Technical University of Munich
TUM School of Engineering and Design

Doctoral Thesis

Computational Knowledge Generation for Cooperative Set-based Concept Optimization

Nidhi Nivesh Varma Dommaraju

Technische Universität München
TUM School of Engineering and Design

Computational Knowledge Generation for Cooperative Set-based Concept Optimization

Nidhi Nivesh Varma Dommaraju

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr.-Ing. Kai-Uwe Bletzinger

Prüfer*innen der Dissertation:

1. Prof. Dr.-Ing. habil. Fabian Duddeck
2. Priv.-Doz. Dr.-Ing. habil. Stefan Kollmannsberger
3. Assoc. Prof. Dr. Jun Wu

Die Dissertation wurde am 14.06.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 06.12.2023 angenommen.

Abstract

Computer-aided engineering has greatly supported the product development cycle through simulations and computational optimization of products according to design requirements. In the early stages of the design process, topology optimization methods are widely used to generate novel structural concepts by optimizing the material layout in a fixed domain for a given objective, such as structural compliance or crash energy absorption, subject to given constraints and boundary conditions. Multi-objective optimization methods can further help incorporate multiple objectives across disciplines and yield a large number of potentially useful solutions, which challenges the identification of promising concepts for further analysis and development. While knowledge generation methods such as clustering enable the selection of designs based on experience, manufacturing cost, or other performance attributes, more sophisticated metrics for distinguishing designs are needed for complex engineering applications. Furthermore, topology optimization requires expensive simulations to find solutions in a high-dimensional decision space. So, it is economical to identify regions of interest and generate only the preferred solutions; this approach is used by the so-called *interactive methods* in multi-objective optimization. In this thesis, we adapt this approach for multi-objective topology optimization and develop a novel interactive framework that uses a reference input of solutions to generate the desired solutions. Existing clustering methods are used to conveniently select desired solutions. Since the topology of structures is important in structural and crash mechanics because of performance and aesthetics, we also investigate the use of state-of-the-art deep learning methods such as autoencoders in analyzing complex topologies. For this purpose, we introduce a novel comparison method for metrics of geometry and evaluate them using synthetic datasets representative of optimized structures. Given a reference set of solutions, identified for example using clustering methods with a suitable metric, our main objective is to develop an interactive method to generate designs similar to the reference set using multi-objective topology optimization algorithms, where state-of-the-art methods specify solution preference using weights for the objectives. To avoid unnecessary computations, metamodels are used to predict if a given set of weights will result in the preferred solution. We demonstrate our cooperative framework using a cantilever multi-load-case problem, a crashworthiness optimization problem, and finally a hood optimization problem. Using the proposed method, I could successfully generate designs that are similar to preferred solutions based on geometry or performance. Such cooperative optimizers with surrogate models can significantly improve the applicability of multi-objective topology optimization in solving real-world design problems.

Zusammenfassung

Die computergestützte Entwicklung hat den Prozess der Produktentwicklung durch Simulationen und rechnerische Optimierung von Produkten erheblich unterstützt. In den Anfangsphasen des Entwurfsprozesses werden häufig Topologieoptimierungsverfahren eingesetzt, um neuartige Strukturkonzepte zu entwickeln, indem die Materialauslegung in einem festen Bereich im Hinblick auf ein bestimmtes Ziel optimiert wird, z. B. die strukturelle Nachgiebigkeit oder die Energieabsorption bei einem Crash, vorbehaltlich vorgegebener Beschränkungen und Randbedingungen. Mehrzieloptimierungsmethoden können dazu beitragen, mehrere Ziele über verschiedene Disziplinen hinweg zu berücksichtigen und eine große Anzahl potenziell nützlicher Lösungen zu erhalten, was die Identifizierung vielversprechender Konzepte für die weitere Untersuchung und Entwicklung erschwert. Obwohl Methoden zur Wissensgenerierung wie das Clustering die Selektion von Designs auf der Grundlage von Erfahrung, Herstellungskosten oder anderen Leistungsmerkmalen ermöglichen, sind für komplexe technische Anwendungen anspruchsvollere Metriken zur Unterscheidung von Designs erforderlich. Außerdem erfordert die Topologieoptimierung teure Simulationen, um Lösungen in einem hochdimensionalen Lösungsraum zu finden. Daher ist es wirtschaftlicher, Regionen von Interesse zu identifizieren und nur die bevorzugten Lösungen zu generieren; dieser Ansatz wird von den *interaktive Methoden* in der Mehrzieloptimierung verwendet. In dieser Arbeit adaptieren wir diesen Ansatz für die Mehrziel-Topologie-Optimierung und entwickeln ein neuartiges interaktives Framework, das Referenzlösungen verwendet, um die gewünschten Lösungen zu generieren. Bestehende Clustermethoden werden verwendet, um die gewünschten Lösungen bequem auszuwählen. Da die Topologie von Strukturen in der Struktur- und Crash-Mechanik aus Gründen der Leistung und der Ästhetik wichtig ist, untersuchen wir auch den Einsatz Deep-Learning-Methoden wie Autoencoder bei der Analyse komplexer Topologien. Zu diesem Zweck führen wir eine neuartige Vergleichsmethode für Geometriemetriken ein und bewerten sie anhand synthetischer Datensätze, die für optimierte Strukturen repräsentativ sind. Unser Hauptziel ist die Entwicklung einer interaktiven Methode zur Generierung von Entwürfen, die der Referenzmenge ähnlich sind, unter Verwendung von Algorithmen zur Mehrziel-Topologie-Optimierung, bei denen modernste Methoden die Lösungspräferenz mithilfe von Gewichten für die Ziele spezifizieren, wobei eine Referenzmenge von Lösungen gegeben ist, die beispielsweise durch Clustering-Methoden mit einer geeigneten Metrik identifiziert wurde. Um unnötige Berechnungen zu vermeiden, werden Metamodelle verwendet, um vorherzusagen, ob ein bestimmter Satz von Gewichtungen zu einer bevorzugten Lösung führen wird. Wir demonstrieren unserem kooperativen Framework anhand eines cantilever Multi-Load-Case-Problems, eines Crash-Problem und schließlich eines Optimierungsproblems für die Motorhaube. Mit der vorgeschlagenen Methode konnten wir erfolgreich Designs generieren, die den bevorzugten Lösungen in Bezug auf Geometrie oder Leistung ähnlich sind. Wir glauben, dass solche kooperativen Optimierungsverfahren mit Ersatzmodellen die Anwendbarkeit der Mehrziel-Topologie-Optimierung bei der Lösung realer Designprobleme erheblich verbessern können.

Acknowledgement

For the past 4 years, I have worked on my doctoral thesis at the Technical University of Munich (TUM) in the School of Engineering and Design. I am grateful for the funding provided for my research by Honda Research Institute Europe GmbH (HRI), Germany. I am grateful for the constant guidance and support provided by my advisors Prof. Dr. Fabian Duddeck from TUM, Dr. Stefan Menzel, and Dr. Markus Olhofer from HRI. I am deeply grateful to Dr. Mariusz Bujny who was strongly involved in my day-to-day research activities when he was working at HRI. Special thanks to my mentor Dr. Michael Gienger at HRI for his guidance and support.

I wish to express my sincere thanks to Prof. Dr.-Ing. Fabian Duddeck for securing funding for my research. As my advisor, he constantly helped me understand the research methodology and guided me with his experience. I would like to thank Mariusz Bujny, who was my supervisor at HRI. Thanks to the regular meetings with Prof. Duddeck and Mariusz, I could make steady progress in my research. I developed the data and software used in this project with help from Mariusz. I am thankful to Markus Olhofer for his extremely helpful insights and for supervising my research in the past year. I would like to also thank Stefan Menzel who always found the time to give feedback about my research and publications. Finally, I once again express my gratitude to all my advisors for their research insights, time, support, and patience.

I am happy to have collaborated with Ernest Hutapea, Yasuyuki Shimizu, and Yuanze Wu for their master's thesis at TUM. Thanks to them I could explore new research ideas and questions.

I would like to thank my good friends and colleagues from the TUM and HRI. During the Covid pandemic when everyone was working from home, we would regularly meet online to discuss research and our daily activities. I will fondly remember my regular discussions with Tobias Lehrer, Sheikh Hoque, Felix Lanfermann, and Sneha Saha. Thank you, Arne Kaps, Norbert Ludwig, and Catherina Czech for your warm welcome and support throughout my Ph.D.

I am joyful about the time I could spend with my friends and family. It had always been relaxing and rejuvenating to visit my brother and parents back home. I am truly thankful to my friends and family for their help and support.

Contents

Acronyms

AMI	Adjusted mutual information score
BESO	Bi-directional evolutionary structural optimization
CAE	Computer-aided engineering
CD	Chamfer distance
CDO	Cooperative design optimization
CV	Cross-validation
DBSCAN	Density-based spatial clustering of applications with noise
DE	Differential evolution
DM	Decision-maker
DR	Dimensionality reduction
DTW	Dynamic time warping
EA	Evolutionary algorithm
ED	Euclidean distance
EMD	Earth mover distance
FE	Finite element
GA	Genetic algorithm
GD	Generational distance
GMM	Gaussian mixture models
GPR	Gaussian process regressor
HCA	Hybrid cellular automata
HV	Hyper volume

IR	Inclusion ratio
iSMO	Interactive framework for set-based multi-objective optimization
KNN	Prediction model: k-nearest neighbors
MMA	Method of moving asymptotes
MMC	Moving morphable component
MOEA	Multi-objective evolutionary algorithm
MOP	Multi-objective optimization problem
MSE	Mean square error
MTO	Multi-objective topology optimization
NMF	Non-negative matrix factorization
NVH	Noise, vibration, and harshness
OC	Optimality criteria
OPTICS	Ordering points to identify the clustering structure
PCA	Principal component analysis
PCAE	Pointcloud autoencoder
PF	Pareto front
RC	Rejection criterion
RM	Reference metric
RR	Rejection ratio
SEW-HCA	Scaled energy weighting - hybrid cellular automata
SIMP	Solid isotropic material with penalization
SOM	Self-organizing map

t-SNE	Manifold learning method: t-distributed stochastic neighbor embedding
TM	Target metric
TO	Topology optimization
UMAP	Uniform manifold approximation and projection
w-MTO	Weighted-sum method for MTO

Symbols

$\mathbf{p}$	Input hyperparameters of TO
$\mathbf{F}$	Pareto front
E	Young's modulus
$E_{\tan}$	Hardening modulus
ν	Poisson's ratio
σ_Y	Yield strength
Ω	Solution space
N_f	Number of objectives
s_m	Scaling factor of an objective
Φ	Level-set function
$r_{\min}$	Minimum filter radius
ν_f	Volume fraction
p'	Penalization parameter
K_p	Parameter for the proportional controller
S	Field variable
S_e	Field value of an element
$\overline{S}_E$	Average field value of neighboring elements
S^*	Target set-point for field variable
P_0	Material property without penalization
P	Material property after penalization
V_0	Volume of the design space

v_e	Volume of an element
$\tilde{G}_j$	An inequality constraint
$\tilde{H}_i$	An equality constraint
$\boldsymbol{u}$	State vector of the structure
ρ_e	Element density value
$\boldsymbol{\rho}$	Density vector
N_e	Number of elements
$\mathbf{f}$	Vector of objective functions
f_i	An objective function
$\mathbf{w}$	Set of weights for the objectives
w_i	Weight for an objective
U	Uniform distribution
p	Cluster label
C_p	Cluster with a label p
J_p	Feature variance of a cluster
$\boldsymbol{\mu}_p$	Mean of samples in a cluster
s	Silhouette score
$\bar{s}$	Average silhouette score of the clusters
D	Distance measure (metric)
D_{M}	Distance value measured using a metric M
$\mathbf{g}$	Geometric feature-vector
p_k	Principal component score
$\mathbf{G}$	A collection of geometries

G_m	A geometry
θ	Geometric property
ρ_p	Pearson correlation
ρ_s	Spearman correlation
g_i	Ground-truth label
$\mathbb{S}$	Preferred/reference set of solutions
$\mathbb{S}'$	Non-preferred solutions
$\mathbb{P}$	Preferred region
d	Number of features
$\mathbf{d}$	Optimal design
$C(\mathbf{w})$	Constraint function of iSMO
$C_{\mathbb{S}}$	Class of preferred solutions
$C_{\mathbb{S}'}$	Class of non-preferred solutions
$P_{\mathbf{f}}$	Regressor model for objective values
P_p	Regressor model for cluster labels
Δ	Spacing metric
s_{sil}	Average silhouette score
s_{div}	Diversity score
$\mathbf{E}$	Euler angles
$\mathbf{L}$	Principal axes
$\mathbf{C}$	Center of mass
q	Form factor of MMC
$\mathbf{W}$	Set of weight-vectors

$\mathbf{F}$	Set of objective-vectors
s_{CV}	Cross-validation score
B	Boundary patch

Part I

Introduction

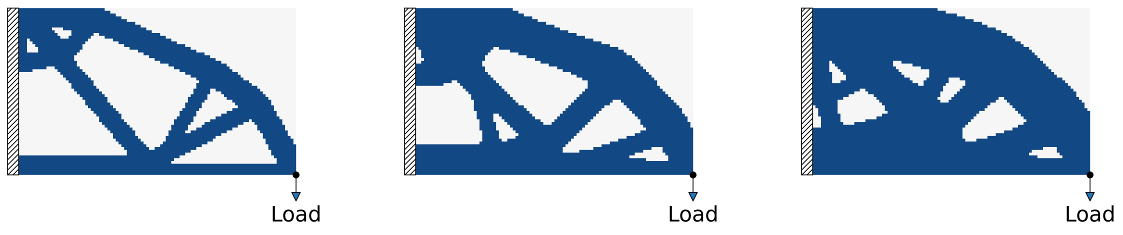
1. Motivation

Computer-aided engineering (CAE) has greatly supported the product development process through the creation, analysis, and optimization of designs. Recent advances in simulation tools and high-performance computing enable numerical optimization techniques to automatically generate a large set of concepts satisfying design requirements. Engineers can iteratively analyze the resultant designs and use computer-aided optimization methods to develop better products. Topology optimization (TO) methods, e.g., [1–6], are widely used to generate novel structural concepts; they optimize material layout—subject in general to a volume constraint in a given design space—for an objective such as structural stiffness or crash absorption under specific loads and supports. TO methods are more flexible than shape or size optimization methods [7] since the design can occupy any shape and have any topological structure within the design space. Despite its origins in structural mechanics, TO has found applications in a wide range of physical disciplines such as fluid mechanics [8], electromagnetics [9], and acoustics [10]; it is widely used in the aerospace and automotive industry, civil engineering, materials science, and biomechanics.

Design optimization methods such as TO can generate numerous designs using one or more of the following approaches:

1. *Parameter sampling.* Novel designs can be generated by varying the material properties, boundary conditions, and hyperparameters used by the optimization algorithm (e.g., [11]). Figure 1 shows exemplary designs that are generated using TO with different values for a hyperparameter called *volume fraction*.
2. *Multimodal optimization.* Highly complex and nonlinear objectives are normally multimodal, i.e., they have several local optima. Since not all the constraints are known in the early development phase, having a set of local optima is useful, in case some of them violate unknown constraints in the future. Such designs can be identified using evolutionary algorithms (EAs) [12] or by restarting gradient-based optimization algorithms, such as TO, from different initial configurations [13], converging to a different optimum in each of the runs.
3. *Multi-objective optimization.* In practice, designs may need to be optimized for multiple objectives, e.g., energy absorption under crash loads, and structural stiffness under smaller static loads. Multi-objective TO [14] with conflicting objectives yields a set of Pareto-optimal designs, where choosing a design with better performance for one objective results in performance deterioration of at least one other objective.

Using these methods, a large number of potentially useful solutions can be generated, which makes it challenging for an engineer to find interesting solutions. *Knowledge generation* meth-



(a) $\nu_f = 0.3$ (b) $\nu_f = 0.5$ (c) $\nu_f = 0.7$

Figure 1 TO can be used to generate multiple optimized structures by varying the allowed volume fraction ν_f of material in the design space. We minimize the compliance of an MBB beam, given a 2D rectangular design space of 100×60 elements with the fixed left boundary and a static load of 1 unit, using the 88 line code developed by Andreassen et al. [15]. Some of the hyperparameters used are: filter radius is 2.5, penalization factor is 3, and sensitivity filtering (ft $= 1$). The remaining hyperparameters and the material properties are as described in the reference paper/code.

ods such as statistical analysis [16], visualization tools [17], and clustering [18] can assist engineers in exploring the design dataset and result in *concept identification*, i.e., selection of promising concepts for further analysis and development among the given set of solutions. Despite the availability of such methods, engineers still need to formulate the criteria for selection based on the application. Furthermore, TO methods need to incorporate user preference to explore regions of interest in the solution space.

In practice, TO methods may need to yield designs similar to a reference design chosen because of economical or manufacturing limitations. Yousaf et al. [19] use similarity constraints to adapt TO methods such as SIMP (solid isotropic material with penalization) and HCA (hybrid cellular automata) for generating solutions similar to a reference solution. Zhang et al. [20] use neural style transfer to embed artistic flavor into designs by adapting the SIMP method. However, it is still challenging to identify a single reference solution or style. Furthermore, these methods cannot directly handle multi-objective TO methods.

Multi-objective TO typically needs expensive simulations to find solutions in a high-dimensional solution space. In multi-objective optimization, so-called *interactive* methods [21] generate only the preferred solutions given reference solutions or directions, which is more economical since solutions are found only in the region of interest. In this thesis, we generalize this notion to multi-objective topology optimization (MTO) and refer to it as cooperative design optimization (CDO), as outlined in Figure 2. As part of this, we propose a *cooperative framework*, which emphasizes the cooperation between users and a computer, where users select preferred solutions with help from the framework, which then predicts inputs required by TO to generate more solutions similar to the preferred set.

