engineers in optimization. Most publications use different test functions, and the authors validate the algorithms for computationally inexpensive test functions. Another problem is the number of validated objectives. Furthermore, the behavior of the black box (multimodality, non-linearity) and requirements from the framework itself (particular environment, e.g., Python 3) need to be considered. Papers and articles present different measures to choose a suitable optimization algorithm. Hence, this paragraph briefly reviews the requirements for the well-founded selection of an optimization algorithm. Chugh and Sindhya [35] surveyed a wide variety of algorithms; the authors found that for 45 published algorithms from 2008 to 2016, a maximum of three objective functions, mainly two, were evaluated. The amount of DVs commonly lies below 30. Mostly, no time for the training of the surrogate models is mentioned, which is founded on the argument that the expensive function evaluations are predominant. Still,

training can be a significant factor for large data sets or quick assessment. In addition, the factors are mainly relative to the other algorithms analyzed in a specific paper: the following criteria can be formulated considering the efficiency of the algorithms:

- The dimension of the objective and decision space;
- Maximum number of expensive function evaluations;
- Applicable problem characteristics (e.g., multimodality, divided Pareto-frontier).

Hence, the authors summarize that a good algorithm for optimizing black-box functions generates legitimate solutions with few evaluations even for unknown characteristics. The main questions towards the choice of algorithms are stated in [36], which are:

- Which algorithm deals the best with the problem?
- For which kind of problem can the algorithm be used?

Commonly, both questions can only be answered after testing the algorithm for the problem. The user's experience influences the algorithm's choice, as do the software, licensing, calculation (evaluation) cost, and the maximum time for the optimization. Because of their discontinuity, the numerical noise, and the multimodal character of crash simulations' local search strategies are not applicable for multi-objective optimization. Global search algorithms such as SAs, GAs, and EAs can usually be applied [7], while newer (adaptive) approaches focus on a balanced behavior of exploration and exploitation [12].

The following section reviews the choice, implementation, and validation of one of the mentioned algorithms into the IKOS framework.

2.4. Choice of the Optimization Algorithm

The mentioned algorithms are all potential candidates for the optimization problem. The linear load case does not play an important role, as calculation times are about 30 s. With 30 min for a simple and more than 12 h for a complex crash calculation, the algorithm and its efficiency are more important for the non-linear optimization time. From the survey, and according to the experience with the framework, the following requirements are suggested:

- Multi-objective optimization algorithm;
- Effective search strategy;
- A limited number of costly function evaluations (e.g., use of surrogates);
- Able to deal with the constraint function;
- Parallelization possible.

As all these criteria are case-dependent and notably hard to evaluate for each optimization algorithm, a pre-selection with a weighted cost–benefit analysis is processed to choose an optimization algorithm.

Accordingly, the three main criteria for this specific optimization scenario are defined with the weights:

- Convergence ratio (40%);
- Parallel data evaluation (20%);
- Global search strategy (40%).

The evaluation and ranking of all the algorithms are still subjective, but it helps quantify and compare the benefits and drawbacks of all algorithms and frameworks. Appendix A shows the detailed cost–benefit analysis in tabular format. From this analysis, the GOMORS framework is chosen and used for the following analysis due to its potentially high convergence ratio, the parallel data evaluation with up to four parallel calculations, and its global search strategy. Furthermore, GOMORS is already implemented in Python.

Accordingly, the following paragraphs briefly describe the NSGA-II and the GOMORS to understand both methods better.

2.4.1. NSGA-II Algorithm

The Elitist Nondominated Sorting Genetic Algorithm (NSGA-II) is one of the most commonly applied GAs in engineering optimization. The main principle of the algorithm is based on fast, nondominated sorting, with diversity preservation, called crowding distance sorting. Deb described the NSGA-II initially in [18]. The following paragraph gives a brief overview of the articles' implementation.

- Step 0—Define Algorithm inputs: Firstly, the user sets the maximum number of expensive function evaluations, the number of parents, and the mutation rate.
- Step 1—Initial Evaluation Points Selection: The experimental design is processed using expensive simulations for a predefined sampling size.
- Step 2—Iterative Improvement: To select the Pareto points of the sampling and iteratively optimize the designs, the algorithm repeats the steps of fast, nondominated sorting, diversity preservation, crossover, and mutation until it reaches the maximum number of costly evaluations.
    - Step 2.1—Fast, Nondominated Sorting: For each design, a domination count, which represents the number of designs dominating the current design, is set up. Furthermore, a set of designs, dominated by the current design, is assigned. By sorting the designs by their domination count, frontiers are built, for which the most dominating frontiers represent the individuals with the highest potential.
    - Step 2.2a—Density Estimation: The method calculates the crowding distance by the distance of the current solution to its two closest neighbors for each point of each frontier. It then sorts the designs by their crowding distance.
    - Step 2.2b—Crowded Comparison Operator: The method prioritizes designs if they exhibit a prior nondominated rank. Accordingly, if two designs are in the same frontier and have the same nondomination rank, it prefers the design in the less crowded area.
    - Step 2.3—Offspring Creation: The initial population based on the sampling is now sorted according to the mentioned strategies. Then, binary tournament selection, recombination, and mutation create the offspring population. Finally, elitism selects a certain percentage of parents.
    - Step 2.4—Calculate Responses: Expensive simulations return the demanded objective function values.

Finally, the procedure is repeated until the maximum number of expensive evaluations is reached.

The parameter values for the NSGA-II are set by experience. Hence, the number of parents for each generation is set to 20% of the sampling size. A total of 30% of the created offspring are mutated for more extensive exploration. Furthermore, the change of the selected value by mutation is 10%.

2.4.2. GOMORS Framework

Akhtar and Shoemaker [21] implemented the Gap-Optimized Multi-objective Optimization using Response Surfaces (GOMORS) framework. Indeed, GOMORS is a surrogate-assisted algorithm that uses the response surface method to estimate costly objective functions for the optimization with a GA. GOMORS updates the response surface model with the points evaluated in parallel. It identifies new, high-potential points in each iteration for the expensive data evaluation by selection rules based on the surrogate model. The following steps comprehensively describe the algorithm based on the original article.

- Step 0—Define Algorithm inputs: Firstly, the maximum number of expensive function evaluations is selected. Furthermore, the gap parameter and the number of costly evaluations after each iteration are chosen.
- Step 1—Initial Evaluation Points Selection: The experimental design selects the initial set of points and expensively evaluates the selected points via simulation.

- Step 2—Iterative Improvement: The algorithm runs iteratively until the maximum number of expensive function evaluations (abort criteria) is reached.
  - Step 2.1—Fit/Update the Response Surface Models: The response surface models fit each objective for the expensively evaluated simulation points.
  - Step 2.2—Surrogate-Assisted Global Search: GOMORS uses an MOEA to minimize the objective functions represented by the response surface models, selecting a set of potential candidates.
  - Step 2.3a—Identify Least Crowded Solution: The least crowded design is selected by calculating the crowding distance of the expensively computed data.
  - Step 2.3b—Local Search: The framework uses the least crowded solution as input for an MOEA-based neighborhood search, based on the response surface models, called the "Gap Optimization Problem". It processes the search in the radius of the initially defined gap parameter around the least crowded solution.
  - Step 2.4—Select Points for Expensive Function Evaluation: A candidate selection is processed based on selection rules, the data sets from the surrogate-assisted global search, and the local search.
  - Step 2.5—Perform expensive function evaluations and update the non-dominated solution set: The method evaluates candidate points by costly simulations and updates the response surface model with the data of the candidate points.
- Step 3—Return Best Approximated Front: After the final loop, when the maximum number of expensive function evaluations is reached, the best non-dominated frontier is returned.