As mentioned previously, design exploration is a challenging task and is critical in the *concept identification* task. The preferred designs can be selected based on experience, manufacturing cost, or other performance attributes. For instance, the dream lens tool [11] is an interactive

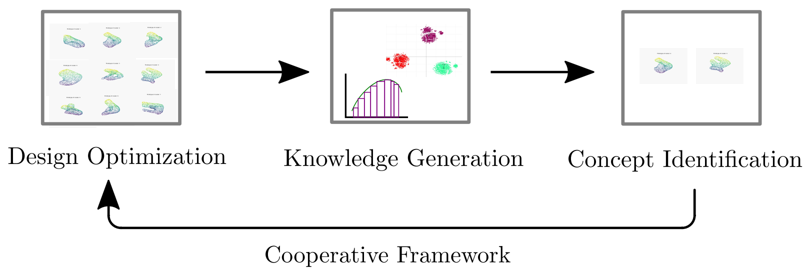


Figure 2 Cooperative design optimization: Generate a set of solutions using a design optimization method. Analyze solutions using knowledge generation methods and identify preferred concepts based on user criteria. Based on the preferred designs, the *cooperative framework* predicts the inputs required by the optimization method to yield a new set of preferred solutions.

framework that guides a designer to a set of *interesting* designs using, for example, range constraints on a performance attribute. By contrast, clustering is an unsupervised machine learning method that can be used for exploring designs; it partitions the designs into groups of designs, where each group contains *similar* designs according to a *metric* [22]. For example, airplanes can be grouped into shapes that have similar performance *features* (say, lift and drag values), if the Euclidean distance in the *feature* space is used as the metric for clustering. Hagg et al. [23] recommend the use of representative solutions from each cluster for presenting the user with diverse solutions. So, given a set of optimized designs, we can identify structures with distinct properties, provided a *metric* is available to measure differences between designs. While simple metrics such as Euclidean distance in objective space can be used, more sophisticated metrics are needed for complex engineering applications. In this work, our first aim is to develop metrics for distinguishing geometrical structure and deformation behavior of TO results; these design properties are of critical importance in the field of structural mechanics. Furthermore, such metrics along with clustering are very useful in selecting solutions, which are used to demonstrate our cooperative framework later in this thesis.

Multi-objective evolutionary algorithms (MOEAs) such as NSGA-II [24] are successful in identifying diverse Pareto-optimal solutions. Furthermore, *interactive methods* exist in MOEAs that can generate only the preferred solutions, given reference solutions or directions [21, 25]. Since the search space in TO problems is in general high-dimensional, it is difficult or not feasible to use MOEAs. Feature mapping methods can reduce the dimensionality of the problem so that TO methods such as the evolutionary level set method [3, 26] can be used. However, MOEAs require a large number of objective function evaluations, and using them for MTO can be prohibitively expensive. It is more economical to use weighted-sum approaches for MTO such as scaled energy weighting - hybrid cellular automata (SEW-HCA) [27], where the users specify their preferences with a set of weights associated with the objectives. In particular, SEW-HCA is useful for optimizing crash performance, where analytical gradient information is not available. So, the second and final aim of this thesis is to develop a cooperative framework for such methods. Since TO is computationally expensive, we research the use of metamodels that can predict when a given parameter will lead to a desired solution. Chapter 2 summarizes the aims and objectives of this research.

While there are different approaches for TO (Chapter 3), each method can be understood as an operator with input hyperparameters $\mathbf{p} = \{p_1, p_2, ..., p_{n_\mathrm{p}}\}$. By varying the values of **p**, a set of feasible solutions can be generated. For example, in SEW-HCA [27], one may vary the weights ($\mathbf{p} := \mathbf{w}$) for different objectives to yield multiple non-dominated solutions. Using knowledge generation methods such as clustering, the user can find preferred solutions in the dataset (Chapter 4), which can be used as input to the cooperative framework. Here, we use clustering to conveniently identify preferred solutions using the following approach: (i) cluster designs into distinct groups, and (ii) choose one or more clusters as the preferred set. Our objective is then to find the new input **p** that will result in more of the desired solutions.

Concept identification: A critical component in CDO is the reference set of solutions used to generate the new iteration of solutions. Given a criterion for selecting solutions, clustering is an effective data mining tool that can help identify diverse solutions without supervision [23]. Clustering requires a suitable *metric* that measures the degree of similarity between designs based on the user-defined criterion. In multi-objective optimization, the difference between the objective values is a commonly used criterion to select designs [17]. In practice, engineers may need more sophisticated metrics based on the application. In MTO, differences in the topology of structures might be critical because of aesthetics or engineering performance; the latter being very important in the field of structural mechanics [28, 29]. While state-of-the-art deep learning methods exist for distinguishing 3D shapes such as pointcloud autoencoder (PCAE) [30, 31], it is not clear if they can be used with TO results, which tend to have very complex topologies. Here, we evaluate the utility of PCAE in clustering designs based on the 3D geometrical structure using the methods and data described in Chapter 5. PCAE is found to be extremely useful in object classification of TO results, as shown in Chapter 7.

Cooperative optimization: Interactive MOEAs can be efficiently used to generate only the desired solutions in a high-dimensional solutions space by formulating user preference through reference directions or solutions [21]. Since MOEAs are very expensive to use as an MTO, we focus on weighted-sum approaches such as SEW-HCA, where the user preference as weights for the objectives is needed but is unknown in practice. It is easier for an engineer to select solutions—e.g., using clustering—instead of specifying the exact inputs needed by the MTO method to generate the new solutions. So, we develop a novel and general interactive framework for set-based multi-objective optimization (iSMO) that can predict the required inputs, given a set of reference solutions (Chapter 6). MOEAs such as NSGA-II are very effective in generating diverse solutions in the Pareto front while also addressing the user constraints [24]. Following the framework of NSGA-II, we use a genetic algorithm (GA) with the input **p** of MTO as the decision variable and evolve the populations of solutions into the preferred region in the Pareto front, while using *crowding distance* [24] for diversity in the objective space. The diversity measure ensures that the solutions are well-distributed in the objective space. To avoid too many expensive simulations, we propose the use of machine learning models for the following tasks in iSMO: (i) A classifier model that can decide if a given input **p** will result in the preferred solution; (ii) A regression model to predict from an input **p** the objective values, which are required by the crowding distance operator to ensure diversity.

In this work, we demonstrate our iSMO approach with SEW-HCA as the MTO method. In Chapter 8, we use two optimization problems, a static compliance problem with a cantilever plate and a crashworthiness optimization problem with a simply supported beam, to generate initial solutions. Clustering methods are conveniently used to find desired solutions based on objective values or the geometrical structure. iSMO is then successfully used to generate more of the desired solutions. In Chapter 9, we choose a more complex engineering problem of optimizing a hood model for crashworthiness. Once again, iSMO is used to generate the preferred solutions, which are then evaluated using measures of diversity and similarity to the reference solutions.

In the following, we summarize the aims and objectives of this thesis in Chapter 2 followed by Chapter 3 on various design optimization methods such as SEW-HCA. Chapter 4 discusses the challenge of design exploration using methods such as clustering that can be used to select preferred solutions. Furthermore, we research the use of autoencoders in identifying distinct structures in TO results as well as develop novel metrics of geometry and deformation behavior, which are critical properties for analyzing structural and crash mechanics. Chapter 5 introduces a new approach for evaluating different metrics of geometry while considering the topological complexity of TO results. Chapter 6 introduces the proposed cooperative method for MTO, which uses an evolutionary algorithm (EA) for evolving solutions while avoiding expensive simulations using machine learning models. We demonstrate our cooperative framework using different TO problems in Chapters 8 and 9. Finally, we have the discussion and conclusion in Chapter 10.

2. Aims and Objectives

The following are the aims and corresponding objectives of this work. As mentioned previously, design selection is a challenging task. So, the first aim is related to analyzing and identification of *preferred designs* among a set of given solutions. Given the *preferred* designs, the second aim is to develop a *cooperative method* that yields a new set of hyperparameters needed by TO for generating more of the *preferred* solutions. This integrated approach helps to efficiently generate only the desired designs, and hence avoid expensive simulations for cases where they provide no relevant new information.

1. *Aim:* Develop a method to analyze TO results and support the identification of a set of *preferred* solutions.

   Objectives:

   - Investigate the use of clustering, an unsupervised machine learning method, for design selection.
   - Develop and evaluate methods to cluster designs based on their geometrical structure, which is a critical attribute of the TO results.
   - Develop other novel metrics for distinguishing designs such that they can be used with clustering for design selection.

2. *Aim:* Given a set of *preferred* designs, develop a cooperative method to generate more of such designs.

   Objectives:

   - Develop the cooperative method to use multi-objective TO methods that are based on the weighted-sum approach. This class of TO methods can potentially generate a large set of distinct structures and are indispensable in the multi-disciplinary design process.
   - Incorporate metamodels in the cooperative method such that expensive simulations are avoided while predicting the new set of hyperparameters.
   - Investigate the use of a cooperative method to find solutions similar to a preferred set based on objectives and geometrical structure.

3. Design Optimization

In computer-aided design, engineers iteratively improve solutions based on certain desirable properties while also satisfying multi-disciplinary requirements. For example, lightweight vehicles result in the reduction of required material, manufacturing costs, and fuel consumption. However, vehicles should also fulfill regulatory requirements for crashworthiness as well as the customer requirements for noise, vibration, and harshness (NVH) characteristics [7].

The design optimization process using physical prototypes and experiments is very expensive and hence restricts the available solution space for exploration. By contrast, computational design optimization using virtual models and simulations allows engineers to easily find potential solutions, analyze their properties using simulations, and obtain suggestions for improvement. Furthermore, we can generate novel solutions that meet multi-disciplinary requirements.

In design optimization, engineers first define parameters that can be varied to modify the structure of a design. In size or shape optimization, parameters controlling the size or shape of different components of the design are identified [32–35]. Given an objective, optimal parameters are derived using optimization algorithms such as the gradient-descent method [36] or evolutionary algorithms [37]. By contrast to size and shape optimization methods, TO methods parametrize a fixed design domain using a set of material distribution functions to determine the optimal size, shape, and topology of the design [1]. TO typically needs to find solutions in high-dimensional decision space and needs efficient optimization algorithms such as solid isotropic material with penalization (SIMP) [1], bi-directional evolutionary structural optimization (BESO) [38], hybrid cellular automata (HCA) [39], and level-set methods [40, 41]. Level-set methods optimize the boundary between the material and void space [42]. Evolutionary algorithms can even be used with geometric basis functions for level-sets as proposed by Bujny et al. [3, 26, 43]. At the early stages of design, these TO methods are especially useful since they automatically yield optimized concepts.

In this thesis, the cooperative approach uses multi-objective TO methods (Chapter 6), where we find the weight-vectors for objectives that can generate the preferred solutions. For other methods such as ESO/BESO and level-set methods—or even SIMP and HCA, our approach needs to be tailored to use the relevant hyperparameters, e.g., volume fraction or/and filter radius for SIMP, HCA, and level-set methods; rejection and inclusion ratio may be varied for BESO. Given a set of preferred designs, new hyperparameters can be found by our approach so that more of the preferred solutions can be generated. In this work, as a start, we restrict the investigation to multi-objective TO methods that integrate weighted-sum approaches with SIMP and HCA. These TO methods can yield distinct structures that are trade-offs and might be equally interesting to the user. As background knowledge, the following sections describe the TO methods from the literature: Section 3.1 describes single-objective TO methods including

SIMP and HCA, which form the basis for the multi-objective TO methods in Section 3.2.

3.1. Topology Optimization

TO methods optimize the material distribution within a given design space for a given objective. Among the different approaches, density-based TO methods discretize the design space into elements, whose density is used as a decision variable for optimization. An exemplary density-based TO called SIMP uses the power-law approach to penalize material properties of elements with intermediate densities [1]. So, SIMP results in a continuous optimization problem, which enables the use of gradient information, unlike the discrete optimization problem which allows elements to either have a fixed material density or no density at all.

Given an objective function f, density-based TO discretizes the design space into N_e elements and optimizes the density vector $\boldsymbol{\rho} = [\rho_e]_{e=1}^{N_e}$ (decision variables), i.e., the vector of relative densities of the elements. A general density-based TO problem can be formulated as follows:

$$\begin{aligned}
\min_{\boldsymbol{\rho}} \quad & f(\boldsymbol{\rho}, \boldsymbol{u}(\boldsymbol{\rho})) \\
s.t.: \quad & H_0(\boldsymbol{\rho}) = \sum_{e=1}^{N_e}(v_e \rho_e) - V_0 = 0, \\
& \tilde{G}_j(\boldsymbol{\rho}, \boldsymbol{u}(\boldsymbol{\rho})) \leq 0, \quad j = 1, ..., J, \\
& 0 \leq \rho_{\min} \leq \rho_e \leq 1, \quad e = 1, ..., N_e,
\end{aligned} \tag{3.1}$$

where $\boldsymbol{u}(\boldsymbol{\rho})$ is a state vector representing the response of the structure to specific load conditions, with the material distribution in the structure given by $\boldsymbol{\rho}$. The volume of an element e is v_e and the constraint H_0 restricts the cumulative volume to V_0. Additionally, there can be J inequality constraints $\tilde{G}_j$ that depend on the state and density vectors. Depending on the optimization algorithm, each elemental density ρ_e may need to have a minimum value $\rho_{\min} > 0$ to avoid numerical issues. In the following, we briefly describe the methods SIMP and HCA used for TO in this thesis.

3.1.1. Solid isotropic material with penalization (SIMP)

Since the dimension of the density vector is generally in the order of 10^3 to 10^6, global search algorithms such as evolutionary algorithms cannot be used for TO. So, sensitivity values (gradients) of the objective function with the decision variables need to be used for solving the TO problem efficiently. For example, SIMP with optimality criteria (OC) [1] or the method of moving asymptotes (MMA) [44] can use gradient information to iteratively optimize the structures. The OC method can also be extended to handle multi-objective problems [45].

Given a base material property P_0, the material property of an element with intermediate density ρ_e is given by

$$P(\rho_e) = \rho_e^{p'} P_0, \tag{3.2}$$

where p' is the penalization parameter. Using this method, material properties such as Young's modulus are penalized and used in finite element simulations. To use SIMP with the OC method, we calculate the sensitivity (gradient) of the objective function with respect to changes in element densities, and the OC method uses the sensitivity values to iteratively optimize the densities.

In vehicle design, gradient-based methods such as SIMP cannot be easily used with crash simulations, which involve highly nonlinear phenomena due to multiple contact surfaces with friction, nonlinear materials with strain-rate effects, material failure, and significant element rotations. Using simplified crash models, it is possible to use gradient-based optimization such as the ground structure approach [46]. However, in industrial applications, the gradient information cannot be used for optimization because of the complexity of crash models, strong nonlinearities, and numerical noise. Furthermore, crash simulation software does not generally give access to the underlying methods and gradients. Therefore, heuristic approaches such as HCA [39] are preferred.

3.1.2. Hybrid cellular automata

HCA [39] is a non-gradient-based approach that can be used to minimize the compliance for a static load or to maximize the energy absorption under a crash load. Since the response of structures to crash loads is noisy and the gradients are not reliable, HCA is preferred to SIMP in crash applications since HCA does not require gradients. Furthermore, HCA can be easily extended to include multi-objective problems using the approach proposed by Aulig et al. [27], as described in the next section.

HCA is based on the idea of a cellular automaton which consists of a regular grid of cells. In each generation, the state of each cell is updated based on the states of its neighboring cells in the previous generation. HCA tries to achieve a uniform distribution of a field variable S among the elements of the design domain, which results in optimal structures for certain objectives. In linear elastic problems with static loads, HCA yields structures with minimum compliance by defining strain energy density as S [39]. Similarly, for crash loadcases, HCA yields structures with improved crashworthiness performance by defining internal energy density as S [47]. While HCA includes material plasticity and contact mechanics in the optimization, Patel et al. [47] indicate that HCA is not suitable for large deformations due to buckling, where homogenization of internal energy is not an appropriate optimization criterion.

HCA uses a control-based update scheme to achieve a uniform distribution of field variables S throughout the design space, which corresponds to the idea of distributing the load equally throughout the structure. As described previously, this approach can yield structures with minimal compliance or maximal crash energy absorption depending on the load case and the field variable. For the density update, we use the proportional controller with a parameter K_p, as described by Tovar et al. [48]:

$$\Delta\rho_e^{(k)} = K_p(\overline{S}_e^{(k)} - S^{*(k)}), \quad \forall e \in \{1, ..., N_e\}, \tag{3.3}$$

where $\Delta\rho_e^{(k)}$ is the required change in the density of element e at iteration k. Additionally, the changes are limited by a move limit parameter to avoid numerical instability. The effective field state $\overline{S}_e$ is defined as the average of field states of the neighboring elements which are defined by the filter radius parameter $r_{\min}$. The usage of $\overline{S}_e$ prevents numerical issues such as the checkerboard patterns [13] and imposes a minimum length scale. For dynamic loadcases such as crash loads, the $\overline{S}_e$ is additionally averaged over time. The target set-point S^* is iteratively adjusted such the volume constraint is satisfied when the density is updated using S^* (3.3). The optimization process is stopped when the changes in density are smaller than a prescribed tolerance or when the maximum iterations are reached. The latter is more common in practice, given that the simulations are expensive and satisfactory design improvements may be achieved with a fixed number of iterations. For more details on HCA, see [5, 6, 48–50].