For further details on GOMORS, the reader is referred to [21]. One problem with the current version of the GOMORS framework is that it is an implementation in Python 2.7, incompatible with Python 3.7. Hence, the GOMORS framework was ported to Python 3.7 to run in the existing IKOS environment. The article provides this modified GOMORS-code via GitHub [37]. Mainly, it consists of the following changes compared to the implementation from [21]:

- It uses a symmetric Latin hypercube sampling instead of a Latin hypercube sampling;
- The MOEA of choice is the epsilon-NSGA-II following Kollat and Reed;
- Instead of the hypervolume improvement, the epsilon-progress-based selection is used.

3. Results

The optimization setup was already established and surveyed in recent papers. The b-pillar has two main load cases: Nastran calculates the linear load case and RADIOSS a non-linear load case. Accordingly, for the linear and the non-linear load case, the b-pillar offers multiple objective functions for evaluation. The potential objective functions are the mass $f_m(x)$, the static deflection (torsion) $f_s(x)$, the intrusion in the lateral direction $f_i(x)$, and the absorbed energy during the impact $f_{ae}(x)$.

The optimization scenario is always multi-objective and can be a multidisciplinary optimization scenario in the case of the deflection and crash case. Figure 3 shows the two load cases.

The Nastran load case is torsional and evaluates the stiffness of the b-pillar. The RADIOSS load case is an impact load case, where an impactor with 15.56 m/s and one ton of mass hits the b-pillar in the lateral direction. Twenty-four DVs can change the b-pillar. Eight DVs each control, the outer sheet metal, the inner sheet metal, and the stiffener. Again, the stiffener is positioned in-between the upper and the lower shell.

The following first two examples compare the performance of the two algorithms NSGA-II and GOMORS for the linear and non-linear load case for a simple optimization scenario without infeasibility. The sampling size is 100 members, while the number of additional evaluations is 100 members. The variation of the b-pillar takes place in the eight DVs of the stiffener.

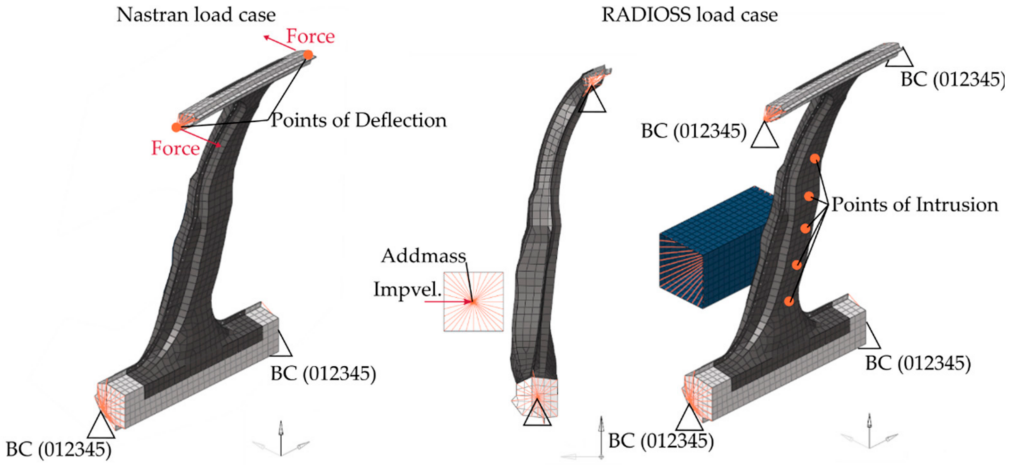

Figure 3. The two load cases (linear and non-linear) of the b-pillar, following [18,20].

The stiffener varies by implicit parametrization between the upper and the lower shell, as shown in Figure 4. Eight sections control the shape and create many possible designs. By setting the DVs implicitly, no infeasible designs are created. Still, the effort it takes to develop such an implicit design is very high.

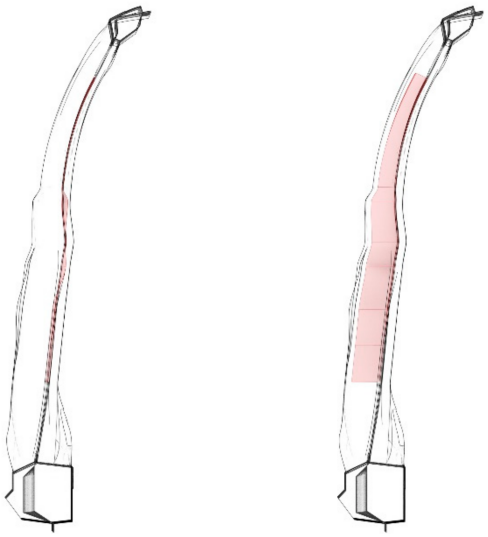

Figure 4. Minimum and maximum configuration of the stiffener sheet metals implicitly parameterized following [8].

3.1. Linear Load Case Comparison without Infeasibility

The first scenario optimizes the b-pillar performance towards the static torsional load case and the mass. The challenge is the minimization of both objective functions. Hence, the unconstrained MOO optimization problem is formulated to:

$$\min_{x} \quad f_m(x), f_s(x) \tag{4}$$

In Figure 5, the optimization results are displayed, which compares the two objective NSGA-II optimization runs with the two objective GOMORS optimization runs for the linear displacement load case. The z-displacement and the mass are normalized to give the viewer a better understanding of the improvement on the scale by the following formula for feature scaling:

$$x'_i = \left(\frac{x_{init} - x_i}{x_{init}} \right) \times 100 \tag{5}$$

where x' is the normalized data for the *ith* data point for the objective function value x_i. All values are relative to the initial design x_{init}. The weight change is relative to the initial part for each design, so it excludes the rest of the structure. In this case, the mass of each design only relates to the initial stiffener design in the first scenario.

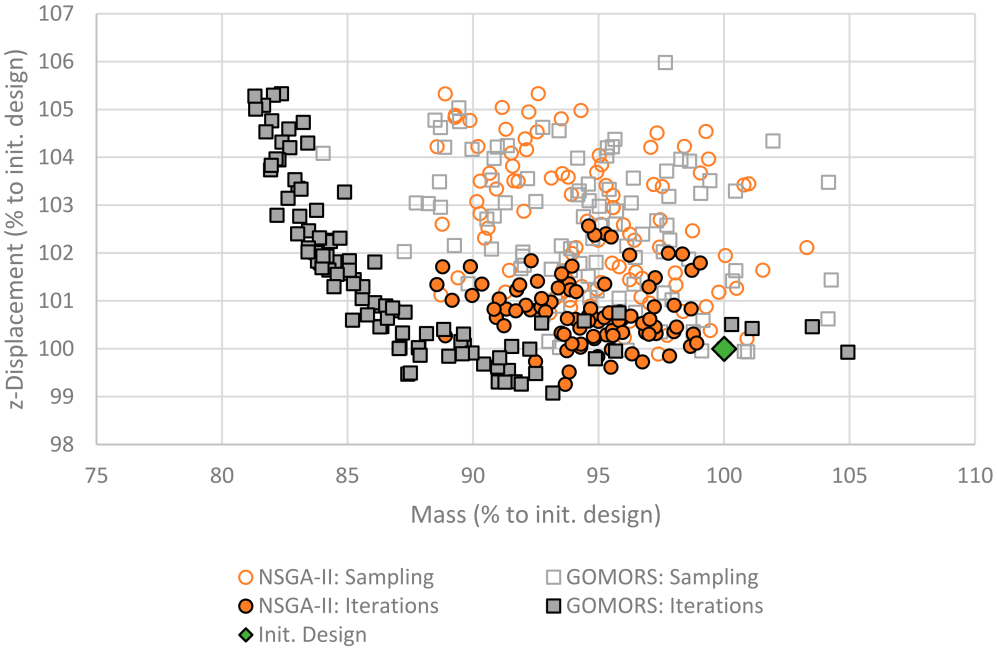

Figure 5. Optimization result of the linear load case without any infeasibility constraint for the NSGA-II and GOMORS. It shows the sampling of both algorithms shaded and the initial design in green. All values are relative to the weight of the initial sheet metal and initial displacement.