To calculate the field states, HCA uses the same material interpolation method as SIMP. For crash loadcases, additional properties such as yield stress and the strain-hardening modulus are also interpolated. However, the penalization for different properties can be different. HCA is suitable for optimizing energy absorption in crash scenarios but not for other crash objectives such as minimization of intrusion, forces, or acceleration under impact loads. In this thesis, we use HCA for the optimization of crash energy absorption.

3.1.3. Other approaches

ESO [51] is a simple evolutionary structural optimization method that removes material based on a hyperparameter called rejection ratio (RR) based on a rejection criterion (RC) such as von Mises stress. ESO removes elements in the design space whose RC (e.g., von Mises stress) value is less than the RR times the maximum RC value over the structure. This process is repeated until a steady state is reached, completing an iteration of ESO. Material is gradually removed in multiple iterations by increasing the RR value in steps from an initial value (e.g., 0.01) to a final value (e.g., 0.10).

BESO [38] is a bidirectional ESO method since it allows for the addition and removal of elements, unlike ESO which only allows for the removal of material. Hence, BESO can yield better solutions since it uses a larger search space. Material is added or removed in the structure using RR and inclusion ratio (IR) respectively. Given an optimization criterion such as von Mises stress, *under-stressed* elements, whose criterion value is less than RR times the max value, are removed. Similarly, *over-stressed* elements with values greater than IR times the max value are marked and new elements are added in their vicinity.

Allaire et al. [42] describe a level-set method, which progressively changes the boundary of the structure using a shape sensitivity analysis, while constrained to a fixed volume fraction of the design space. The structure is represented by a level-set function (Φ), whose value is positive, zero, and negative in the interior, boundary, and exterior of the structure respectively. In each iteration, the level-set function value changes and the boundary advances with normal velocity given by the shape derivative. Bujny et al. [3] propose an evolutionary level-set method, which can handle highly nonlinear and discontinuous problems, such as crashworthiness optimization problems, where sensitivity analysis is not possible or too expensive. Each design is represented using a finite set of elementary components; each component is defined by a simple level-set function that can be morphed using a small set of design variables. This approach reduces the dimensionality of the problem and enables the use of evolutionary algorithms. For a given objective, Bujny et al. use an evolutionary strategy to find the design variables of the optimal solution.

For design generation, given a TO method, we need to select relevant hyperparameters whose variation results in a distinct set of structures. As described later in Chapter 6, given a set of preferred solutions and the associated hyperparameters that generated them, the proposed cooperative method can learn to find the hyperparameters required to generate solutions similar to the given set. In this work, instead of using the single-objective TO methods described in this section, we use a multi-objective TO described in the next section. We chose the latter class of methods since they provide a convenient way to generate multiple solutions with distinct structures. However, the choice is arbitrary to some extent. By changing the *input features* (hyperparameters) used to train the regression and classification models in the presented collaborative approach, one can directly incorporate other TO methods.

3.2. Multi-objective Topology Optimization

In automotive design, engineers need to consider a large number of multi-disciplinary requirements. For example, Duddeck [7] investigates use cases with multiple requirements related to structural statics/dynamics, crashworthiness, and NVH properties. If the design goals are conflicting, this can lead to multi-objective optimization problems. Focusing on topology optimization, we describe the methods for MTO starting with the general formulation of multi-objective optimization.

3.2.1. Multi-objective Optimization

In an optimization problem, we search for a solution, represented using a vector $\mathbf{x}$ of decision variables, which is to be found in a feasible space of solutions Ω. In a multi-objective optimization problem (MOP), the multiple criteria used to select solutions can be represented by a vector of N_f objective functions: $\mathbf{f}(\mathbf{x}) = [f_i(\mathbf{x})]_{i=1}^{N_f}$. MOP can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{x}\in\Omega} \quad & \mathbf{f}(\mathbf{x}), \\ \text{subject to :} \quad & \tilde{G}_j(\mathbf{x}) \leq 0 \quad j = 1, ..., J, \\ & \tilde{H}_i(\mathbf{x}) = 0 \quad i = 1, ..., I, \end{aligned} \tag{3.4}$$

Hence, in an MOP, we are trying to minimize the objective functions subject to equality and inequality constraints. If we need to maximize an objective function f_i, we can always reformulate it as $f'_i(\mathbf{x}) = -f_i(\mathbf{x})$ so that it fits the above definition. For conflicting objectives, MOP does not have a single solution that optimizes all the objectives. Instead, we have multiple solutions—called non-dominated or Pareto-optimal solutions—that are equally important. This can be understood based on the concept of dominance among solutions.

A solution $\mathbf{x}$ is said to *dominate* $\mathbf{y}$, i.e., $\mathbf{x} \prec \mathbf{y}$, if the following conditions are satisfied:

$$\begin{aligned} f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \ \forall \ i \in \{1, ..., n\}, \\ f_j(\mathbf{x}) < f_j(\mathbf{y}) \ \exists \ j \in \{1, ..., n\}. \end{aligned} \tag{3.5}$$

So, $\mathbf{x}$ is better than $\mathbf{y}$ for at least one of the objectives. Any optimal solution of MOP is a non-dominated solution $\mathbf{x}$, where no other solution $\mathbf{y} \in \Omega$ dominates $\mathbf{x}$. The set of all non-dominated solutions is the Pareto-optimal set; the set of corresponding objectives is the Pareto front (PF). Since the Pareto-optimal set can have unlimited solutions, we typically approximate it using a set of solutions near the PF.

MOEAs are widely used methods for solving MOPs. Diverse solutions can be obtained using MOEAs such as NSGA-II [24] or NSGA-III [52, 53]. MOEAs find the Pareto set, the set of all the Pareto-optimal solutions, in a single run. However, they need a large number of objective function evaluations, which may be prohibitive when the number of decision variables is high and/or the objective functions are expensive to evaluate. MTO problems generally have extremely high-dimensional decision space (in the order of millions) and expensive objective evaluations since every evaluation corresponds to a finite-element solution. Therefore, MOEAs cannot be directly used for solving MTOs. The problem of high-dimensionality can be alleviated to some extent by using feature mapping methods such as the evolutionary level-set method [3, 26]. Instead of MOEAs, weighted-sum MTO approaches such as SEW-HCA [27] can be used to efficiently optimize the material distribution. We will discuss this approach next.

3.2.2. Weighted-sum Approach

As discussed previously, SIMP can efficiently optimize the material layout for a single objective by calculating the sensitivities of the objective function with respect to each element density in each iteration. For multiple objectives, we can use the weighted sum of the sensitivities for different objectives to obtain for each element an aggregate sensitivity value. Using these new sensitivity values for elements, any of the single-objective TO methods can be used for solving multi-objective problems. Similarly in HCA, where the goal is to homogenize a field variable across the elements, a weighted-sum of field variable values in a given element can be used to handle the multi-objective problem.

Aulig et al. [27] proposed a method called SEW-HCA, which can concurrently minimize compliance of a structure for one or more static loadcases as well as maximize its energy absorption for one or more crash loadcases. Given a set of weights for the objectives by the user, SEW-HCA scales the weights since each objective may involve different scales of element sensitivities. Due to rescaling, SEW adheres better to user preference. For example, consider an MOP with equal weights, say $(0.5, 0.5)$, for two objectives: (i) minimize the structural compliance under a static load, which is measured by the total strain energy stored, and (ii) maximize the energy absorption under a crash load, which is measured by the total internal energy absorbed. Since energies involved in crash loads are generally much higher than in static loadcases, directly using the weights may give higher importance to the elemental sensitivities of the crash loadcase. So, SEW-HCA scales the weights and the user can express his/her preference more easily.

Given a set of weights $\{w_m\}_{m=1}^{N_f}$ for the N_f objectives, SEW-HCA weighs the field variables as follows:

$$S_e = \sum_{m=1}^{N_f} w_m S_e^m / s_m, \tag{3.6}$$

where S_e^m is the field state of element e for m-th objective function. s_m is the corresponding scaling factor used to account for the magnitude of objective values to avoid the dominance of crash loadcases whose energies are usually orders of magnitude higher than those of static loadcases. The scaling factor is given by

$$s_m = W_m / W_{\min}, \tag{3.7}$$

where $W_{\min} = \min W_m$. For static loads, W_m is the total strain energy stored. For crash loads, W_m is the total internal energy absorbed. Aulig et al. [27] indicate that the scaling factor is usually calculated only at the initial iteration since the energy levels change only slightly during the optimization. Depending on the use case, they may need to be calculated at every iteration. Note that Aulig et al. refer to w_m in Eq. (3.7) as user preferences in the original paper.

While SEW-HCA decouples the weight-vectors from the scale of objective function values, it is still not easy to choose the weight-vectors required to generate desired solutions. A possible approach is to analyze a sample set of diverse solutions, using an appropriate weight sampling method [54–56], before deciding on a particular design using data analysis techniques. In this thesis, I sample a set of weight-vectors using the Das-Dennis approach [57] and then generate a solution using SEW-HCA for each weight-vector.

3.3. TO Datasets

In this section, we describe TO methods as well as the material properties used to obtain three datasets. Later, in the following chapters, we analyze these datasets based on geometrical structure, deformation behavior, and other performance attributes of the designs.

3.3.1. Dataset-1: Cantilever Plate with Two Static Loads

For this dataset, the cantilever plate is optimized for two objectives corresponding to two static loads (Figure 3). For each static load $i \in [1, 2]$, the objective f_i is to minimize the structural compliance which is measured by the total internal strain energy stored in the structure after load application. SEW-HCA iteratively optimizes the design by calculating the strain energies for each load case separately using the simulation software LS-DYNA® [58].

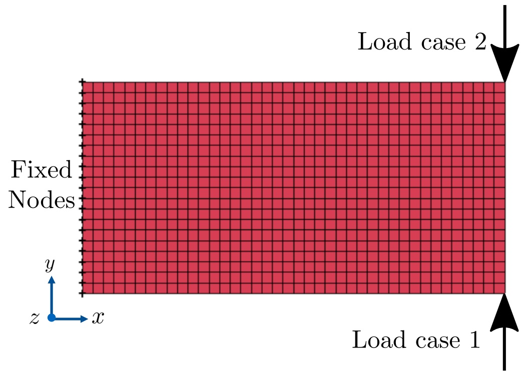


Figure 3 A cantilever plate with fixed nodes and two applied loads. The magnitude of each load is 0.2 N.

The design with the dimensions $400\ \mathrm{mm} \times 200\ \mathrm{mm} \times 10\ \mathrm{mm}$ is discretized into $40 \times 20 \times 1 = 800$ solid elements (8-noded hexahedrons). The design contains a bilinear elastoplastic aluminum material (MAT 24 in LS-DYNA) with the following properties: maximum mass density $\rho_{\max} = 2.7 \times 10^3\ \mathrm{kg/m^3}$, Young's modulus $E = 70\ \mathrm{GPa}$, Poisson's ratio $\nu = 0.33$, yield strength $\sigma_Y = 117\ \mathrm{MPa}$, and hardening modulus $E_{\tan} = 49\ \mathrm{GPa}$. For a given set of weights, SEW-HCA iteratively optimizes the elemental densities using the following hyperparameters: the maximum number of iterations is 25, the allowed volume fraction is 0.4, the move limit is 0.1, and the penalization factor is 3.

3.3.2. Dataset-2: Simply Supported Beam with a Crash and a Static load

A simply supported beam is optimized for two objectives: maximize energy absorbed f_1 for a crash load and minimize compliance f_2, i.e., total strain energy stored, for a static load. Figure 4 shows the load configurations and the boundary conditions. The static and crash objective values are calculated using implicit and explicit LS-DYNA solvers respectively. Once again, for a given set of weights for the objectives, SEW-HCA iteratively optimizes the design by calculating the objective values for each load case separately.

The design shown in Figure 4 has the dimensions $600\ \mathrm{mm} \times 50\ \mathrm{mm} \times 50\ \mathrm{mm}$ and is discretized into $120 \times 10 \times 10 = 12,000$ elements. The material properties and hyperparameters for SEW-HCA are as in test-case 1 (Section 8.2.1) except that a smaller move limit of 0.05 is chosen for stabilizing the optimizer. The crash load is applied using a rigid hollow cylinder with the following properties: thickness is 3 mm, inner diameter is 30 mm, and length is 100 mm. The cylinder weighs 0.2 kg while the beam weighs 4.2 kg.

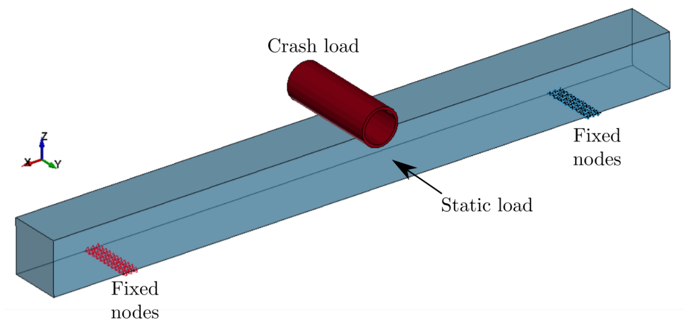


Figure 4 A simply supported beam is optimized for a crash load from a cylinder (top) and a static load (right face). The supports are implemented using fixed nodes with zero prescribed displacement. The crash load is applied using a rigid hollow cylinder, which crashes into the beam and travels a displacement of $100\ \mathrm{mm}$ in $0.1\ \mathrm{s}$, while constrained to move only along the z-axis without rotation. The static load of $10^4\ N$ is radially distributed at the center of a lateral face with a radius of 20 mm.

3.3.3. Dataset-3: Cube Design Space

For this dataset, TO [1] is used to generate designs with widely varying topologies. We optimize the material layout in a unit cube to have minimal structural compliance for arbitrary loads and fixed nodes (Figure 5). Hence, we simulate an extreme use case with flexible boundary conditions and obtain a set of designs with widely different geometrical structure. Note that in practice, boundary/load conditions are generally fixed for a given design task [1]. However, here we generate a dataset by using varied boundary conditions in order to generate an artificial, yet, challenging dataset that can be used to evaluate the clustering approach discussed in the next chapter.

Each boundary configuration, defined by the fixed nodes and load conditions, is obtained using the following method:

- Nodes are fixed (zero displacements in all directions) in an arbitrary rectangular patch B in the face with $x = 0$ (Figure 5). The edges of rectangle B are parallel to the edges of the face with $x = 0$. The bounds of B along y and z are obtained by drawing four samples from the uniform distribution $U(0, 1)$.
- Two arbitrary unit loads are applied to the design. Each load is distributed in a radius of 0.1 units about a randomly chosen center c in the unit cube. The three coordinates of c are sampled from a uniform distribution $U(0, 1)$. The components of each load vector are also sampled using $U(0, 1)$, but are normalized after obtaining the three components.

For each boundary configuration, SIMP [1] is used as TO for generating an optimized design with minimal structural compliance. We use a linear elastic material for steel with the following properties: mass density of $7.83 \cdot 10^3\ \mathrm{kg/m^3}$, Young's modulus of $207\ \mathrm{GPa}$, and Poisson's ratio of 0.33. By iterating over multiple configurations, a total of 100 designs are generated. This dataset has no prescribed subclasses, but it is representative of TO results obtained in

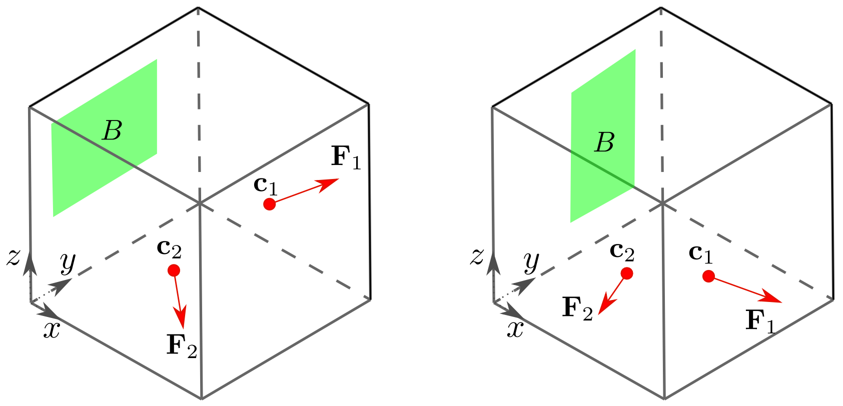


Figure 5 Two possible boundary configurations for the unit cube. TO minimizes structural compliance under radially distributed loads $\mathbf{F}_1$ and $\mathbf{F}_2$ with centers at $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively while the nodes are fixed in the patch B.

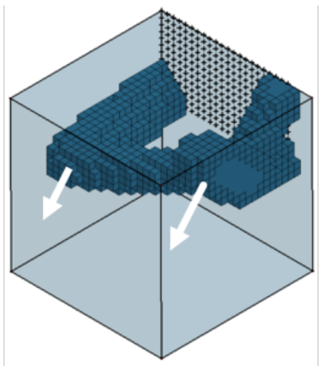


Figure 6 The optimized design occupies a volume fraction of 0.3 with the maximum stiffness when the given two loads are applied (white arrows). The dotted points on the face to the right mark the boundary patch B.

practice. Figure 6 shows an example design from this dataset. Using this dataset, geometric feature-vectors can be qualitatively evaluated as discussed later in Chapter 7.

Summary

In this chapter, we presented two TO methods: SIMP and HCA. SIMP with OC method is a gradient-based TO widely used for structural optimization. HCA is a non-gradient-based heuristic method that can optimize for crash energy absorption or static load compliance. SEW-HCA is an extension of HCA that can concurrently optimize for multiple objectives such as compliance under static loadcases and energy absorption under crash loadcases.