One can see that the GOMORS outperforms the NSGA-II. The GOMORS converges faster and reaches a Pareto-frontier earlier. Still, due to the conflicting objective functions, the algorithms strive into the minimum mass, and the structural stiffness decreases. Still, the GOMORS finds designs that reduce structural weight by 13% while maintaining equal stiffness. The shape optimization of the stiffener reduces the mass on the component level by approximately 7%, while the design achieves a minor displacement of 1%. Further structures lie in the lower left quadrant.

3.2. Non-Linear Load Case Comparison without Infeasibility

Figure 3 also shows the second scenario, the intrusion of the impactor in the b-pillar. In this scenario, the algorithms minimize the intrusion and the mass. Again, the min–max feature scaling method is used for the value scaling in Figure 6.

Again, the GOMORS converges significantly faster than the NSGA-II. The NSGA-II slightly improves the design compared to the sampling points, as multiple iterations lie in the sampling area. The GOMORS, on the other hand, has a more explorative behavior. The designs are determined to converge towards the Pareto-frontier. Nearly no points are in the sampling area. The GOMORS can decrease the weight by 12.5% while maintaining equal intrusion. The NSGA-II can only reduce the weight by 8% under similar intrusion. Significant decreases in the intrusion up to 12% are possible, reducing weight by 5%.

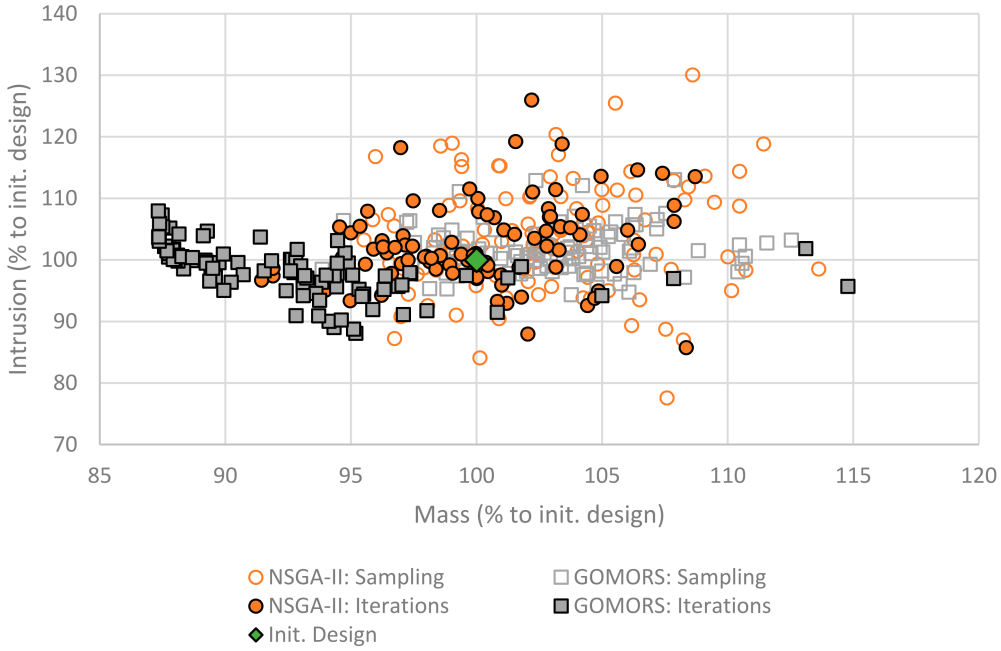

Figure 6. Optimization result of the non-linear load case without any infeasibility constraint for the NSGA-II and GOMORS. The figure shows the sampling of both algorithms shaded and the initial design in green. All values are relative to the initial sheet metal's weight and the initial setup's intrusion.

The preceding paragraphs demonstrate the performance of the GOMORS in comparison to the NSGA-II for linear and non-linear calculations. Hence, further analysis will only assess linear calculations due to the high computational resources needed for non-linear calculations.

3.3. Optimization Results for the Linear Load Case *with Infeasibility*

The third analysis compares the two algorithms for the same b-pillar with two more design variables and the infeasibility constraint. Figure 7 shows three different configurations of the b-pillar in its thinnest, the initial, and the thickest configuration. The used design and package-constraint surfaces are the same as in [19]. The stiffener can still vary between the upper and the lower shell. Furthermore, the design space, representing the external geometric constraint from package and design, is highlighted in red.

Another problem is the ratio of infeasible designs by the intersecting hull surface, which reduces the number of calculated designs [18,20]. Accordingly, the sampling size of actual calculated samples shrinks by a factor of 0.2 for the given sample. The sampling size is initially set to 1000 members to compensate for that effect. Hence, approximately 200 members are calculated during the optimization; still, the number of design variables is increased by two, which increases the size of the solution space. The number of iterative samples is set to 200. Finally, Figure 8. shows the resulting point diagram with the normalized mass and z-displacement.

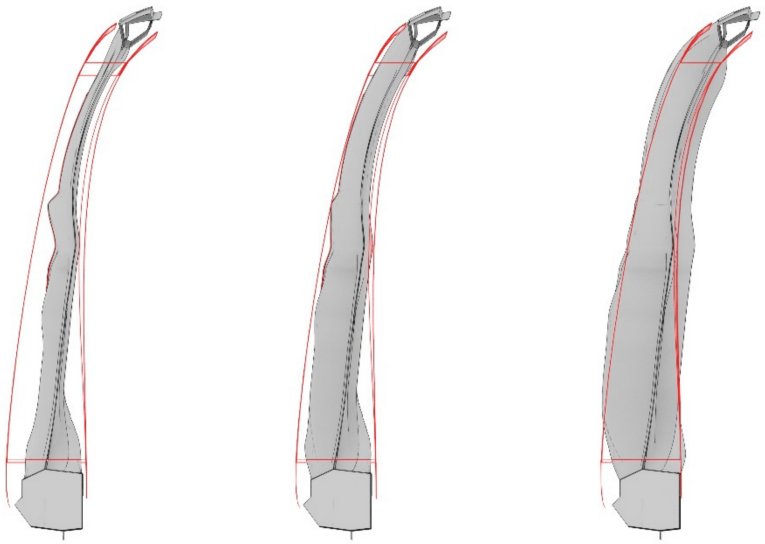

Figure 7. The b-pillar for the third scenario. The red line represents the geometric hull surface. The stiffener can be changed the same way as in the first scenario. Additionally, the thickness of the b-pillar is variable by the upper and lower shell.

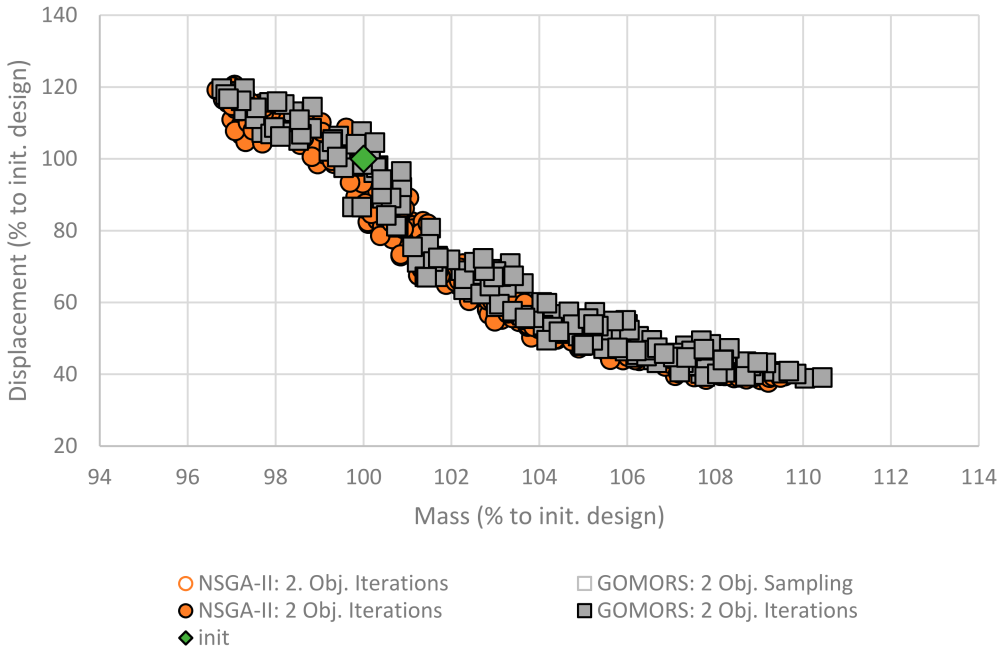

Figure 8. Optimization result of the linear load case with infeasibility constraint for the NSGA-II and GOMORS. The figure shows the sampling of both algorithms shaded and the initial design in green. All values are relative to the initial sheet metal's weight and the initial setup's intrusion.

The distribution of the samples differs due to the different model setups. Conversely to the previous optimization runs, the optimization algorithms perform very differently. The NSGA-II and the GOMORS act pretty similar for the shape optimization task. Both form an interval of potential candidates but do not converge into a region of ideal design. For equal mass, both algorithms find solutions, which decrease the displacement by 20%. For a similar displacement, the max decrease is about 1.5%. Notably, the NSGA-II shows a slightly better performance than the GOMORS, which the small sampling size could cause. Both algorithms mostly create feasible designs considering the infeasibility constraint. The GOMORS creates three infeasible members, and the NSGA-II creates four infeasible

members. Still, both results seem to be insufficient in the improvement of performance. Hence, a different approach is needed, discussed in the following section.

3.4. Implementation for Surrogate-Based Optimization Schemes

The preceding Sections 3.1 and 3.2 demonstrate that the surrogate-based GOMORS framework mostly has a superior performance to the NSGA-II in the case of classical shape optimization. However, penalty factors may cause surrogate models to map the behavior of the actual physical model poorly. This behavior was already predicted in [20] and is now validated in Section 3.3 of this article. Hence, the following section tests a second approach to validate the feasibility of the penalization scheme for surrogate-based processes, which are represented in this case by GOMORS.

As the unsteady behavior of the penalization scheme may cause an inferior quality of the surrogate models [20], the penalty is therefore set as the third objective function. The approach benefits from the lower amount of needed mesh evaluations, as it does not manipulate the infeasible designs. Still, the number of required calculations increased. The objective function value of the infeasibility factor $f(x)_{feas}$ is zero if no infeasibility is detected, and $f(x)_{feas}$ is larger than zero if a surface penatration is detected. As GOMORS performed well on the unconstrained and provided worse results in the constrained approach, the approach is only validated for GOMORS under consideration of the following three objective functions: mass $f_m(x)$, stiffness $f_s(x)$ and feasibility $f_{feas}(x)$:

$$\min_x \quad f_m(x), f_s(x), f_{feas}(x) \tag{6}$$

With respect to the preceding optimizations, the sampling size is set to 200, and the number of iterations is set to 100. Figure 9 shows the Pareto-frontier, which is built during the optimization. The normalization is relative concerning the initial design, and it considers the outer sheet metal, the inner sheet metal, and the stiffener. Hence, the percentage relations are the same as in Figure 9.

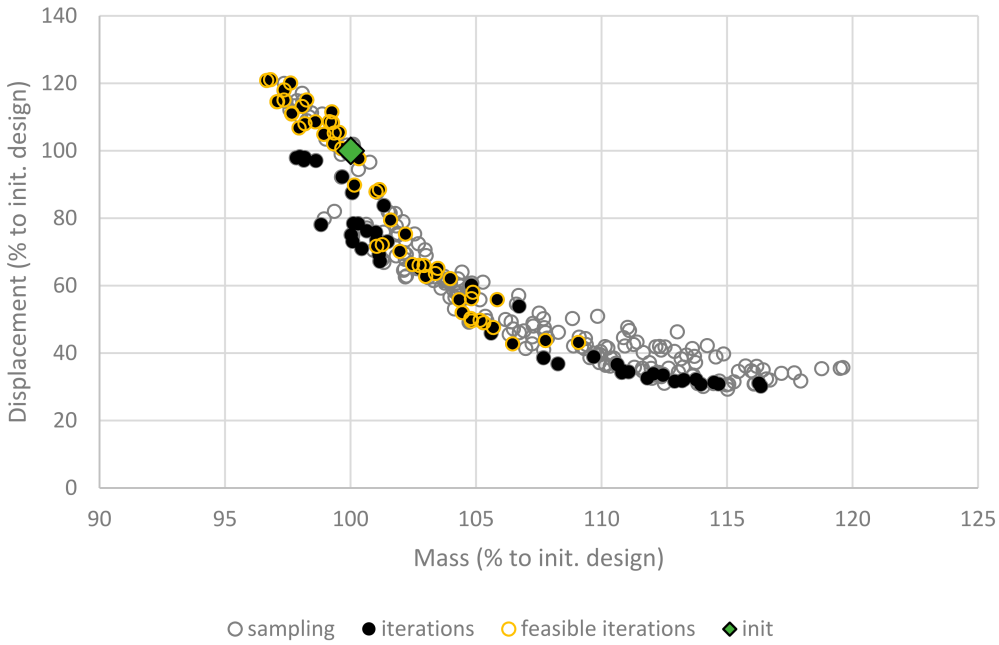

Figure 9. The Pareto-frontier for the GOMORS optimization concerning three objective functions: mass, displacement, and feasibility. All objectives are relative to the initial design.

With the presented optimization setup, only a feasible design outperforms the initial setup in terms of Pareto-optimality. Still, the improvement is minimal, which means no increase in terms of mass and ~10% in displacement. Conversely, there are potential lighter

candidates, but they exhibit a more significant static deformation. This lack of potential candidates may result from the larger solution space (10 DV) and the strong influence of the outer and inner shell movement on the weight and resulting displacement due to the feature importance.

The sampling size is increased to 1000 samples to improve the surrogate model quality, while 200 iterations allow a more detailed optimization run. Furthermore, the number of design variables is increased to 24. The 16 DVs for the outer and inner shells are activated to ensure that the influence of the DVs is evenly weighted and that more design freedom leads to candidates of higher potential. Furthermore, the feasibility approach with penalty constraint and the feasibility approach with three objective functions are directly compared for the chosen sampling and the NSGA-II and the GOMORS iterations, as shown in Table 1.

Table 1. The table shows the setup of the different optimization runs for the GOMORS and the NSGA-II.