Most of the TO methods involve several hyperparameters that need to be chosen. For example, both SIMP and HCA require a filter radius and a volume fraction; each choice of parameter results in a different design. For SEW-HCA, the weights for different objectives need to be chosen as well. In this thesis, we addressed this problem and provided methods to support the user in choosing the weight-vectors. We believe that the proposed methods in the later chapters can be directly extended to handle any optimization method and for choosing the hyperparameters required to generate desired solutions.

Part II

Methodology

4. Knowledge Generation

As discussed in the previous chapter, design optimization methods can potentially generate a multitude of designs. One of the challenges is to explore and select *interesting* solutions for further analysis. Furthermore, the set of desired solutions can be used to inspire a new generation of solutions, e.g., using *interactive* multi-objective optimization. In this thesis, methods for analyzing designs are referred to as *knowledge generation* methods.

In this chapter, we describe different methods for analyzing a given set of solutions to identify desired solutions based on different criteria. Section 4.1 describes the state-of-the-art methods for knowledge generation using descriptive statistics [16], visualization tools [17], and clustering methods [22]. Clustering can be used to group structures that are similar based on a given *metric*, where each group (cluster) of solutions can be analyzed and the desired cluster can be chosen (Section 4.2).

In this thesis, we emphasize the use of clustering as a basis for design selection in topology optimization. Depending on the application and the selection criteria, different metrics can be used for clustering, some of which are discussed at the end of this chapter. Section 4.3 describes geometrical clustering, where designs are clustered based on their geometrical structure using autoencoders from literature [30]. The key contribution of this thesis is to evaluate geometric clustering on diverse TO results. Furthermore, I helped develop novel metrics to compare the topology and deformation behavior of designs. In collaboration with us, Yuanze Wu developed a novel metric to compare the topology of complex TO structures, as part of her master thesis [59]. We also proposed new methods in Section 4.4 to cluster designs based on their deformation behavior under the application of given loads. These metrics for deformation behavior are developed as part of the master's theses and resulted in conference papers by Ernest Hutapea et al. [60] and Yasuyuki Shimizu et al. [61] in collaboration with the main author of this thesis.

4.1. Data Mining

In multi-objective optimization, the decision-maker (DM) can decide on the importance of different objectives (performance attributes) through a weight-vector to generate the desired solutions [62, 63]; however, it is difficult to define the weights, especially for a large number of objectives [64]. Furthermore, it is not feasible or practical to consider every performance attribute while optimizing a design. Therefore, data mining methods may be preferred to analyze a given set of solutions.

An intuitive method is to filter designs based on attributes such as manufacturing cost or other performance attributes. For example, the dream lens tool [11] is an interactive framework that guides a designer to a set of *interesting* designs using, for example, range constraints on the performance attributes. However, this framework may not be useful if a large number of parameters have to be simultaneously considered.

More sophisticated data mining approaches can assist in identifying interesting solutions in a large database of solutions based on certain desired properties. In data mining, each solution is associated with a *feature-vector* which comprises the values of certain desired properties called *features* associated with the solution. For example, in the automotive domain, each design can be analyzed based on the drag coefficient and fuel efficiency values. In multi-objective optimization, the objective values may be considered as the *feature-vector*. Given a dataset of solutions, designs with desired *feature-vector* values can be identified using one or more of the following analysis methods: (i) descriptive statistics [16], (ii) data visualization tools such as manifold learning techniques, box plots, and parallel coordinate plots [17], or (iii) clustering methods [22, 65].

Statistical quantities describe the multivariate distribution of the features and help summarize the data. The simplest way to describe individual features is to use univariate descriptors, such as mean, median, variance, and quartiles. Higher-order statistical measures such as skewness or kurtosis contain more information about the shape of the distribution but they require a larger dataset for accurate computation. However, these measures are more difficult for engineers to interpret and identify desired solutions.

Visual descriptors are more useful in design selection when a small number of features has to be considered. The distribution of a given feature can be analyzed using histograms, box plots [66], violin plots [67], and bean plots [68] to name a few of the methods. Biplots [69] use principal component analysis (PCA) to visualize data in a 2D plot. Each of the input features is shown as directed vectors in the biplot. Bar plots or pie charts are simple methods to show the distribution of categorical variables. Scatter plots can reveal the correlation between quantitative variables whereas mosaic plots reveal the relation between the categorical features [70]. Coplots [71] use conditionals on a variable to yield subsets of solutions whose features are analyzed two at a time, using scatter plots. These methods allow visualization of the data and the selection of multiple desired solutions.

Using one or more of the methods discussed in this section, an engineer can select desired solutions. However, a class of algorithms called clustering is more useful in summarizing data in an unsupervised manner, especially when the number of features is high. Furthermore, clustering methods can be combined with other data mining methods to select desired solutions, as described in the next section.

4.2. Clustering

Classification algorithms organize data—a collection of objects—into sensible groups. Clustering is an unsupervised classification method that clusters data without using any prior information related to the classes of objects. By contrast, supervised classification models are trained using labeled data—where the classes of the objects are known—to predict the classes of unknown objects.

Clustering methods discover the underlying structures in a given data; they identify groups of *similar* objects in the data based on certain intrinsic properties. In general, the clustering algorithms require a *metric* which quantifies the distance between the objects, where a smaller distance implies that the objects are more similar. Note that metrics (e.g., Euclidean distance) should satisfy the axioms of non-negativity, null condition, symmetry, and triangle inequality. However, in practice, semi-metrics—which do not satisfy the triangle inequality—are still useful for clustering similar objects. For example, chamfer distance (CD), a semi-metric, is used to cluster 3D objects [30]. In this thesis, CD is loosely referred to as a metric for brevity. Data representation and the corresponding metric influence the clusters found in the data. These depend on the domain of data and the goals of clustering.

The clustering method used also plays a strong role since each method has its own objective and hence, its own definition for clusters. k-means method [72] tries to minimize the variance of data in each cluster, which leads to the identification of spherical clusters. Clustering using Gaussian mixture models (GMM) assumes that data contains clusters that arise from a summation of Gaussian models [73]. In contrast to these definitions, algorithms such as density-based spatial clustering of applications with noise (DBSCAN) [74] and ordering points to identify the clustering structure (OPTICS) [75] define clusters as high-density regions separated by low-density regions. Here, density in a region is inversely related to the average distance between neighbors.

Clustering can be done using certain performance features of designs, e.g., lift or drag values for cars/airplanes. If an input vector of features is used for clustering, Euclidean distance in feature space is used implicitly as the metric by most of the clustering algorithms, e.g., k-means, GMM, DBSCAN, and OPTICS. However, some of the algorithms such as DBSCAN and OPTICS, also accept an arbitrary metric for comparing designs, in which case, we provide distances from each sample to other samples as a pair-wise distance matrix.

In the following, we briefly describe k-means and OPTICS, which are primarily used for clustering in this thesis. k-means is probably the most commonly used clustering method because of its simplicity and efficiency [76] but the number of clusters needs to be pre-specified. OPTICS is a density-based algorithm that discovers the natural clusters in the data based on *density* without specifying the number of clusters. Unlike DBSCAN, OPTICS can identify clusters with different densities; in addition, it is easier to tune its hyperparameters.

4.2.1. k-means clustering

The k-means method [72] is a partitioning algorithm that identifies a given number of clusters in the data. It clusters a collection of objects, where each object has a set of d features, $\mathbf{x} = \{x^i\}_{i=0}^{d}$. The k-means method splits the data into a fixed k number of clusters, with the objective of minimizing the feature variance in the clusters. We define below the feature variance J_p in the cluster C_p of m_p objects, whose summation over the clusters gives the objective to be minimized in k-means:

$$\begin{aligned} &\sum_{p=1}^{k} J_p, \quad \text{where} \\ &J_p = \sum_{i=1}^{m_p} \|\mathbf{x}_i - \boldsymbol{\mu}_p\|^2, \quad \mathbf{x}_i \in C_p, \end{aligned} \tag{4.1}$$

$\boldsymbol{\mu}_p$ is the mean of samples $\mathbf{x}_i \in C_p$ and $\|.\|$ is the Euclidean norm.

Since the above minimization problem is NP-hard, the k-means method uses a heuristic algorithm [72] to obtain a local minimum, which tends to be close to the global minimum for well-separated clusters [77]. The main steps of the k-means algorithm can be summarized as follows:

1. Initialize a random partition with k-clusters.
2. Calculate cluster means and assign each object to its nearest cluster mean.
3. Compute new cluster means and repeat from step 2 until convergence.

k-means is a fast algorithm. Since it implicitly uses Euclidean distance as the metric, it finds spherical clusters irrespective of the data. A variation of k-means called k-medoids uses arbitrary metrics but is computationally more expensive since the distance between every pair of samples needs to be measured [78].

For contiguous data, any value of k is equally good. However, if the data contains well-separated clusters, there is an optimal choice of k. For example, we can select k that yields the smallest objective value j. Alternatively, the silhouette score can be used [79] (see Section 4.2.4).

4.2.2. Density-based clustering

Examples of density-based methods are DBSCAN [74] and OPTICS [75]. Density-based approaches identify clusters by high-density regions where the average distance between neighbors is high. While the density in the clusters needs to be specified for DBSCAN, OPTICS automatically finds the appropriate densities for the different clusters in the data.

These methods are expensive and may fail for high-dimensional data where the data tends to be sparse which makes it difficult to distinguish high- and low-density regions. Furthermore, these algorithms may identify only a single cluster depending on the data. While this information is useful, a partitioning algorithm such as k-means may be more useful for contiguous data. More sophisticated clusterings of design concepts are possible which consider multiple metrics concurrently [80].

4.2.3. Dimensionality Reduction

For high-dimensional data, dimensionality reduction (DR) techniques are needed to process the data before clustering. This alleviates the so-called *curse of high-dimensionality*, where the relative difference of the distances of the closest and farthest data points goes to zero as the dimensionality increases. So, it is challenging for clustering methods—or any other machine learning method—to process high-dimensional data. Some examples of these methods are principal component analysis (PCA) [81] and non-negative matrix factorization (NMF) [82]. PCA extracts non-redundant features using linear transformations [83, 84]. Similar to PCA, NMF is a linear transformation but is constrained to extract non-negative components, which can be more easily interpreted compared to PCA.

Manifold learning methods can be used to further reduce the dimensions to 2D or 3D without losing the cluster structure in the data. For example, t-distributed stochastic neighbor embedding (t-SNE) [85], uniform manifold approximation and projection (UMAP) [86], and self-organizing map (SOM) [87] are widely popular in the machine learning field for cluster visualization [18]. Therefore, it is easier to interactively explore the clusters and identify preferred solutions.

Empirical studies using t-SNE [85] and UMAP [86] show how data can be embedded into 2D while preserving the cluster structure. For example, image data of handwritten digits can be visualized as clusters in 2D, where images of different digits belong to a distinct cluster. UMAP preserves the intercluster distance better than t-SNE. SOM is a neural network model that maps the nearby points in the input space to nearby points in the low-dimensional output space, typically with 2 or 3 dimensions. t-SNE and UMAP are relatively inexpensive methods for visualizing clusters and design exploration.

4.2.4. Cluster Evaluation using Silhouette Score

We will now discuss a method called silhouette score [79] to evaluate a set of clusters identified using a clustering method. This will help us to choose for the given data an appropriate clustering method as well as tune its hyperparameters to obtain optimal clustering. For example, the appropriate number (k) of clusters for the k-means method can be chosen using the silhouette score [79].

Silhouette score indicates the quality of clustering; a high score means that dissimilarities between the objects within the cluster are smaller compared to the objects outside the cluster.

The silhouette sample score of an object reveals if it belongs to the interior or boundary of a cluster or the region between clusters.

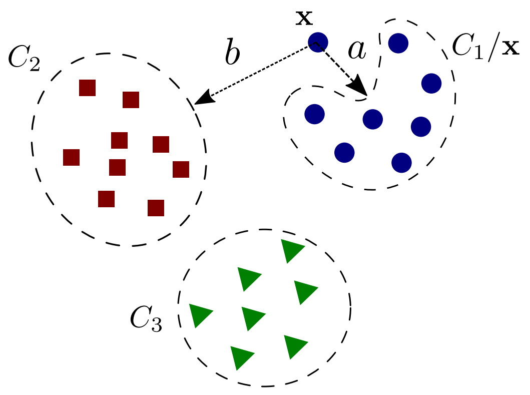


Figure 7 Silhouette score s for a sample $\mathbf{x} \in C_1$ uses a, the average distance to other samples in C_1, and b, the average distance to the nearest cluster: $s = \frac{b-a}{\max(b,a)}$.

Given a set of clusters $\{C\}_{i=0}^{k}$, obtained using a given metric, the silhouette score s of a sample $\mathbf{x}$ in C_i is a measure of its closeness to other samples in C_i, relative to samples in other clusters C_j. $D(\mathbf{x}, \mathbf{y})$ is the distance measured by the given metric between two samples x and y. Before we can define s, the following terms need to be defined. The average distance $\bar{d}$ from a sample $\mathbf{x}$ to a set S of samples is

$$\bar{d}(\mathbf{x}, S) = \text{mean}\ \{D(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} \in S\}. \tag{4.2}$$

The average distance of $\mathbf{x}$ to other solutions in its cluster is

$$a = \bar{d}(\mathbf{x}, \{\mathbf{y} \mid \mathbf{y} \in C_i/\{\mathbf{x}\}\}). \tag{4.3}$$

The average distance of $\mathbf{x}$ to its nearest cluster is

$$b = \min_j \bar{d}(\mathbf{x}, C_j). \tag{4.4}$$

Finally, the silhouette score s of $\mathbf{x}$ is given by

$$s = \frac{b-a}{\max(b,a)}. \tag{4.5}$$

The above calculation is defined only when the data contains at least two clusters. The sample score ranges from $[-1, 1]$. Figure 7 illustrates the different terms used in the definition. For samples that are between the clusters, $a \approx b \implies s \approx 0$. For samples that are well within the clusters, $a << b \implies s \approx 1$. For samples assigned to a wrong cluster, $a >> b \implies s \approx -1$.

The average score $\bar{s}$ over all the samples indicates how well the clusters are separated. When the clusters are not well-separated, i.e., the data is contiguous, the average score is expected to be low since the sample score is low for samples on the boundary of clusters.

Silhouette score can be used to find the best of a given set of clusterings [79]. For example, with the k-means method, we can vary k to obtain different clusterings. The optimal k_{opt} results in the clustering with the optimal score, i.e., the largest $\bar{s}$. For $k < k_{\text{opt}}$, $\bar{s}$ is lower than the optimal value because certain separate clusters are combined, which increases the average a value in those clusters. For $k > k_{\text{opt}}$, $\bar{s}$ is lower than the optimal value because one or more natural clusters are partitioned, which decreases the average b value in those clusters. Further, the average s for a cluster indicates how well-separated it is from other clusters.

4.2.5. Cluster Analysis and Preferred Designs

As discussed previously, clustering methods can be used to classify the designs into groups, where each group contains *similar* designs according to a *metric*. Here, we discuss methods for analyzing different clusters, which support an engineer in selecting a preferred cluster of designs.

Parallel coordinates [88] contrast the ranges of objective functions for different subsets of solutions in a single plot. Since this approach allows different scales for objectives, i.e. no normalization step, it is preferred to star plots (spider/radar plots) [89]. Radial coordinate visualization [90] represents different objectives as vertices of a polygon and the proximity of each data point to the vertices is influenced by the corresponding objective values. Other techniques that may be of interest are dimensional stacking [91] and glyph plots [92]. Using these approaches, the properties of different clusters can be contrasted.

A simpler approach for analyzing a cluster is to obtain a representative design in the cluster. For example, the cluster medoid can be used as the *cluster representative*, where medoid is defined as the solution with the least average distance to other solutions in the cluster [23]. If a cluster representative has desired properties, the complete cluster may be chosen as a set of preferred solutions. This is better than selecting a single solution as the preferred solution since it may become infeasible with emerging constraints in the design process; this is less likely to occur if a set of preferred solutions is available.

Until now, we discussed how data mining and clustering methods can be used to identify preferred solutions. For the remainder of the chapter, we discuss different metrics to compare solutions. Depending on the application, one or more of these metrics can be used for clustering and data mining.

4.3. Geometric Clustering

Clustering is a flexible method that can be tailored to an application using custom metrics. Since the geometrical structure of designs is critical for their performance, engineers might be interested in identifying distinct structures in a design database. Clustering based on geometry referred to henceforth as *geometrical clustering*, is interesting in the initial stages of product development.

In this thesis, we build two methods for geometrical clustering: (i) using a pointcloud autoencoder proposed by Achlioptas et al. [30] with our preprocessing framework for optimized designs [93], and (ii) using a novel method that compares topologies of designs. The latter is developed by Yuanze Wu [59] under the supervision of the author of this thesis. These two methods are described below.

4.3.1. Pointcloud Autoencoders

Pointcloud autoencoder (PCAE) can be used to extract *features* that are useful in clustering geometries with a similar structure. In the literature, PCAE has been used to identify everyday objects such as chairs, cars, and airplanes in public datasets [30, 94, 95]. PCAE model is trained using a set of pointclouds; each pointcloud represents the surface of a geometry using a set of 3D points. Compared to mesh or voxel representations of geometry, pointcloud representation is simpler and compact since it takes up smaller space without losing necessary geometric details. After training, PCAE can extract for each geometry a latent code, which can be used for clustering geometries. The latent code is low-dimensional compared to pointclouds. Rios et al. [96] show that PCAE can identify nonlinear subregions in the design space that are preferentially occupied by a subclass of designs. So, PCAE achieves *non-linear dimensionality reduction* of geometric data and can help in geometric clustering. Interestingly, latent code can also be used in building surrogate models for predicting the performance of car models [97], proving their use as an effective dimensionality-reduction tool.

Typically, density-based TO methods discretize the design space using a regular grid. Additional preprocessing methods are needed to convert TO results into pointclouds. Each cell in the grid can be treated as a voxel. The voxel data is converted to a surface mesh and then to *pointcloud data*. Finally, the pointcloud data of the designs are analyzed by the autoencoder [30] to yield a feature-vector, called *latent code*, which can be used to cluster designs with a similar geometrical structure.

Here, we discuss the method used in this thesis to convert voxel data to pointclouds. The voxel data of the design is a binary vector $\mathbf{x} \in \{0, 1\}^n$, where n is the number of voxels and each component of the $\mathbf{x}$: x_i corresponds to a specific voxel in the grid. If the voxel i contains material, $x_i = 1$, otherwise, $x_i = 0$. For partially occupied voxels, we assign $x_i = 1$ if the center of the voxel has material. In TO, the voxels are associated with the elements in the design space whose density is optimized; if they do not correspond, additional mapping methods need to be developed to determine if a voxel is occupied. Using the marching cubes algorithm [98, 99], the voxel data can be converted to a triangulated surface mesh. For pointcloud data, points are uniformly sampled on the triangulated surface mesh using the *trimesh* library [100, 101]. Here, the points are proportionally sampled according to the area of the triangular faces in the surface mesh.

PCAE used by Achlioptas et al. [30] consists of two stacks of neural-network layers: encoding and decoding stack (Figure 8). The encoding stack reduces the dimension of the input pointcloud and yields a so-called latent code, which the decoding stack uses to reconstruct a pointcloud similar to the input. The encoder uses five 1D convolution layers of kernel size 1, where each layer is followed by a ReLU unit and a batch normalization layer [102], whose final output is reduced by a max pool layer to a latent code of size 128 (also called bottleneck layer), which ensures that permutations of the input points for a given design will always result in the same latent code. The decoder consists of two fully connected dense layers, whose final output is reshaped into a set of 2048 3D points representing the reconstructed pointcloud. As discussed previously, the PCAE model uses the chamfer distance as the loss function. For more details, refer to the work of Achlioptas et al. [30].

Before training, the network weights used by the PCAE are randomly initialized using the Glorot uniform method [18] to ease training but the reconstruction is inaccurate as expected. By comparing the input and reconstructed pointclouds using a loss function, the autoencoder learns the network weights needed to reconstruct the input accurately. Chamfer distance (CD) and earth mover distance (EMD) are exemplary metrics used to measure the differences between any two pointclouds [30]. Since the latent code is restricted to having fewer dimensions than the input, PCAE reduces the dimensionality of the input.

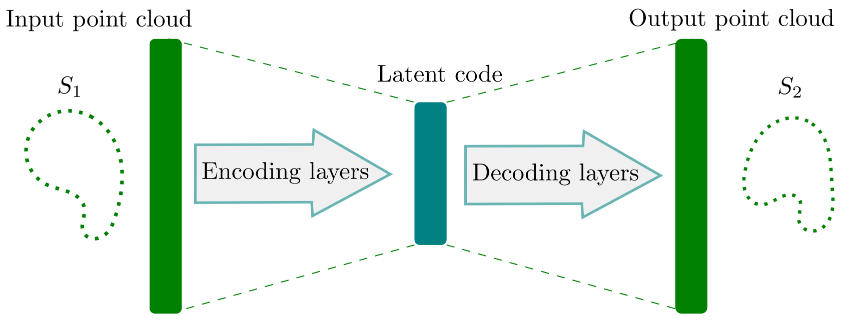


Figure 8 Schematic representation of PCAE.

After training with an input set of pointclouds, the autoencoder can transform each design into a latent code. Using clustering methods such as k-means or OPTICS, structures can be grouped. The effectiveness of this method for geometrical clustering of TO results is evaluated in Chapter 7 using the methods described in Chapter 5. Since this method concurrently analyzes the topology, size, and shape of the designs, we introduce next a metric that can better analyze the topological differences exclusively.

4.3.2. Topology Clustering

Comparing the topology of designs provides distinct advantages over metrics based on voxel or pointcloud differences discussed in the previous section. Topology captures the connectivity between different components of the structure, which may provide insights into the load paths

[28, 29, 103]. Light-weight designs tend to be composed of beam-like components which can be better analyzed by describing the topology of the design. In this section, we describe the skeletonization method to obtain for a given design a skeleton, which is a low-dimensional interpretation of the high-dimensional material layout.

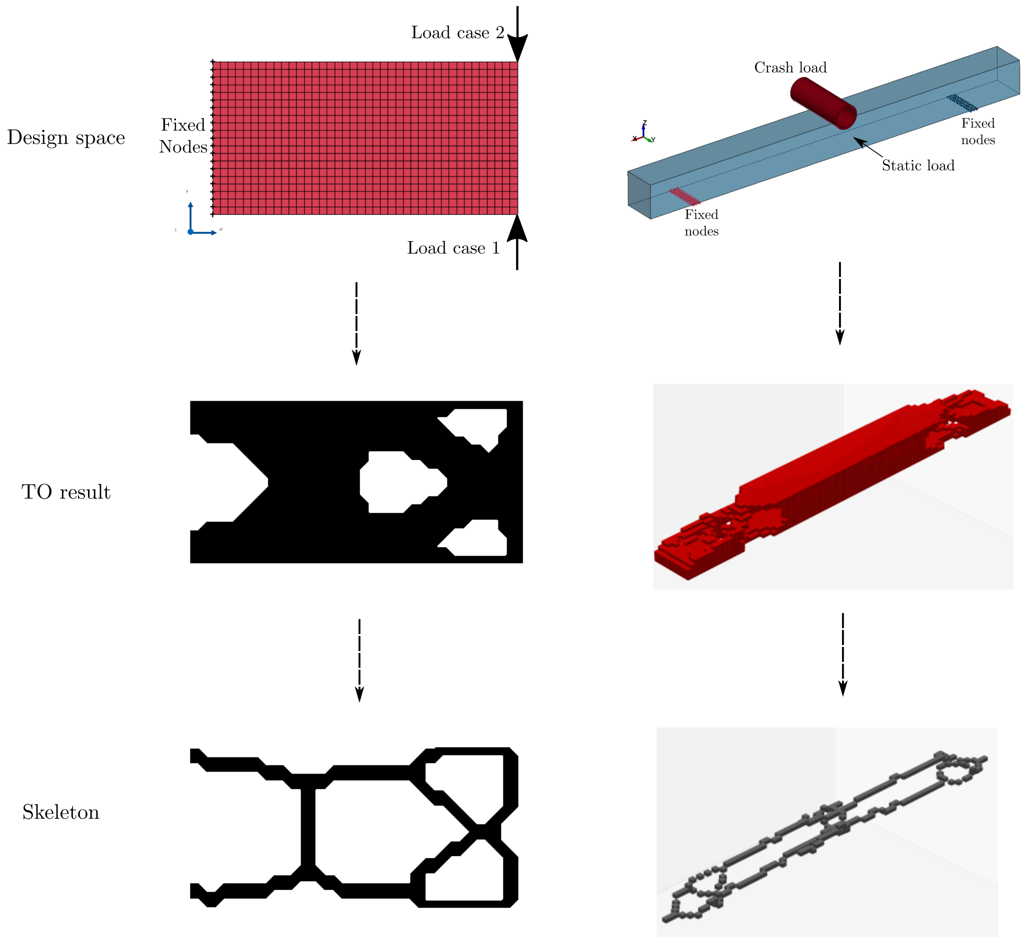


Figure 9 Skeletonized TO results obtained using SEW-HCA. The first row shows the first two problems discussed in Section 3.3. For each problem, we show in the second row the optimized design obtained using SEW-HCA with equal weights on the corresponding loads. In the last row, we show the skeletons, whose thickness is equal to that of a voxel.

Skeletonization methods [104–106] extract a skeleton from a design while preserving the topology. While 2D objects yield a curve skeleton, 3D objects can yield either a curve skeleton or a surface skeleton. For 3D objects—such as the TO results—Lee et al. [107] propose an iterative thinning process that deletes the border points while satisfying the topological and geometrical constraints, i.e., preserving the number of connected parts, cavities, and holes in the original design. We use Scikit-image Python library [99] to convert 3D voxel data to voxel skeletons using the method proposed by Lee et al. [107]: each skeleton is represented by a set of voxels. For example, see Figure 9, which shows optimized designs for different load configurations and the resultant skeletons.

Voxel skeletons can be easily converted to pointclouds by identifying the center of each voxel as a 3D point. As discussed previously, pointclouds can be compared using CD or EMD. Using these metrics, pair-wise distance matrices can be generated and used for clustering similar skeletons using methods such as k-medoids or OPTICS.

4.4. Deformation Clustering

The deformation behavior of a design is yet another interesting property for an engineer. Here, deformation behavior refers to the response of a structure to a given set of loads constrained to certain boundary conditions. Skylar et al. [108] parametrize the nonlinear plastic deformation of geometries, which is used to identify similar simulation results. However, their method analyzes the surface deformation but not the volumetric deformation, which may involve more complex phenomena. Furthermore, we need methods that can analyze TO results, where designs can have different material layouts and nodes.

In this section, we first discuss the use of dimensionality reduction techniques to analyze volumetric deformation. Later, we describe a method to analyze the evolution of deformation as time series (sequential) data. The main contribution is to evaluate and adapt these state-of-the-art methods to the deformation behavior observed in structural mechanics.

4.4.1. Dimensionality Reduction

As a first step, we regard the use of manifold learning techniques such as t-SNE [85] and UMAP [86] to analyze the nodal displacements for a single time step. We obtain reasonable results for the dataset discussed in Section 3.3.3, which are partially published in [60].

Since TO results tend to have different material distributions, it is necessary to identify common nodes before the dimensionality reduction techniques can be used. In our dataset (Section 3.3.3), SIMP is used to optimize the densities of elements in the design space. The relative density is allowed to vary from a minimum value of 0.05 to 1. So, no elements are removed and all the TO results have the same nodes and the dimensionality reduction methods can be directly applied to the nodal displacement data.

Figure 10 shows the clusters in the dataset. UMAP is used to reduce the displacement data of each design into a 2D point. k-means method is used to cluster the UMAP coordinates. The number of clusters $k = 18$ is chosen since it yields the highest silhouette score (Section 4.2.4). Note that the clusters are arbitrarily numbered. The coordinate values in a UMAP visualization do not have any significance. However, the relative distance between points indicates data similarity [86]. Figure 11 shows a sample set of designs in a cluster. Qualitatively, we can see that the deformation pattern is similar for all the designs. The designs tend to have the largest deformation in the foremost corner.

Garcke et al. [109] recommend clustering as a preprocessing step before analyzing the volu-

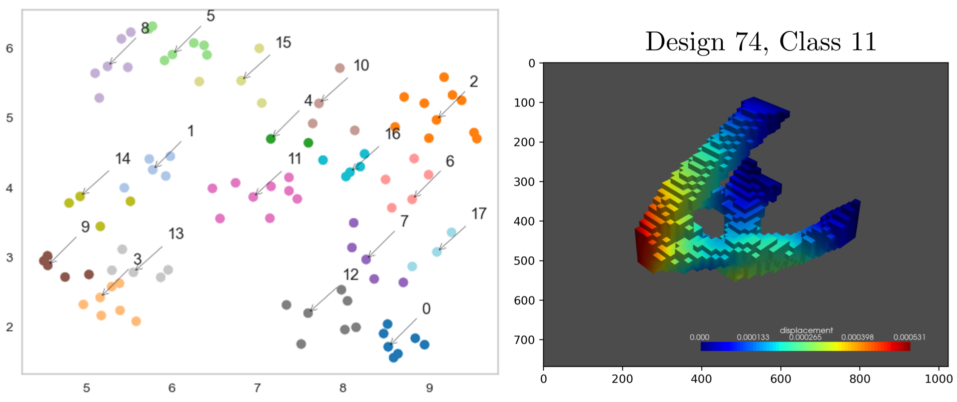
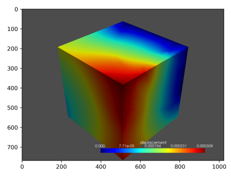
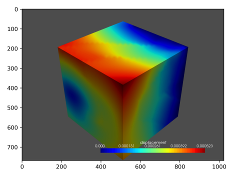
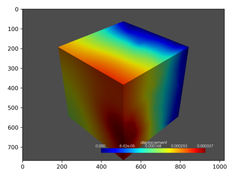
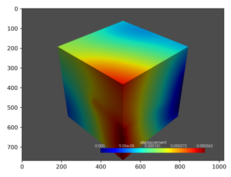


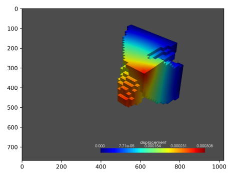
Figure 10 TO designs are mapped to 2-D using UMAP components, where each point represents a design. The data samples are colored according to the clusters obtained by using the k-means method on the reduced UMAP data. A sample design in cluster/class 11 is shown to the right.

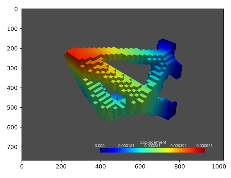

(a) Design 36

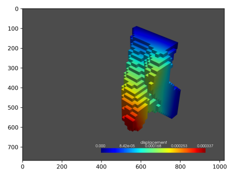

(b) Design 43


(c) Design 95


(d) Design 97


(e) Design 36


(f) Design 43


(g) Design 95

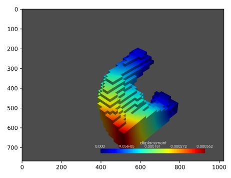

(h) Design 97

Figure 11 Different designs from cluster 1. The top row shows the design space with the elements colored according to the displacement magnitude. The bottom row shows the underlying design, i.e., only the elements with high-density elements.

metric deformation in finite element (FE) simulations. They cluster the FE nodes with similar displacement, followed by an analysis of each cluster of nodes using dimensionality reduction techniques such as PCA [81] and diffusion maps [110]. Therefore, it would be interesting to research in the future if the preprocessing step has an effect with t-SNE and UMAP as the reduction methods. Further research is needed to select the appropriate nonlinear reduction method for displacement data.

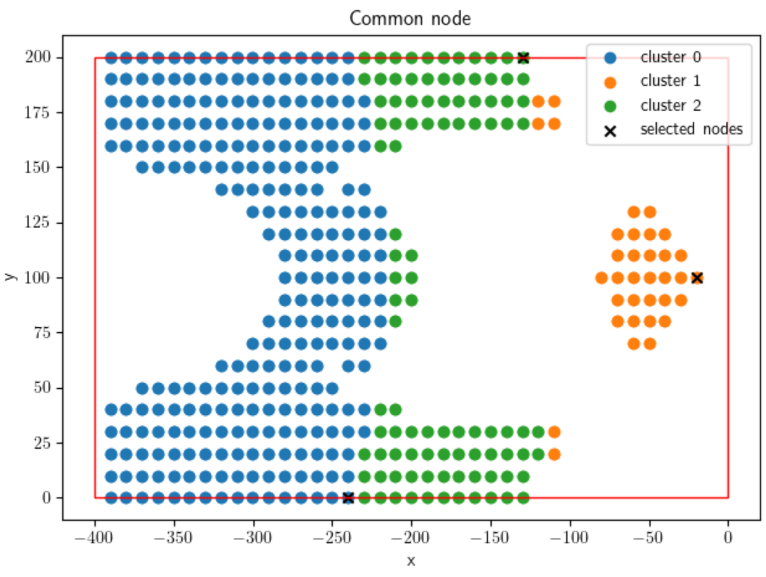
4.4.2. Time-series Analysis

As mentioned previously, dimensionality reduction methods are pivotal in analyzing high-dimensional displacement data. Furthermore, nodal displacements can involve multiple time steps and time series analysis may be needed. Given these challenges, we propose an alternative approach for analyzing volumetric deformation in TO results. First, we identify the common nodes among the designs. Depending on the application, the nodes to be analyzed could be further reduced, e.g., by choosing the node near the center of mass or nodes on the boundary for analysis. When the number of nodes is large, clustering based on displacement magnitude can be used to group nodes, and the cluster medoids can be chosen for analysis. In the second step, the displacement time series of the nodes are compared using state-of-the-art metrics such as dynamic time warping (DTW) [111].