Algorithm	Sampling Size	Nr. Iterations	Objective Functions	DVs
GOMORS	1000	1000	2	24
NSGA-II	1000	1000	2	24
GOMORS	1000	200	3	24

According to Table 1, the three scenarios are optimized and visualized in the following paragraphs. Again, the b-pillar is analyzed, but the number of DVs is increased to 24. The optimization problem can be formulated as:

$$\begin{aligned} &\min_{x} f_m(x),\ f_s(x) \\ &subject\ to\ f_{feas}(x) = 0 \end{aligned} \tag{7}$$

For a better understanding of the outer shell movement, four possible feasible candidates are shown in Figure 10. The stiffener can move in between the upper and lower shell, as presented in Figure 4.

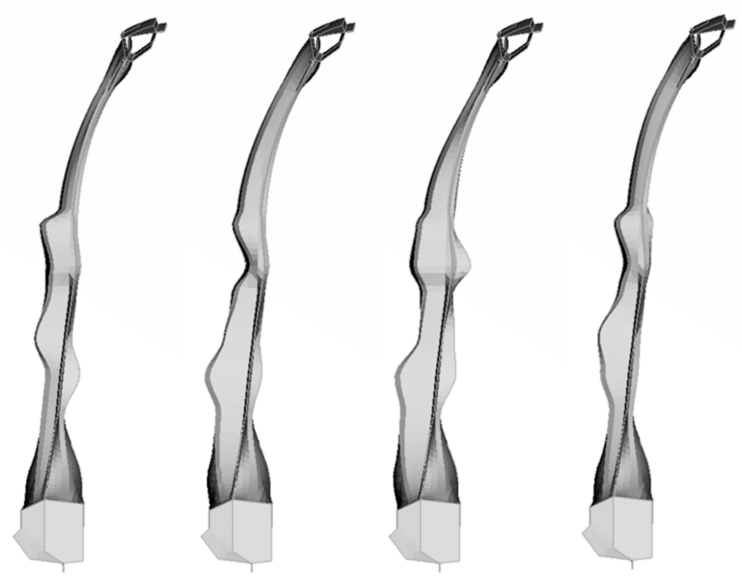

Figure 10. Four possible configurations of the b-pillar controlled by the 16 DVs of the outer shell. The stiffener is governed by the 8 DVs used in the previous optimizations.

Figure 11 displays the overall performance of the GOMORS three objective implementation. It plots the third objective function for feasible results beside the performance of the

objective functions mass and displacement. Hence, the yellow circles represent the feasible results of the optimization run.

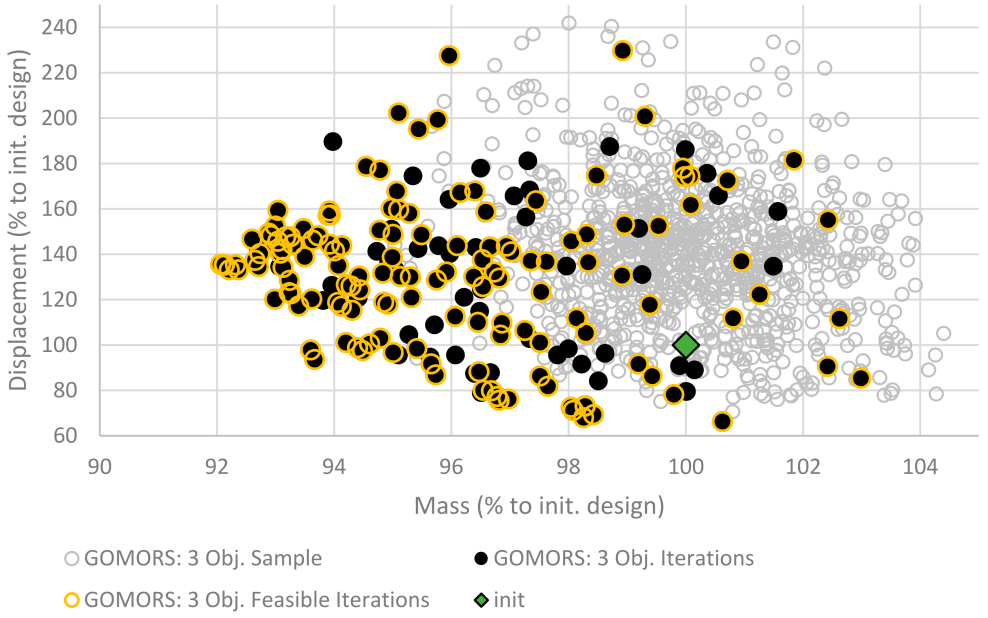

Figure 11. The Pareto-frontier for the GOMORS optimization concerning three objective functions (mass, displacement, and feasibility) plotted over mass and displacement. The yellow dots represent the feasible iterations.

The figure shows that the mass and displacement can be decreased significantly, while mostly feasible designs are created. The decrease in mass goes up to 8% for an equal static deformation, while other candidates exhibit a reduced mass by 5.5% and nearly 10% lower deformation. The sampling contains almost no feasible designs, but the algorithm produces mostly feasible designs during the optimization.

Figure 12 shows the convergence of the feasibility for the three other optimization runs to demonstrate the effectiveness of the different approaches towards the feasibility. Therefore, it offers the infeasible designs per 10 samples for the sampling and optimization iterations. The sampling size, as described in Table 1, is 1000.

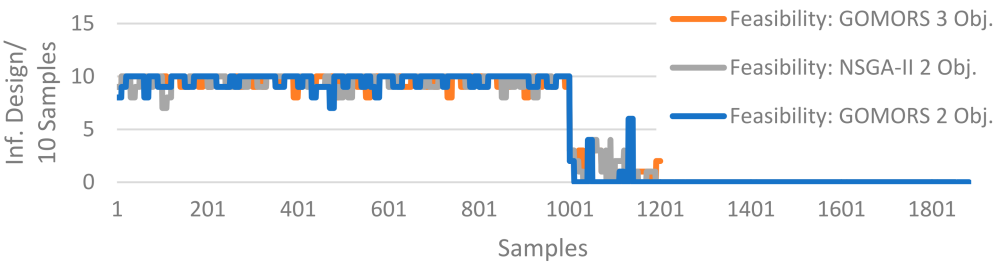

Figure 12. Feasible designs per 10 samples plotted over the sampling and the iterations of the optimization algorithms.

Figure 12 shows that infeasible designs are very high in the sampling. After the sampling, the number of infeasible designs decreases to zero in the case of the constrained two objective approach. Conversely, the three-objective approach still creates some infeasible designs. To visualize the convergence of the algorithms, Figure 13 shows how the NSGA-II and the GOMORS perform for the given optimization with 24 DVs for the two-objective constraint approach and the three-objective approach. In the two-objective approach, the infeasible designs are penalized and not part of the frontier, while in the

three-objective approach, the final designs are all created feasible. For each algorithm, the frontier only represents feasible designs.

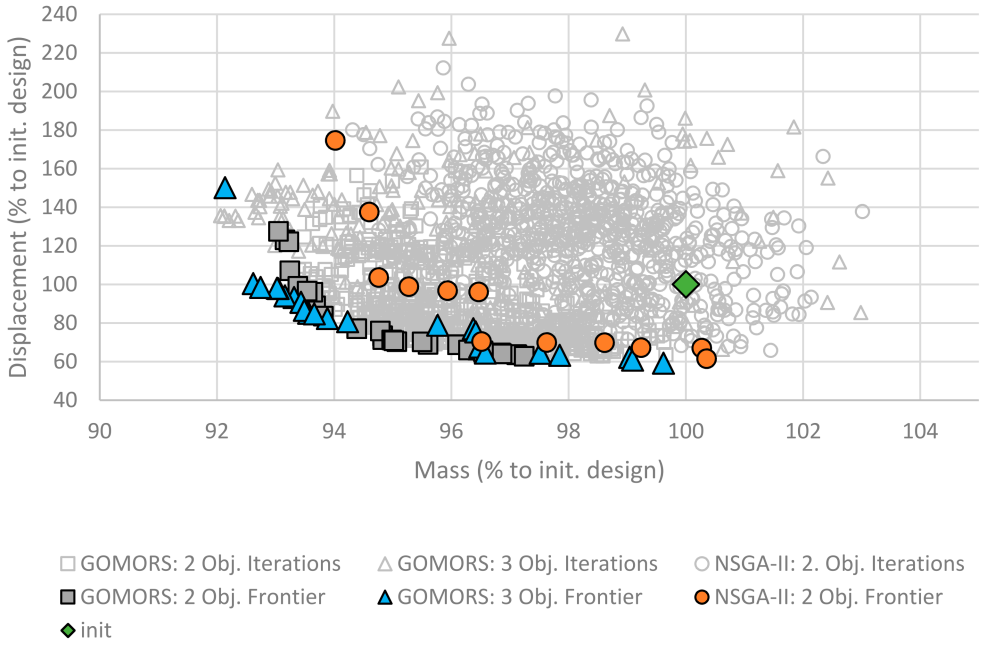

Figure 13. The Pareto-frontier for the GOMORS with two and three objective functions and the NSGA-II with two objective functions. The performance is relative to the initial design.

One can see that the GOMORS outperforms the NSGA-II. Both GOMORS frontiers nearly converge equally, while the NSGA-II frontier does not converge this far. For an equal displacement of 100%, the GOMORS with three objective functions reduces the mass to 92.5%, while the GOMORS with two objective functions reduces the mass to 93.5%. The NSGA-II, on the other hand, only achieves 95% of the initial mass.

4. Discussion

The presented analysis dealt with implementing the GOMORS optimization algorithm into an existing framework and its potential to optimize the structural performance of BIW parts under a geometrical constraint. The results of the structural optimization for a linear and a non-linear crash-load case are promising, as the GOMORS outperforms the NSGA-II in terms of convergence ratio. Figures 4 and 5 show that GOMORS reduces the time for calculation and the number of expensive computations. The second part of the paper dealt with whether surrogate-based algorithms can handle the feasibility constraint. This question was successfully answered with Figure 7. The surrogate represents the model behavior properly so that the optimization converges quickly if the sampling size is of sufficient size. In [20], the authors tested Random Forrest Regression as a surrogate model and found that the penalized infeasible sampling's mean squared error was very high. Concerning the optimization of the linear load case with infeasibility constraint, this hypothesis can be partly refuted, as the GOMORS converges much better than the NSGA-II, which could be reasoned by a higher performance of the RBF surrogate model.

The final test of the GOMORS framework for three-objective optimization analyzes the performance of the constraining approach by direct penalization of infeasible designs. It considers the displacement, mass, and feasibility. Firstly, Figure 11 shows the functionality of the approach as the final frontier only exhibits feasible results, which also show better structural performance than the initial design. Figure 13 compares the performance of the two-objective with the three-objective approach. While comparing the two-objective to the three-objective GOMORS approach, the three-objective approach performs slightly better in this specific scenario. Furthermore, GOMORS outperforms the NSGA-II for the

24 DV example. The results demonstrate the performance of the GOMORS framework in comparison to the NSGA-II. Still, one drawback of the GOMORS implementation is the number of parallel evaluations, which is limited to four, and for the current implementation, to one. Long calculation runs may benefit a substantial parallelization; a wider parallelization will make the GOMORS even more performant. It could help to reduce the time of optimization even further. Even if the convergence ratio of the NSGA-II is lower than the one of GOMORS, the nearly unlimited scalable parallelization makes it attractive for cheaper optimization tasks, such as linear calculations. Still, GOMORS may be of benefit here, as areas that tend to be not explored by the NSGA-II are explored.

5. Conclusions and Outlook

The multi-objective optimization of structural BIW parts, especially with costly crash calculation, is challenging for standard metaheuristic algorithms, such as the NSGA-II. Especially when the design is constrained, as in the demonstrated geometric constraint, the NSGA-II tends to converge slowly and may use many costly evaluations. Implementing the GOMORS optimization framework into the IKOS framework showed that surrogate-based frameworks can outperform typical metaheuristic approaches on a large scale, even concerning constrained objective functions and linear and non-linear calculations. Here, the method of infeasibility penalization was validated for the GOMORS algorithm and can now be effectively used for the presented scenarios. Furthermore, the article demonstrated the importance of choosing influential DVs as a critical factor for the success of optimizations. The study confirmed that more DVs enable larger improvements in the initial design proposal, e.g., 7.5% less weight while maintaining equal structural stiffness without any need for different tooling or manufacturing technologies.

Further research will focus on analyzing the optimization parameter of the GOMORS framework, for example, the influence of the gap and additional internal parameters, which have been defaulted in this analysis. Furthermore, the optimization of more complex non-linear calculations will be of significant interest, mainly non-linear crash calculations of longitudinal structures. Problems such as bifurcation and high uncertainties occur during the non-linear calculation for these calculations. This behavior may be a problem for the radial basis function, failing to handle the significant uncertainties. Here, different surrogate approaches may be of major interest. In particular, the question should be answered if the surrogate-assisted process has major benefits compared to classic metaheuristics such as the NSGA-II [7,19].

At the same time, further applications of the feasibility factor are of significant interest. The approach could be tested in all domains, where shape optimization is used under geometric constraints, such as structural applications, e.g., automotive chassis components [38] or aeroelastic analysis for planes [39]. Primary applications tend to be aerodynamics in planes [40] or preliminary race car design, where large geometrical designs are explored under strictly regulative design spaces [41].

Author Contributions: Conceptualization, Y.W.; methodology, Y.W.; software, Y.W., V.S.R.G. and T.v.H.; validation, Y.W. and T.v.H.; formal analysis, T.v.H. and V.S.R.G.; investigation, T.v.H. and V.S.R.G.; resources, T.V.; writing—original draft preparation, Y.W.; writing—review and editing, Y.W.; visualization, Y.W.; supervision, Y.W.; project administration, T.V. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge support from the Open Access Publication Funds of Technische Universität Braunschweig.

Data Availability Statement: The implemented GOMORS for Python 3.7 can be downloaded via the following link on GitHub [37].

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table 1. Choice of the Optimization Algorithm.