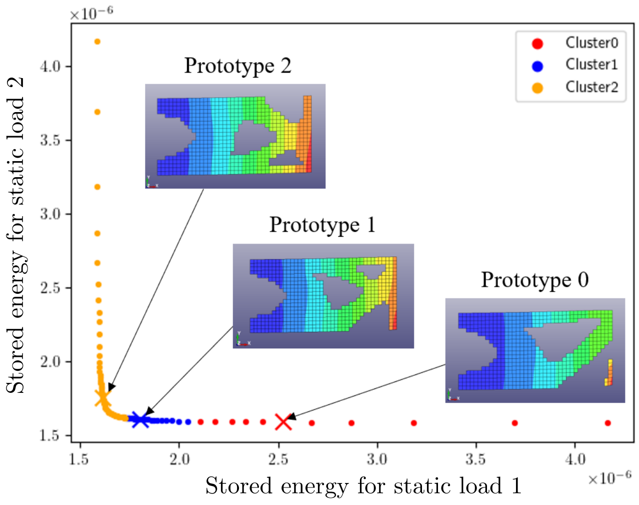
DTW is a metric used to compare multi-dimensional time series data. The displacement of each common node n_i is a 3D time series denoted as $\mathbf{d}_i$. For a pair of time series samples $(\mathbf{d}_i, \mathbf{d}_j)$, DTW warps the temporal sequences nonlinearly in the time dimension. Loosely speaking, this allows the DTW to ignore phase shifts. Given a set of TO results and their common nodes, each TO result has a distinct deformation subject to the given load configuration. So, each design will assign a distinct temporal sequence to each of the common nodes: $D_I = \{\mathbf{d}_I^i\}_{i=1}^N$, where N is the number of common nodes and I is the identification number for the design. So, given two designs and their corresponding deformation behavior D_I and D_J, the dissimilarity can be defined using DTW as follows:

$$\sum_{i=1}^{N} \mathrm{DTW}(\mathbf{d}_i^I, \mathbf{d}_i^J). \tag{4.6}$$

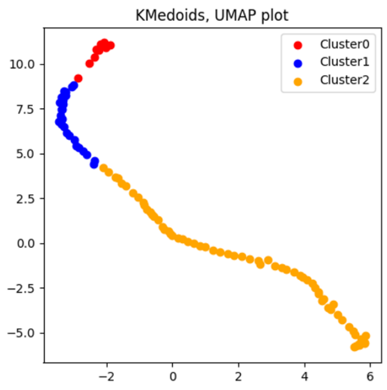
We validated the above method using the dataset-2 described in Section 3.3 and the results are partially published in [61]. Figure 12a shows the common nodes, which are clustered based on the displacement magnitude using the k-means method. $k = 3$ is arbitrarily chosen, which results in 3 selected nodes. Since dataset-2 is optimized for two static loads, the Pareto front is two-dimensional, as shown in Figure 12b. For different designs, the deformation behavior under the static load 1 is compared using the metric defined in Equation (4.6). The metric can be used to calculate the pair-wise distance matrix, which can be used to cluster deformation behavior. k-medoids with $k = 3$ is used to cluster the dataset-2 (Figure 12b). Finally, UMAP can be used with the pair-wise distance matrix to reduce each deformation into a 2D point. The UMAP visualization of the three contiguous clusters is shown in Figure 12c. Since we expect the deformation pattern to gradually change, the clusters identified seem to be qualitatively correct. However, further research with engineering datasets is needed to evaluate our proposed method.



(a) Common nodes



(b) Pareto front



(c) UMAP visualization

Figure 12 Deformation behavior analysis of cantilever plate dataset.

Summary

In this chapter, we discussed different knowledge generation methods for TO results, with emphasis on using clustering to group *similar* designs. Multiple clustering methods, as well as cluster analysis methods, are briefly described. Since the clustering method requires a metric to measure the dissimilarity between designs, we develop different metrics for clustering in this thesis. We discussed the metrics for distinguishing the geometrical structure as well as the deformation behavior. Since the literature is lacking on the use of metrics of geometry with TO results, we discuss next the evaluation methods for different metrics of geometry while also validating the use of PCAE with pointcloud data.

5. Metric Comparison Method for Geometric Feature Vectors

At the early stages of product development, the differences in geometric properties such as size, shape, and topology can be as important as the performance requirements. Different metrics of geometry can be used for geometric clustering, i.e., cluster designs with similar geometrical structures, which provides insights into the material distribution of different designs. Furthermore, the metrics can be integrated with the similarity-controlled optimization methods [19, 112] to yield designs similar to a reference design that is chosen because of economic reasons or manufacturing limitations.

In this chapter, we discuss numerous metrics of geometry from literature with varying degrees of accuracy and computational complexity. Furthermore, we use existing dimensionality techniques on voxel representation to yield simple metrics of geometry. Finally, we present a novel method to validate a given metric with desirable properties by comparing it with a metric that is accurate but is not preferable, e.g., because of its computational cost. The metric comparison methods discussed in this chapter have been partially published by the author of this thesis [93, 113].

While numerous metrics of geometry exist, some of the metrics assign each design with a *feature-vector* obtained through processing the geometric data that encapsulates the material distribution. The *feature-vector*, hereafter referred to as a *geometric feature-vector* ($\mathbf{g}$), can be directly used for geometrical clustering. If the Euclidean distance in the space of $\mathbf{g}$ is a meaningful metric of geometry, the k-means method can be used inexpensively for clustering. Furthermore, it simplifies the building of a metamodel to judge if a design is desirable based on geometry, which will form the basis of our cooperative optimization framework (Chapter 6).

Due to the high-dimensionality of 3D geometric data, any metric can only compare a few of the geometric properties, e.g., hand-crafted metrics may use the distribution of surface curvature, mass, or spectral descriptors [114]. By contrast, data-centric methods—which are more successful in practice—extract features relevant to a specific 3D geometric dataset. For example, from voxel data of geometries, different $\mathbf{g}$ vectors can be extracted using dimensionality reduction techniques such as PCA [81], NMF [82], t-SNE [85], or UMAP [86]. Neural network models called autoencoders can also reduce high-dimensional data: for example, Qi et al. [94] use an autoencoder to learn features from the pointcloud data which is obtained by sampling points on the surface of the design. However, it is not clear if a given feature-vector ($\mathbf{g}$) and the corresponding Euclidean metric is as meaningful as a *reference metric* which is a baseline metric that is demonstrated in the literature to accurately identify different classes of objects based on geometry, e.g., chamfer distance for pointclouds [94]. So, we present a method to evaluate novel metric systems with different $\mathbf{g}$ using the *reference* metrics.

We evaluate different metrics of geometry based on the following criteria:

- Metrics should be sensitive to differences in size, shape, topology, position, and orientation of a design. Invariance to rotation, reflection, and translation operations is an advantage in 3D design classification, but not for TO designs, where the configuration of a design relative to the boundary conditions is important.
- Metrics that associate each design with a feature-vector $\mathbf{g}$ are preferred, where the Euclidean distance in the space of $\mathbf{g}$ is meaningful. This allows the use of k-means clustering which is computationally inexpensive and widely used in machine learning [22].
- Metrics should be able to cluster designs with similar geometrical structures, even for datasets with topologically complex designs.

In what follows, Section 5.1 presents exemplary voxel and pointcloud data of 3D geometries, followed by a few intuitive *reference* metrics of geometry from literature (Section 5.2). Several of these metrics do not fulfill the above requirements. So, Section 5.3 introduces appropriate dimensionality reduction techniques to develop new metrics that yield $\mathbf{g}$ from the voxel and pointcloud data. In Section 5.4, we propose a new method for comparing reference metrics with the Euclidean metric that uses $\mathbf{g}$ as input. Besides this, we need to evaluate the metrics using exemplary data, which will be presented later in Chapter 7.

5.1. Geometric Data for Metrics Evaluation

The type of geometric data extracted from a design determines the applicable metrics and feature extraction methods. Figure 13 shows the underlying representations of voxel and pointcloud data used in this study. The former is easily available in TO since the design domain is generally already discretized for the optimization into voxels [1], while the latter is a compact and expressive datatype popular in 3D object recognition, classification, and segmentation, e.g., [30, 94]. Both data types as understood in this thesis are defined in the following.

Voxel data: A cuboidal domain containing the design is discretized into voxels using a regular grid. The voxel data of the design is a binary vector $\mathbf{x} \in \{0, 1\}^n$, where n is the number of voxels and each component of the $\mathbf{x}$: x_i corresponds to a specific voxel in the grid. If the design occupies the voxel i, $x_i = 1$, otherwise, $x_i = 0$. For partially occupied voxels, we assign $x_i = 1$ if the design occupies the center of the voxel. The dimension of $\mathbf{x}$ can be large, depending on the grid resolution required to capture the complexity of the design. When the voxel grids of two designs do not match, a common space containing them is discretized into voxels, where each voxel may be marked as occupied if a majority of the voxel is occupied according to the original grid. In this thesis, we ensure that the underlying voxel grid is common among the designs.

Voxel data can be extracted for TO results since, in general, TO optimizes material in a voxelized domain [1]. An interesting research question that we address later is how well the dimensionality reduction techniques, popular in machine learning, can identify underlying design patterns and extract **g** from voxel data of designs.

Pointcloud data: Compared to voxels, a pointcloud is a compact representation that uses sampled points on the surface to represent the design. Geometric learning methods such as autoencoders with pointclouds as the input are interesting since they can identify different classes of shapes such as cars and airplanes [30, 94].

Octrees and meshes, among numerous other representations [115], can also be used to represent 3D data. The octree-based representation [116] alleviates the high memory usage of uniform voxel size by varying the resolution of voxels in different regions, e.g., by using a higher resolution near the surface of a design. Surface meshes use a set of polygon faces to represent the surface of a geometry and are more widely used in object recognition [117] than volumetric meshes, which are more predominant in finite element analysis but are high-dimensional [118]. Since the dimensionality of pointclouds is very low, it is increasingly used by researchers in 3D object recognition [30]. For our initial analysis of TO results, we consider the metrics using voxel data for their simplicity and pointcloud data for their usefulness in object recognition. More sophisticated autoencoders using pointcloud data with surface normals [119] or 3D mesh data [120] are possible and can be the subject of future research.

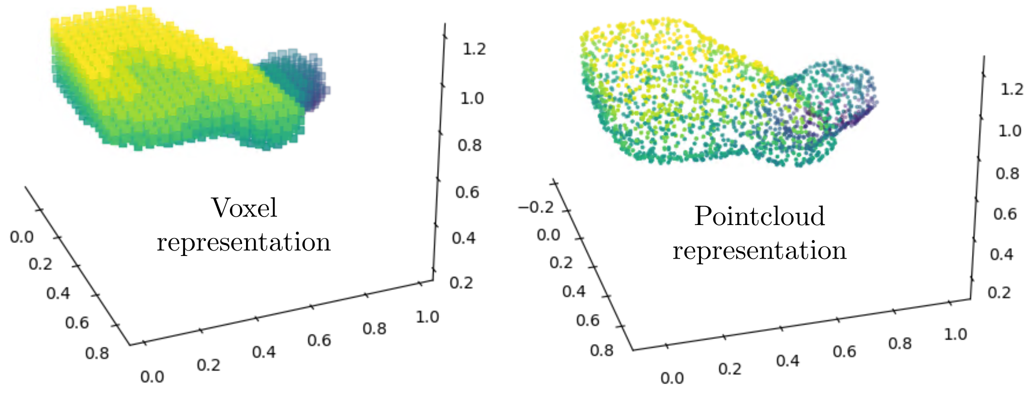


Figure 13 Geometric representations of TO results

5.2. Reference Metrics

In this section, we describe a few intuitive metrics used in the literature to quantify geometrical differences between designs. These metrics serve as a reference to evaluate the **g** described later. For the sake of clarity and brevity, we choose a few popular and reference metrics from the literature, although there are other variants and methods to define metrics.

Voxel distance (VD): Voxel data of two designs can be compared directly when they correspond to the same regular grid in the 3D domain. The Euclidean distance between the voxel data of the two designs is equal to the square root of the number of *non-overlapping* voxels that are only occupied by one of the two designs. A disadvantage of this metric is that it is insensitive to the position of non-overlapping voxels. For example, given any two designs without any overlap, the metric is invariant to their relative position as long as the designs do not overlap.

Chamfer distance (CD): Normally, chamfer distance is used to compare pointcloud data of geometries [30]. CD between two pointclouds, say $S_1, S_2 \subset \mathbb{R}^3$, is defined as follows:

$$\mathrm{CD}(S_1, S_2) = \sum_{\mathbf{a} \in S_1} \min_{\mathbf{b} \in S_2} \|\mathbf{a} - \mathbf{b}\|_2^2 + \sum_{\mathbf{a} \in S_2} \min_{\mathbf{b} \in S_1} \|\mathbf{a} - \mathbf{b}\|_2^2, \tag{5.1}$$

where **a, b** $\in \mathbb{R}^3$ are points in the pointcloud.

Earth mover distance (EMD): Similar to CD, EMD [30] or Wasserstein metric [121] compares the pointcloud data $S_1, S_2 \subset \mathbb{R}^3$ of two designs. It solves an optimization problem to find a mapping $\Psi : S_1 \to S_2$ such that the objective $\sum_{\mathbf{a} \in S_1} \|\mathbf{a} - \Psi(\mathbf{a})\|_2$ is minimized. We use an approximate but fast algorithm for EMD calculation proposed by Achlioptas et al. [30]. EMD is still much slower than CD.

The metrics—VD, CD, and EMD—are sensitive to changes in rotation, translation, or reflection. While VD compares very high-dimensional voxel data, CD and EMD compare the pointcloud data. These metrics are sufficient since we only need a few reference metrics to demonstrate our metric evaluation method. As discussed previously, the emphasis of this study is to evaluate **g**, the benefits of which will be discussed next.

5.3. Dimensionality Reduction

Dimensionality reduction techniques identify the underlying patterns in a dataset and summarize them in a lower-dimensional feature space. Each dimensionality reduction method can extract features from the high-dimensional 3D geometric data, yielding a new set of **g**. In this thesis, we investigate the following dimensionality reduction methods to extract **g** from voxel data: PCA [84], NMF [82], t-SNE [85], and UMAP [86] which are widely used in different fields. PCA extracts non-redundant features using linear transformations [83, 84]. Similar to PCA, NMF is a linear transformation but is constrained to extract non-negative components, which can be more easily interpreted compared to PCA. t-SNE and UMAP are nonlinear manifold learning methods used in machine learning to visualize high-dimensional data. Note that voxel data can itself be used as **g** without any reduction since the Euclidean distance leads to a meaningful metric called voxel distance. However, Euclidean distance with pointcloud data is not meaningful since it is sensitive to the order of points in each pointcloud. For example, given two pointclouds of the same size, the Euclidean distance between them is not unique since it will change with the order of points in either of the pointclouds. While it is not clear if PCA, NMF, t-SNE, and UMAP can be used for pointclouds; autoencoder is an effective feature extractor with pointcloud data as demonstrated by its use in object classification [30, 94]. The main aspects of these methods are given in the following. For details, see the references mentioned in the text.

Principal component analysis: PCA analyzes a set of n data points each with d features, $X_{n,d} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^d,\ i = 1, ..., n\}$, to linearly project each sample $\mathbf{x}_i$ to a new set of orthonormal basis vectors: $\{\mathbf{b}_i \mid \mathbf{b}_i \in \mathbb{R}^d,\ i = 1, ..., \min(n, d)\}$, i.e., $\mathbf{x}_i = \sum p_k \mathbf{b}_k$. The new basis is constructed such that the principal component scores p_k are uncorrelated and the variance in the components p_i decreases as the order i increases [122]. It is often sufficient to consider only a few initial principal components to represent the data, resulting in dimensionality reduction. For voxel data, n is the number of designs and d is the number of voxels.

Non-negative matrix factorization: NMF [123] factorizes the input matrix $\mathrm{X}_{d\times n} = [\mathbf{x}_i]_{i=1}^{n} = \mathrm{W}_{d\times r}\,\mathrm{H}_{r\times n}$ where d is the number of features, n is the number of data samples; W and H are matrices with non-negative entries. In general, $r \ll \min(n, d)$ such that each sample $\mathbf{x}_i$ (column i of X) can be expressed as a linear combination of columns of W, i.e., $\mathbf{x}_i = \mathrm{W}\mathbf{h}_i$, where $\mathbf{h}_i$, a r-dimensional column vector of H, is the reduced representation of $\mathbf{x}_i$. Since H has non-negative entries, voxel data is decomposed into interpretable components.

t-distributed stochastic neighbor embedding: t-SNE embeds the high-dimensional data in 2D or 3D such that the clusters in the data become visible [85]. It achieves this by keeping data samples, that are close as per a given metric, i.e., close in the reduced coordinates with high probability. So, t-SNE is a nonlinear transformation that preserves the cluster structure.

Uniform manifold approximation and projection: UMAP, similar to t-SNE, is a manifold learning technique, useful in visualizing high-dimensional data [86]. UMAP preserves the inter-cluster distance better than t-SNE. Empirical studies using t-SNE [85] and UMAP [86] show how data can be embedded into 2D while preserving the cluster structure. For example, image data of handwritten digits can be visualized as clusters in 2D, where images of different digits belong to a distinct cluster.

Pointcloud autoencoder: As discussed in Section 4.3.1, PCAE is a nonlinear dimensionality reduction method that uses deep neural-networks. PCAE learns to reconstruct the input pointcloud from a low-dimensional latent code. It is demonstrated later in Chapter 7.2 that PCAE with CD loss function provides meaningful $\mathbf{g}$ vectors that can be used to cluster similar TO results.

Autoencoders were effectively used with pointclouds to extract features from everyday objects such as chairs, cars, and airplanes [30, 94, 95]. Rios et al. [96] demonstrate that a pointcloud autoencoder identifies nonlinear subregions that are preferentially occupied by a subclass of designs. This explains the usefulness of the latent code in object classification. In this thesis, we use the PCAE architecture proposed by Achlioptas et al. [30] since it is simple, effective, and has no additional assumptions on the shape.

Euclidean distance with $\mathbf{g}$ as input, when meaningful, allows the use of k-means clustering (Chapter 4.2.1) as well as simplify the process of building metamodels that can predict the geometric properties without actually running an expensive design generation method. Since it is not clear if the Euclidean distance with $\mathbf{g}$ obtained using different methods is a valid metric, we propose a method to compare it with reference metrics. The proposed method can also be used to compare any new metric with the reference metric. Furthermore, we introduce measures to evaluate the clustering accuracy of a given metric.