Algorithm/Framework		**Ant Colony Optimization (ACO)**	**Bacterial Foraging Optimization (BFO)**	**Multi-Objective Bees Algorithm (Bees)**	**Cooperative Bacterial Foraging Algorithm (CBFO)**	**Continuous Genetic Algorithm (CGA)**	**Differential Evolution for Multi-Objective Optimization (DEMO)**	**The Expected Improvement in Pareto Hypervolume (EHI)**	**Gap-Optimized Multi-Objective Optimization Using Response Surface (GOMORS)**	**Max-Value Entropy Search for Multi-Objective Optimization (MESMO)**
Approach		Metaheuristic	Metaheuristic	Metaheuristic	Metaheuristic	Metaheuristic	Metaheuristic	Bayes Approach	Metaheuristic	Bayes-Approach
Source		[3,8]	[5]	[6–8]	[3]	[3]	[6–8]	[2]	[1]	[2]
Criteria										
Convergence ratio * (1)		-	-	-	-	-	o	+ * (2)	+ * (2), (3)	++ * (2), (4)
Parallelization		?	++	?	?	++	?	–	++	–
Global search strategy		+	+	+	+	+	?	++	++	++
weighted	weight (%)									
Convergence ratio * (1)	40	0.1	0.1	0.1	0.1	0.1	0.2	0.3	0.3	0.4
Parallelization	20	?	0.2	?	?	0.2	?	0	0.2	0
Global search str.	40	0.3	0.3	0.3	0.3	0.3	?	0.4	0.4	0.4
Sum	100	-	0.6	-	-	0.6	-	0.7	0.9	0.8

* (1) Time/amount of function calls/evaluations until sufficient convergence
* (2) Update of the surrogate model necessary
* (3) In comparison to the NSGA-II and parEGO, most efficient
* (4) Faster than the PESMO because of the "input scape entropy-based" approach
* (5) Enhancements include vectorization and constraint handling
* (6) By vectorization of the NSGA-II 8x faster convergence to a similar frontier in comparison to the MOEA/D
* (7) Performance is fluctuating
* (8) Problematic due to premature convergence

[1] Akhtar & Shoemaker (2016)
[2] Belakaria et al. (2019)
[3] Georgiou et al. (2014)
[4] Paas & van Dijk (2017)
[5] Wang & Cai (2018)
[6] Yang & Deb (2013)
[7] Yang (2013)
[8] Yang (2014)
[9] Zhao et al. (2016)

Legend: ++ strongly agree; + agree; o neutral; - disagree; – strongly disagree; ? unknown.

Table 2. Choice of the Optimization Algorithm.

Algorithm/Framework		Multi-Objective Cuckoo Search (MOCS)	Multi-Objective Differential Evolution (MODE)	Multi-Objective Evolutionary Algorithm-Based on Decomposition (MOEA/D)	Multi-Objective Firefly Algorithm (MOFA)	Multi-Objective Flower Pollination Algorithm (MOFPA)	Non-Dominated Sorting Based Multi-Objective Evolutionary Algorithm (NSGA-II)	Enhanced Nondominated Sorting Based Multi-Objective Evolutionary Algorithm (Enhanced NSGA-II)	Pareto-Efficient Global Optimization (ParEGO)	Predictive Entropy Search for Multi-Objective Bayesian Optimization (PESMO)
Approach		Metaheuristic	Metaheuristic	Metaheuristic	Metaheuristic	Metaheuristic	Metaheuristic	Metaheuristic	Bayes Approach	Bayes Approach
Source		[6,7]	[6–8]	[4]	[7,8]	[8]	[1,6–8]	[4]	[1,2]	[2]
Criteria										
Convergence ratio * (1)		+	o	-	+	+	-	o * (6)	o * (2), (7)	+ * (2)
Parallelization		++	?	++	++	++	++	++	–	–
Global search strategy		+	?	o	+	+	+	+	++	++
weighted	weight (%)									
Convergence ratio * (1)	40	0.3	0.2	0.1	0.3	0.3	0.1	0.2	0.2	0.3
Parallelization	20	0.2	?	0.2	0.2	0.2	0.2	0.2	0	0
Global search str.	40	0.3	?	0.2	0.3	0.3	0.3	0.3	0.4	0.4
Sum	100	0.8	-	0.5	0.8	0.8	0.6	0.7	0.6	0.7

* (1) Time/amount of function calls/evaluations until sufficient convergence
* (2) Update of the surrogate model necessary
* (3) In comparison to the NSGA-II and parEGO, most efficient
* (4) Faster than the PESMO because of the "input scape entropy-based" approach
* (5) Enhancements include vectorization and constraint handling
* (6) By vectorization of the NSGA-II 8x faster convergence to a similar frontier in comparison to the MOEA/D
* (7) Performance is fluctuating
* (8) Problematic due to premature convergence

[1] Akhtar & Shoemaker (2016)
[2] Belakaria et al. (2019)
[3] Georgiou et al. (2014)
[4] Paas & van Dijk (2017)
[5] Wang & Cai (2018)
[6] Yang & Deb (2013)
[7] Yang (2013)
[8] Yang (2014)
[9] Zhao et al. (2016)

Legend: ++ strongly agree; + agree; o neutral; - disagree; – strongly disagree; ? unknown.

Table 3. Choice of the Optimization Algorithm.

Algorithm/Framework		Particle Swarm Optimization (PSO)	Hybrid Particle Swarm Optimization Incl. Bacterial Foraging Optimization (PSO-BFO)	Hybrid Particle Swarm Optimization Incl. Genetic Algorithm (PSO-GA)	S-Metric Section-Based Efficient Global Optimization (SMSego)	Strength Pareto Evolutionary Algorithm (SPEA)	Probability of Improvement in Stepwise Uncertainty Reduction (SUR)	Vector Evaluated Genetic Algorithm (VEGA)
Approach		Methaheuristic	Methaheuristic	Methaheuristic	Methaheuristic	Methaheuristic	Methaheuristic	Methaheuristic
Source		[3,5,8]	[5,9]	[5]	[2]	[6–8]	[2]	[6–8]
Criteria								
Convergence ratio * (1)		-	+	+	+ * (2)	-	+ * (2)	-
Parallelization		++	++	++	‘–	?	‘–	?
Global search strategy		- * (8)	+	+	++	?	++	?
weighted	weight (%)							
Convergence ratio * (1)	40	0.1	0.3	0.3	0.3	0.1	0.3	0.1
Parallelization	20	0.2	0.2	0.2	0.0	?	0.0	?
Global search str.	40	0.1	0.3	0.3	0.4	?	0.4	?
Sum	100	0.4	0.8	0.8	0.7	-	0.7	-

* (1) Time/amount of function calls/evaluations until sufficient convergence
* (2) Update of the surrogate model necessary
* (3) In comparison to the NSGA-II and parEGO, most efficient
* (4) Faster than the PESMO because of the "input scape entropy-based" approach
* (5) Enhancements include vectorization and constraint handling
* (6) By vectorization of the NSGA-II 8x faster convergence to a similar frontier in comparison to the MOEA/D
* (7) Performance is fluctuating
* (8) Problematic due to premature convergence

[1] Akhtar & Shoemaker (2016)
[2] Belakaria et al. (2019)
[3] Georgiou et al. (2014)
[4] Paas & van Dijk (2017)
[5] Wang & Cai (2018)
[6] Yang & Deb (2013)
[7] Yang (2013)
[8] Yang (2014)
[9] Zhao et al. (2016)

Legend: ++ strongly agree; + agree; o neutral; - disagree; – strongly disagree; ? unknown.