5.4. Methods for Evaluating Metrics

Until now, we discussed existing metrics of geometry and dimensionality reduction methods from the literature. In this section, we propose two general methods to validate a new metric, referred to as the target metric (TM) hereafter, by comparing it with a reference metric (RM), which is a baseline metric known to be useful in capturing geometrical differences.

Reference metric (RM): For example, chamfer distance or other metrics from Section 5.2 may be chosen as the RM. For a given dataset, other metrics may also be used as RM. For example, if the designs only differ in a single geometric property θ, we can also use the difference in θ as RM. For more complex cases, we need more advanced approaches using more than a single property. Hence, we are looking for an evaluation method that can handle both of these cases.

Target metric (TM): Similarly, a dimensionality reduction method from Section 5.3, say PCA, can be chosen to yield a vector $\mathbf{g}$ and the Euclidean distance in $\mathbf{g}$ is a TM to be validated. Since each dimensionality reduction method results in a different $\mathbf{g}$, we have multiple TMs to validate.

In summary, any proven metric can be used as RM and the metric to be tested is referred to as TM. Next, we describe the two proposed methods for comparing metrics in the thesis: the first method measures the correlation between distances measured by TM and RM while the second method compares the clustering performance of TM and RM for one or more datasets. If TM and RM are found to be similar, then we can safely use TM instead of RM if it has better properties, e.g., when it uses Euclidean distance with some vector $\mathbf{g}$.

5.4.1. Comparison with a Reference Metric

As discussed before, the first method compares a TM and the baseline metric called RM based on the distances measured. Our method is based on the following idea: If a given TM is a valid metric, the distances measured by TM should be *similar* to that of RM. So, a high correlation between the distances measured by TM and RM indicates that TM is a valid metric since RM is known to be valid.

Consider a collection of N geometries $\mathbf{G} = \{G_e\}_{e=1}^{N}$ and its pairs $\mathbf{P} = \{p_i = (G_m, G_n) \mid m \neq n\}$. A metric M measures the distance between G_m, G_n of a pair p_i and the distances measured are given by $D_{\mathrm{M}} = \{\mathrm{M}(p_i) \mid p_i \in \mathbf{P}\}$. The similarity between RM and TM is given by the correlation between D_{RM} and D_{TM}, where a high correlation value implies a high degree of similarity. Figure 14 illustrates the proposed method.

Correlation measures: Pearson correlation (ρ_p) [124] calculates the linear correlation between any two input variables, which are $D_{\mathrm{RM}}, D_{\mathrm{TM}}$ in our case. ρ_p ranges from -1 to $+1$. For a positive linear correlation, $\rho_p \approx 1$. In general, the values of $D_{\mathrm{RM}} and D_{\mathrm{TM}}$ may not be linearly related. However, it is enough for the measured distances to have a monotonic relation if the metrics are similar. To this end, we need to measure Spearman correlation (ρ_s) [124] rather than Pearson correlation (ρ_p) to measure the metric correlation. Spearman correlation measures the Pearson correlation of rank values of the given data, which are obtained by ordering all the values and assigning a rank to each value based on their position in the order.

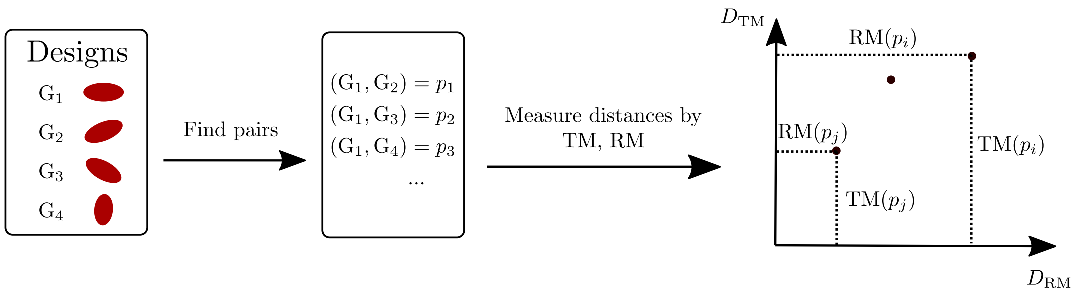


Figure 14 Measuring similarity between two metrics RM and TM. For each pair of geometries, RM and TM yield two different distance values. A high correlation between the measured distances D_{RM} and D_{TM} validates the TM.

Spearman correlation then uses the rank values to obtain a Pearson correlation coefficient, which is useful in measuring the monotonic relationship, whether linear or not, between the given two variables.

In this thesis, when the correlation between the measured distances is high, i.e., $\rho \approx 1$, we say that the *metric correlation is high*. Note that the meaningfulness of a correlation depends on the number of points. So, the significance of the correlation may need to be measured if the dataset contains very few samples.

The proposed method can empirically determine if a TM is at least as good as the RM in quantifying geometrical differences. When the metric correlation is high, we expect TM to yield similar clusters as RM. However, we verify this by evaluating the clustering accuracy with labeled test datasets, using the methods discussed in the next section.

5.4.2. Comparison based on clustering performance

One of our objectives is to inexpensively identify groups/clusters of designs with similar geometrical structure. So, we need to measure the clustering performance of different metrics from Section 5.2 and 5.3. In this thesis, we use test datasets where the expected subclasses are known and the metrics are challenged to identify these subclasses. In these labeled datasets, designs belonging to a subclass i have a common ground-truth label g_i. By clustering the dataset using a metric, we can identify the clusters in the dataset where designs in each cluster i have a common cluster label c_i. Clustering performance can be evaluated by comparing the ground-truth and cluster labels using classification errors such as precision, recall, and F1-score [18, 125] as described below.

Label remapping: In general, due to the stochastic nature of clustering algorithms, the cluster labels and the ground-truth labels are not the same, even if the clustering is accurate in finding the subclasses (Figure 15). So, before using the classification measures, the cluster labels need to be remapped to match the ground-truth labels. Here, we propose a simple method for relabeling cluster labels. In practice, when the clustering is not perfect,

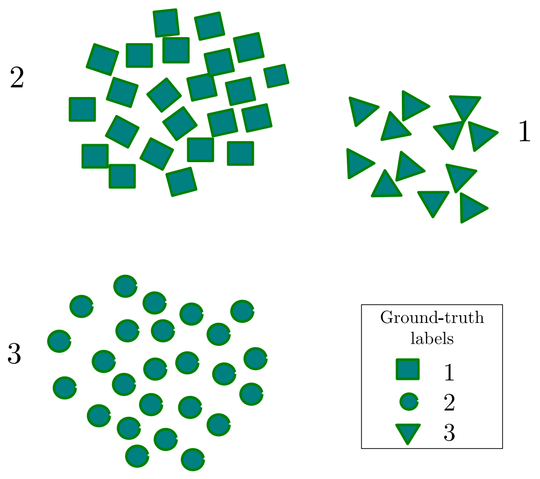


Figure 15 An example where clustering identifies the subclasses accurately. However, the cluster labels (shown beside clusters) are different from the corresponding ground-truth labels. For this example, the majority label method would relabel the cluster label 2 to 1 since a majority of its samples have the ground-truth label 2.

designs in a cluster i may not have a common ground-truth label but we can set the cluster label c_i to the ground-truth label g_i used by the *majority* of samples in the cluster i (Figure 15). After relabeling, measures of classification error can be used to measure the clustering performance.

Here, we provide brief definitions for precision, recall, and F1-score. For each cluster i, precision is the fraction of samples that have the ground-truth label g_i among the total samples that have the cluster label $c_i = g_i$. Recall measure is the fraction of samples that have the cluster label c_i among the total samples with the ground-truth label $g_i = c_i$. F1-score is the harmonic mean of precision and recall. Since both high precision and high recall are desired, the F1-score is more useful than precision and recall. Weighing each of the scores (say, F1-score) with the cluster size gives a corresponding average score (i.e., average F1-score).

In addition to the classification errors, one can use adjusted mutual information score (AMI) [126], a state-of-the-art method to measure multi-label classification accuracy without the relabeling step mentioned above. AMI is invariant to permutations of the labels and is adjusted for the chance, i.e., random labeling gives a zero AMI score. However, AMI is not as intuitive as the classification error measures using the majority label method discussed previously. Furthermore, by analyzing the classification errors for each cluster, one can better identify the mislabeled designs.

Summary

In this Chapter, we presented numerous metrics for identifying geometrical differences using voxel and pointcloud representations. Furthermore, we discussed a method for evaluating a new metric with respect to a *proven* reference metric. We also presented clustering performance as a method to evaluate metrics quantitatively using labeled test datasets. Since the benchmark datasets for computer vision such as ShapeNet [127] are topologically not as complex as the TO results, later in Chapter 7, we discuss methods to generate test datasets with complex topologies as well as evaluate the Euclidean metric based on different geometric feature-vectors using our datasets.

6. Cooperative Topology Optimization

In engineering, TO requires *expensive* simulations to find structures in a *high-dimensional* design space while considering *multiple objectives*. MTO methods can potentially yield an unlimited number of Pareto optima. Therefore, some of the challenges in MTO are:

1. How to generate diverse Pareto optima?
2. How to incorporate user feedback and generate desired solutions?
3. How to avoid expensive simulations while addressing the above challenges?

MOEAs can generate diverse optimal solutions as well as incorporate user feedback [25]. When the objectives are expensive to evaluate, as in TO, it is economical to use the so-called *interactive* methods [21], which progressively generate only the preferred solutions. However, the application of MOEA in the high-dimensional decision space of MTO problems is prohibitively expensive. The dimensionality of such a decision space can be reduced using feature mapping methods such as the evolutionary level-set method [3, 26]. However, even if the dimensionality is reduced, MOEAs still require a large number of objective evaluations and hence are computationally expensive. If we can reduce the number of expensive evaluations, we can use MOEAs to generate diverse, desired solutions.

State-of-the-art TO methods iteratively optimize the material by efficiently analyzing the effect of each decision variable (called sensitivities) on the given objective in each iteration (Chapter 3). Examples of these methods are gradient-based SIMP [1] and heuristic-based HCA [39]. At each iteration, one can combine the effect of multiple objectives using the weighted-sum method for MTO (w-MTO) to develop efficient algorithms such as SEW-HCA [27], where the user preferences are included by an appropriate weighting of the objectives. For more details on SEW-HCA, see Chapter 3. However, it is difficult to articulate the preferences for the objectives without analyzing a representative set of solutions.

In this chapter, we propose the use of the *interactive* method, as proposed by Miettinen et al. [21], to incorporate user preferences and generate preferred solutions. In this method, a given set of solutions is analyzed and the user modifies her/his preference to generate a new set of solutions; this process is repeated until the user is satisfied. In the literature, preferred solutions are defined using reference directions, points, or other supervised methods [25]. In Chapter 4, we discussed the use of clustering methods to support an engineer in the identification of preferred solutions. In this chapter, we develop a novel cooperative algorithm that uses a preferred cluster of solutions as the input and generates only the preferred solutions. For this purpose, we combine an EA with metamodeling techniques to predict weights required by a w-MTO to generate the preferred set of solutions. The method is referred to as iSMO since it uses a set of solutions as a reference. For a given set of weights, iSMO uses inexpensive machine learning models to predict the objectives and therefore allows one to decide if a set of weights will result in a preferred solution. Although we use weights as the input to iSMO in this thesis, it may be generalized to handle other hyperparameters of a design optimization method.

In the next part, Section 6.1, we introduce a novel cooperative optimization method called iSMO. In Section 6.2, we describe evaluation measures for the new solutions generated by iSMO: These measures evaluate the new solutions based on their *similarity* to the preferred set as well as their *diversity*. While these methods are taken from literature, we improve here the diversity measure by using normalization. Later in Chapter 8, we will present corresponding test problems and the results.

6.1. Interactive Set-based Multi-objective Optimization

As discussed in Chapter 4, the preferred set of solutions $\mathbb{S}$ can be found in an input set of solutions $\mathbb{I}$ using methods such as clustering. The objective of iSMO is to generate new solutions similar to the reference set $\mathbb{S}$. A user can repeat this process of identifying preferred solutions and generating more of such solutions as many times as required. This leads to a better understanding of the local Pareto front in the vicinity of the preferred region of the design space. In this chapter, we develop iSMO to work with any w-MTO method such as SEW-HCA. However, using the components developed in this section, iSMO may be extended to handle any design optimization whose hyperparameters can be tuned to yield different solutions.

Figure 16 illustrates two iterations of iSMO to progressively generate solutions in the preferred regions of a Pareto front. In each iteration, clustering is used to select a preferred set of solutions. iSMO is used to predict the new weights required to generate diverse solutions in the preferred region.

Problem Formulation

TO optimizes material layout in a design domain Ω. For example, in a density-based TO, Ω is discretized into N_e number of elements and the TO determines the relative density $\rho \in [0, 1]$ of each of the elements: the optimized design $\mathbf{d}$ is given by a vector of element densities $\boldsymbol{\rho} = [\rho_e]_{e=1}^{N_e}$. For multiple objectives $\mathbf{f} = [f_m]_{m=1}^{N_f}$ and the corresponding weights $\mathbf{w} = [w_m]_{m=1}^{N_f}$, a w-MTO determines the optimal design $\mathbf{d}(\mathbf{w})$ (see Chapter 3 for more details). Given a reference set $\mathbb{S}$ that defines a preferred region $\mathbb{P}$, iSMO needs to find the set $\mathbb{W} = \{\mathbf{w}_n\}_{n=1}^{N}$ of weight-vectors which will result in a diverse set $\mathbb{D}$ of N Pareto-optimal solutions: $\mathbb{D} = \{\mathbf{d}(\mathbf{w}_n)\}_{n=1}^{N} \subset \mathbb{P}$. So, iSMO solves the following problem:

$$\begin{aligned} \min_{\mathbf{w}} \mathbf{f} &= [f_m(\mathbf{w})]_{m=1}^{N_f}, \\ \text{s.t.} \quad C(\mathbf{w}) &= 0, \end{aligned} \tag{6.1}$$

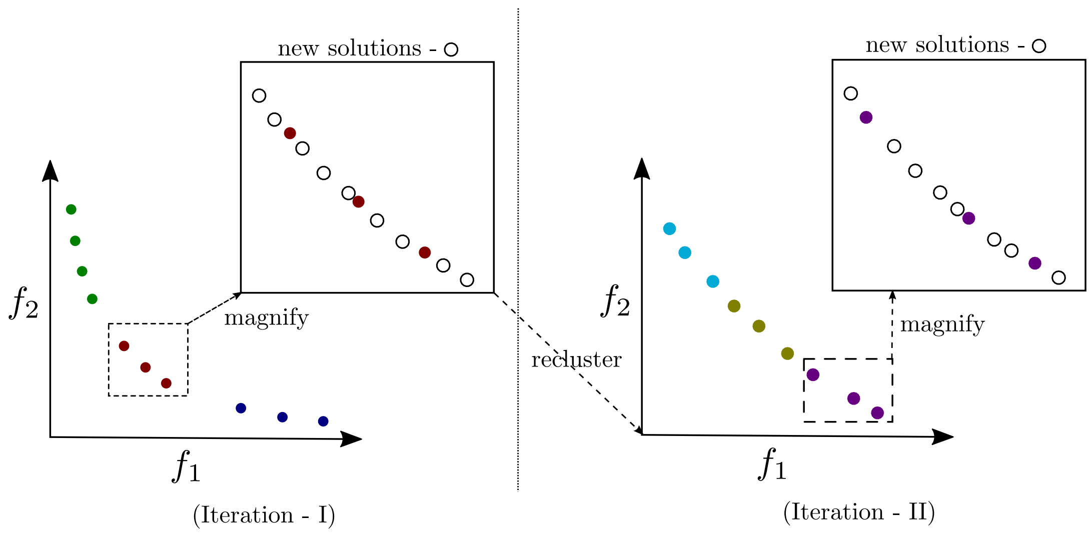


Figure 16 Two iterations of iSMO for a MOP with objectives f_1, f_2. In each iteration, (i) the solutions are clustered, (ii) a cluster is selected, and (iii) new solutions are generated for the selected cluster by iSMO. The new solutions in an iteration are used as input for clustering in the next iteration.

where each objective $f_m(\mathbf{w})$ corresponds to the design $\mathbf{d}(\mathbf{w})$ obtained using w-MTO and the constraint function $C(\mathbf{w}) = 0$ only if $\mathbf{d}(\mathbf{w})$ belongs to the preferred region, i.e., $\mathbf{d}(\mathbf{w}) \in \mathbb{P}$. We propose an EA to solve the above problem.

In the following, we will discuss the different components of iSMO, given by Equation (6.1). First, we need to choose a preferred set as input to iSMO (Section 6.1.1). Since EAs need to consider multiple solutions before reaching the desired optima and w-MTO is expensive, we use regression models to predict **f** from **w** (Section 6.1.2). Furthermore, we need to constrain the solutions to the preferred region, which can be achieved by using a classifier model (Section 6.1.3). Different regressors and classifier models can be compared using the cross-validation method [18], which is briefly described in Section 6.1.4. Finally, we will present an EA to obtain a set of **w** that will result in a diverse set of solutions (Section 6.1.5).

6.1.1. Identification of Preferred Solutions using Clustering

As discussed in Chapter 4, clustering methods can be used to find the preferred solutions. Given a metric for comparing solutions, a clustering method such as k-means organizes the input data into clusters (Section 4.2). After analysis, an engineer may choose one or more clusters as the preferred set $\mathbb{S}$. For example, designs can be grouped based on similar performance [55] or geometrical structure [113]. A cluster of solutions with the desired performance or geometrical structure can be chosen as $\mathbb{S}$.