References

1. Feldhusen, J.; Grote, K.-H. *Pahl/Beitz Konstruktionslehre*; Methoden und Anwendung Erfolgreicher Produktentwicklung; Springer: Berlin/Heidelberg, Germany, 2013.
2. Schumacher, A.; Seibel, M.; Zimmer, H.; Schäfer, M. New optimization strategies for crash design. In Proceedings of the 4th LS-DYNA Anwenderforum, Bamberg, Germany, 20–21 October 2005.
3. Bletzinger, K.-U. Shape Optimization. In *Encyclopedia of Computational Mechanics*, 2nd ed.; Stein, E., de Borst, R., Hughes, T.J.R., Eds.; John Wiley & Sons: Chichester, UK, 2018; pp. 1–42.
4. Zimmer, H. Erweiterte Knotenfunktionalität im parametrischen Entwurfswerkzeug SFE Concept. *FAT* **2002**, *Nr.172*, 1–30.
5. Paas, M.H.J.W.; van Dijk, H.C. *Multidisciplinary Design Optimization of Body Exterior Structures*; Springer: Cham, Switzerland, 2017; Volume 41, pp. 17–30.
6. Schumacher, A.; Vietor, T.; Fiebig, S.; Bletzinger, K.-U.; Maute, K. *Advances in Structural and Multidisciplinary Optimization*; Springer International Publishing: Cham, Switzerland, 2018.
7. Duddeck, F. Multidisciplinary optimization of car bodies. *Struct. Multidiscip. Optim.* **2008**, *35*, 375–389. [CrossRef]
8. Rayamajhi, M.; Hunkeler, S.; Duddeck, F. Geometrical compatibility in structural shape optimisation for crashworthiness. *Int. J. Crashworthiness* **2013**, *19*, 42–56. [CrossRef]
9. Ryberg, A.-B.; Domeij Bäckryd, R.; Nilsson, L. *Metamodel-Based Multidisciplinary Design Optimization for Automotive Applications*; Linköping University Electronic Press: Linköping, Sweden, 2012.
10. Younis, A.; Dong, Z. Trends, features, and tests of common and recently introduced global optimization methods. *Eng. Optim.* **2010**, *42*, 691–718. [CrossRef]
11. Palar, P.S.; Liem, R.P.; Zuhal, L.R.; Shimoyama, K. On the use of surrogate models in engineering design optimization and exploration. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, New York, NY, USA, 13–17 July 2019; pp. 1592–1602.
12. Vahid, G. *Adaptive Search Approach in Multidisciplinary Optimization of Lightweight Structures Using Hybrid-Metaheuristics*; Technische Universität Braunschweig: Braunschweig, Germany, 2020.
13. Rayamajhi, M.; Hunkeler, S.; Duddeck, F.; Zarroug, M.; Rota, L. Robust Shape Optimization for Crashworthiness via a Sub-structuring Approach. In Proceedings of the 9th ASMO UK/ISSMO Conference on Engineering Design Optimization, Product and Process Improvement, Cork, Ireland, 5–6 July 2012.
14. Rayamajhi, M. *Efficient Methods for Robust Shape Optimisation for Crashworthiness*; Technische Universität München: London, UK, 2014.
15. Bäckryd, R.; Ryberg, A.-B.; Nilsson, L. Multidisciplinary design optimisation methods for automotive structures. *Int. J. Automot. Mech. Eng.* **2017**, *14*, 4050–4067. [CrossRef]
16. Yeniay, Ö. Penalty Function Methods for Constrained Optimization with Genetic Algorithms. *Math. Comput. Appl.* **2005**, *10*, 45–56. [CrossRef]
17. Malen, D.E. *Fundamentals of Automobile Body Structure Design*; SAE International: Warrendale, PA, USA, 2020.
18. Werner, Y.; Vietor, T.; Weinert, M.; Erber, T. Multidisciplinary design optimization of a generic b-pillar under package and design constraints. *Eng. Optim.* **2021**, *53*, 1884–1901. [CrossRef]
19. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Computat.* **2002**, *6*, 182–197. [CrossRef]
20. Werner, Y.; Thiele, P.; Gopalan, V.S.R.; Vietor, T. From package and design surfaces to optimization—How to apply shape optimization under geometrical constraints. *Procedia CIRP* **2021**, *100*, 548–553. [CrossRef]
21. Akhtar, T.; Shoemaker, C.A. Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection. *J. Glob. Optim.* **2016**, *64*, 17–32. [CrossRef]
22. Duddeck, F.; Zimmer, H. New Achievements on Implicit Parameterization Techniques for Combined Shape and Topology Optimization for Crashworthiness based on SFE CONCEPT. In Proceedings of the Shape and Technology Optimization for Crashworthiness, Int. Crashworthiness Conf. ICRASH2012, Milano, Italy, 18–20 July 2012; pp. 1–14.
23. Ghaffarimejlej, V.; Türck, E.; Vietor, T. Finding the best material combinations through multi-material joining, using genetic algorithm. In Proceedings of the European Conference on Composite Materials (ECCM 2016), Munich, Germany, 26–30 June 2016.
24. Rayamajhi, M.; Hunkeler, S.; Duddeck, F. Efficient Robust Shape Optimization for Crashworthiness. In Proceedings of the 10th World Congress on Structural and Multidisciplinary Optimization, Orlando, FL, USA, 19–24 May 2013.
25. Hillmann, J. On the Development of a Process Chain for Structural Optimization in Vehicle Passive Safety. Ph.D. Thesis, Technische Universität Berlin, Berlin, Germany, 2009. [CrossRef]
26. Schmitt, B.I. Konvergenzanalyse für die Partikelschwarmoptimierung. In *Ausgezeichnete Informatikdissertationen 2015*; Gesellschaft für Informatik: Bonn, Germany, 2015.
27. Yang, X.-S.; Deb, S.; Fong, S. Metaheuristic Algorithms: Optimal Balance of Intensification and Diversification. *Appl. Math. Inf. Sci.* **2014**, *8*, 977–983. [CrossRef]
28. Georgiou, G.; Vio, G.A.; Cooper, J.E. Aeroelastic tailoring and scaling using Bacterial Foraging Optimisation. *Struct. Multidiscip. Optim.* **2014**, *50*, 81–99. [CrossRef]
29. Yang, X.-S.; Deb, S.; He, X. Eagle Strategy with Flower Algorithm. In Proceedings of the 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, India, 22–25 August 2013; pp. 1213–1217.

30. Deb, K. An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving problems with Box Constraints. *IEEE Trans. Evol. Comput.* **2013**, *18*, 577–601. [CrossRef]
31. Wang, D.; Cai, K. Multi-objective crashworthiness optimization of vehicle body using particle swarm algorithm coupled with bacterial foraging algorithm. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2018**, *232*, 1003–1018. [CrossRef]
32. Haber, R.E.; Beruvides, G.; Quiza, R.; Hernandez, A. A Simple Multi-Objective Optimization Based on the Cross-Entropy Method. *IEEE Access* **2017**, *5*, 22272–22281. [CrossRef]
33. Belakaria, S.; Deshwal, A.; Doppa, J.R. Max-value Entropy Search for Multi-Objective Bayesian Optimization. In Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019.
34. Knowles, J. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 50–66. [CrossRef]
35. Chugh, T.; Sindhya, K.; Hakanen, J.; Miettinen, K. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Comput.* **2019**, *23*, 3137–3166. [CrossRef]
36. Yang, X.-S. *Nature-Inspired Optimization Algorithms*; Elsevier: Amsterdam, The Netherlands, 2014.
37. Raja Gopalan, V.S.; Werner, Y.; van Hout, T. GOMORS Implementation in Python 3.7 Using Pysot Package. Available online: https://github.com/Vijey-Subramani-Raja-Gopalan/GOMORS_Python3.7_PYSOT0.2.0/tree/v1.0.1 (accessed on 27 January 2021).
38. Georgios, K. Shape and parameter optimization with ANSA and LS-OPT using a new flexible interface. In Proceedings of the 6th European LS-DYNA Conference, Gothenburg, Sweden, 28–30 May 2007.
39. Cavagna, L.; Ricci, S.; Riccobene, L. A Fast Tool for Structural Sizing, Aeroelastic Analysis and Optimization in Aircraft Conceptual Design. In Proceedings of the 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Palm Springs, CA, USA, 4–7 May 2009; p. 05042009.
40. Skinner, S.; Zare-Behtash, H. State-of-the-art in aerodynamic shape optimisation methods. *Appl. Soft Comput.* **2018**, *62*, 933–962. [CrossRef]
41. Katz, J.J. *Race-Car Aerodynamics*; McGraw-Hill Professional: Cambridge, MA, USA, 2015.