For a set of N_f objectives and the corresponding weights, w-MTO yields an optimal design. So, w-MTO with an input weight-vector $\mathbf{w} = \{w_m\}_{m=1}^{N_f}$ results in an optimal solution $\mathbf{d}(\mathbf{w})$ with the objective values: $\mathbf{f} = \{f_m\}_{m=1}^{N_f}$. Using w-MTO with a sampled set of weight-vectors

$\mathbb{W}_0 = \{\mathbf{w}_n\}_{n=1}^N$, an initial set of N solutions $\mathbb{I} = \{\mathbf{d}(\mathbf{w}_n)\}_{n=1}^N$ can be generated. Using a clustering method, $\mathbb{I}$ is split into clusters. If there are k number of clusters, each solution $\mathbf{d}_i \in \mathbb{I}$ is assigned a cluster label $p_i \in \{1, ..., k\}$ according to the cluster it belongs to. If the user chooses a cluster p_{ref}, then the preferred set $\mathbb{S} = \{\mathbf{d}_i \mid p_i = p_{\text{ref}}\}$.

6.1.2. Prediction of Objectives Using a Regressor

iSMO needs to inexpensively evaluate the objectives **f** for different weight-vectors **w**. To generate $\mathbb{I}$, w-MTO uses a sample of weights $\mathbb{W}_0 = \{\mathbf{w}_n\}_{n=1}^N$ to generate designs with objective values $\mathbb{F}_0 = \{\mathbf{f}_n\}_{n=1}^N$. Using $\mathbb{F}_0$ and $\mathbb{W}_0$, we can train a regressor model $P_\mathbf{f}$ to predict **f** from an arbitrary **w**. Some of the criteria for choosing an appropriate model for $P_\mathbf{f}$ are as follows:

- Redundant weights: Changing **w** by multiplying a positive value does not change the resultant design of w-MTO. So, it might be easier to train better models, if the redundancy is removed in the inputs. For example, we can normalize the weights using L1-norm such that $\|\mathbf{w}\|_1 = \sum_{m=1,...,N_f} w_m = 1$ (weights are always positive) and remove the last weight to yield the input vector $[w_m]_{m=1}^{N_f-1}$ for the regressor model.
- Dimension of input: If the dimension **w** is large, the input may need to be preprocessed by dimensionality reduction techniques such as PCA for better models.
- Dimension of output: If the dimension of **f** is greater than 1, we need to ensure that the regression models can output multiple values.
- Size of input: Since w-MTO is expensive, the size of training data might be small, in which case simpler models such as random forest regressors [128] might be preferred over complex models such as neural-network regression models to avoid overfitting [18, 129].

6.1.3. Prediction of Cluster Labels using a Classifier

The constraint function $C(\mathbf{w}) = 0$ is used to check if a weight vector **w** results in a solution that is similar to the preferred set $\mathbb{S}$, and hence is in the preferred region $\mathbb{P}$. So,

$$C(\mathbf{w}) = 0 \text{ iff } \mathbf{d}(\mathbf{w}) \in \mathbb{P}. \tag{6.2}$$

As discussed in Section 6.1.1, $\mathbb{S}$ is selected among a given $\mathbb{I}$, where $\mathbb{I} - \mathbb{S} = \mathbb{S}'$ are the remaining solutions. It is difficult to define a preferred region explicitly but we can treat $\mathbb{S}$ and $\mathbb{S}'$ as two classes of solutions $C_\mathbb{S}$ (class label = 0) and $C_{\mathbb{S}'}$ (class label = 1), and train a binary classifier to distinguish the classes. We propose to build a classifier with **w** as input and the corresponding class labels as output. If clustering is used to find the classes in the solutions, as in this thesis, we can predict the cluster labels from **w**, in which case we are building a multi-class classifier model $P_p(\mathbf{w})$ to predict the cluster labels.

The constraint function is then given by:

$$C(\mathbf{w}) = |P_p(\mathbf{w}) - p_{\mathrm{ref}}|, \tag{6.3}$$

where $P_p(\mathbf{w})$ is the multi-class classifier model and p_{ref} is the preferred cluster label. To train the model, we can use the $\mathbf{w}$ (weights) and p (cluster labels) of the initial solutions $\mathbb{I}$.

Other Classifier Models for Predicting Cluster Labels

If clustering uses the features $\mathbf{y}$ of designs $\mathbf{d}(\mathbf{w})$, then the classifier model can be trained with $\mathbf{y}$ as input. While this might lead to an improvement in the classifier accuracy, it requires an additional regressor model $P_{\mathbf{y}}(\mathbf{w})$ to predict the features $\mathbf{y}$ which are used by $P_p(\mathbf{y})$ to predict cluster labels p. Using these two models in serial may negatively affect the overall accuracy since the errors propagate through the models. Especially, if the dimension of $\mathbf{y}$ is large compared to that of $\mathbf{w}$, the accuracy of $P_{\mathbf{y}}(\mathbf{w})$ can be low. For these reasons, we have chosen not to investigate in this direction. However, one may compare these models with our proposed classifier model using the cross-validation method.

6.1.4. Evaluation Methods for Metamodels

The regression performance of metamodels can be measured using mean square error (MSE) or the coefficient of determination (R2 score). The classification performance of models can be measured using precision, recall, or F1 score. MSE, precision, and recall measure model errors and need to be small. By contrast, measures that are called scores need to be high. With accurate metamodels, iSMO will more likely yield more diverse solutions. However, if the training data is used to measure the model error/score (called *training error/score*), we will overestimate the model performance [18].

We can determine whether a model will perform well on new data using cross-validation. First, the dataset is split into training and validation sets. The training set is used to train the model while the validation set is used to measure the model's performance. For example, in k-fold cross-validation [16, 18], the dataset is split into k-folds, where k-1 folds are used for training, and the remaining fold is used for measuring the error/score of the model (called *cross-validation error/score*); there are k possible ways of selecting unique folds here. This process may be further repeated by reshuffling the data. Using different cross-validation splits, we can analyze the variation in the model performance.

If both cross-validation and training scores of a model are low, the model is *underfitting* and we need a more complex model or more data. If only the cross-validation performance is bad, the model is *overfitting* to the data, in which case the model needs to be regularized by adding constraints or a simpler model is chosen [18].

Each regressor or classifier models have multiple hyperparameters that can be tuned to obtain the best model for a given task. Cross-validation scores can help choose the model class as well as tune its hyperparameters. It is recommended that apart from the training and the cross-validation dataset discussed above, an unused test dataset is used to report the model accuracy [18]. However, in this thesis, we do not perform additional hyperparameter tuning since the model accuracy is sufficient for our test cases without it. Hence, we do not use any separate test dataset. However, depending on the use case, one may do hyperparameter tuning and use a test dataset.

In this thesis, we use the scikit-learn [130] library in Python to build the models as well as evaluate them using cross-validation. Unless otherwise specified, we use the hyperparameters shown in Table 3 for different regressors and classifiers in this thesis. These are the default values used by the scikit-learn library (version 1.0.2). For further details, see scikit-learn documentation [130].

Table 3 Default hyperparameters used by regressors and classifiers in this thesis. The parameter name is given with value in brackets.

Metamodels	Default values for hyperparameters
Gaussian process regressor	kernel (radial basis function)
Random forest regressor	number of estimators (100), split criterion (mean squared error), limits on depth/leaf size (None)
k-nearest neighbor regressor	number of neighbors (5), metric (Euclidean distance), weight function (uniform)
k-nearest neighbor classifier	same as above
Linear regressor	least squares method without regularization
Random forest classifier	number of estimators (100), split criterion (gini impurity), limits on depth/leaf size (None)
Support vector classifier	regularization parameter (C = 1), kernel (radial basis function), decision function (one-vs-rest)
Logistic regression classifier	penalty (l2 norm), regularization parameter (C = 1), maximum iterations (100)
Linear support vector classifier	penalty (l2 norm), loss (squared hinge), regularization parameter (C=1), maximum iterations (1000)

6.1.5. Identification of Desired Weight-vectors

As discussed previously, the $P_{\mathbf{f}}(\mathbf{w})$ can predict the objectives from weights, and $C(\mathbf{w})$ can be used to restrict the weights such that the resultant solutions belong to the selected cluster. Using these inexpensive metamodels, EA can be used to find the new weights that will result in a diverse set of preferred solutions, completing a single iteration of iSMO (Figure 17). In this thesis, the EA step in iSMO follows the framework of MOEAs, which have been very successful in identifying a multitude of Pareto-optimal solutions [25].

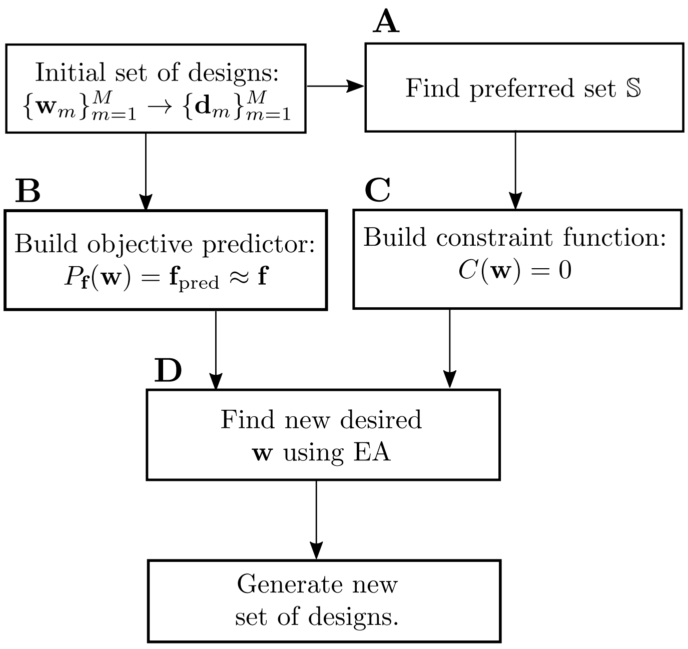

A
Initial set of designs: $\{\mathbf{w}_m\}_{m=1}^{M} \rightarrow \{\mathbf{d}_m\}_{m=1}^{M}$
Find preferred set $\mathbb{S}$
B
Build objective predictor: $P_{\mathbf{f}}(\mathbf{w}) = \mathbf{f}_{\text{pred}} \approx \mathbf{f}$
C
Build constraint function: $C(\mathbf{w}) = 0$
D
Find new desired $\mathbf{w}$ using EA
Generate new set of designs.

Figure 17 An iteration of iSMO. Given an initial set of designs and a preferred set among them (Step: **A**), we build a regressor with a weight-vector **w** as input to predict objectives **f** (Step: **B**) and a constraint function that selects **w** based on the preferred set (Step: **C**). Using EA, we find the set of **w** (Step: **D**) that will result in the desired designs.

MOEAs iteratively apply so-called mutation, crossover, and selection operators to evolve a population of solutions. In each iteration, crossover and mutation operators are used to generate a wide range of solutions, which are then filtered by the selection operator such that they (1) satisfy the constraints of the given problem, (2) are Pareto-optimal, and (3) are well-spread (diverse) in the objective space. Following the framework of an MOEA, iSMO uses these operators iteratively to find the desired weights.

In this thesis, the EA step of iSMO is based on NSGA-II [24], a widely used genetic algorithm (GA) for multi-objective optimization. iSMO uses the same crossover and mutation operators used by NSGA-II. Since NSGA-II is a real-coded GA, we do not encode solution values into binary strings. Therefore, the crossover operator uses the simulated binary method, which simulates the single-point crossover operator for binary-coded inputs. The mutation operator uses the polynomial method, as in NSGA-II. These operators may also be replaced by other variations of crossover and mutation operators [25].

For the selection step, iSMO uses only some of the selection criteria used by NSGA-II. First, we select only the solutions that satisfy the relevant constraint for our problem, i.e., the cluster constraint ($C(\mathbf{w}) = 0$). Since the decision variable is the weight-vector **w**, any resultant solution will automatically be near the Pareto-front if w-MTO is efficient. So, we do not need to ensure Pareto-optimality as in NSGA-II, which uses the non-dominance ranking to move solutions towards the Pareto front. In fact, due to inaccuracies of $P_{\mathbf{f}}(\mathbf{w})$, the predicted objectives may not be strictly non-dominated. Therefore, using the non-dominance ranking might be detrimental to our application. Finally, iSMO ensures diversity in the objective space using the crowding distance [24], which roughly measures the average distance to the nearest neighbors. For diversity, solutions with higher crowding distances need to be preferentially selected. So, iSMO uses only the given constraints and diversity criterion in the selection operator. By repeated application of crossover, mutation and selection operators, iSMO provides the weights required by TO to generate more solutions in the desired cluster.

6.2. Evaluation Methods

Some of the objectives of a multi-objective algorithm ensure that solutions are close to the Pareto front (in short, *optimality*), a good distribution of solutions in the objective space (*diversity*), and a large range of solutions (*range*). Measures such as generational distance (GD), and hyper volume (HV) quantify *optimality*, while measures such as spacing distance and Pareto spread quantify *diversity* [131–133]. Hypervolume or other measures can be an indicator of *range* [134].

In this thesis, our objective is to find new solutions similar to a preferred set $\mathbb{S}$, which we refer to as *preferredness*. The underlying w-MTO method used by iSMO is responsible for the *optimality* of a solution. Hence, we can ignore the *optimality* measures and only measure *preferredness*, *diversity*, and *range* of the new solutions. In this thesis, we use the following methods to measure *preferredness* and *diversity*.

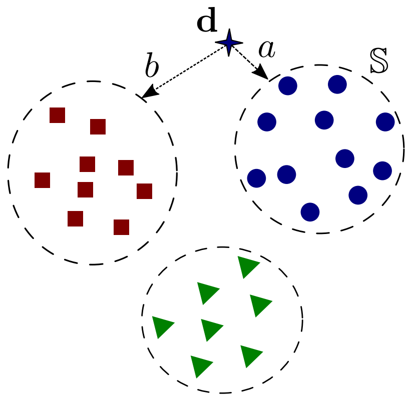


Figure 18 Silhouette score s for each new solution **d**, given a preferred cluster $\mathbb{S}$. a is the average distance to other solutions in the same cluster and b is the minimum average distance to other clusters. The score $s = \frac{(b-a)}{\max(b,a)}$.

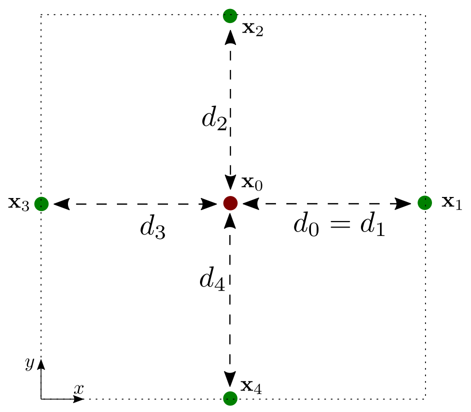
6.2.1. Silhouette Score

Constraint functions used by iSMO cannot be used to evaluate the new solutions, which always satisfy the constraints. However, silhouette score [79] (Section 4.2.4) can be used to quantify the *preferredness* of the new solutions to the selected cluster. Figure 18 illustrates a method to obtain the silhouette score of a new solution in cluster $\mathbb{S}$. In this calculation, the metric used for clustering should be used to measure the distances. The average silhouette score for a set of new solutions indicates the quality of preferredness to the preferred set $\mathbb{S}$. While a high score ($s \approx 1$) is preferred, the silhouette score is expected to be low for contiguous data, where the clusters are not well-separated. In particular, the score is low for samples on the boundary of clusters.

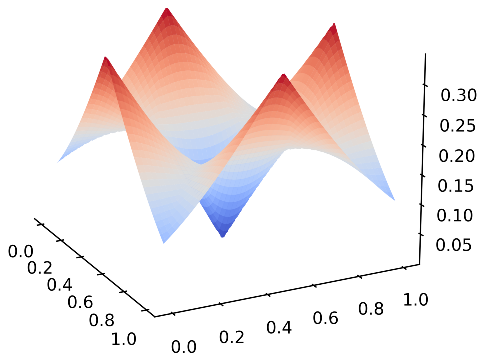
The silhouette score may penalize the boundary solutions to large clusters since the average distance tends to be large for such clusters. For the k-means method, which identifies clusters of similar sizes, this is less of a problem. However, for density-based algorithms that yield clusters of unequal sizes, the silhouette score may not be the right measure. A promising alternative is to use the minimum distance, instead of the mean distance, to the set of solutions while defining a and b in the silhouette score. In this thesis, we do not investigate this further and use the silhouette score as a measure of *preferredness*.

6.2.2. Diversity Score

In multi-objective optimization, multiple measures of diversity are available [133, 135]. The distribution of non-dominated individuals can be measured using the variance of niche count of each individual [135, 136]. However, it is difficult to define the niche size required to calculate the niche count. Deb et al. [24] measure the spread of Pareto optima in 2D using the consecutive distance between the solutions, where no pairwise distance is used more than once. However, this definition cannot be easily extended to higher dimensions. To alleviate this problem, we can use the Spacing metric that measures the variance of the shortest distance between samples, even if certain values may be repeatedly used [137]. In the following, we generalize the definition of the Spacing metric to higher dimensions.



(a) $\Delta = 0$ for well-spread samples. $\mathbf{x}_0 = (0.5, 0.5)$. Other points are along the x- or y-axis at a distance of 0.5 units.



(b) Δ for variation of $\mathbf{x}_0$: In the plot, (x, y) are the coordinates of $\mathbf{x}_0$. Along z is the corresponding spacing metric $\Delta([\mathbf{x}_i]_{i=0}^{4})$.

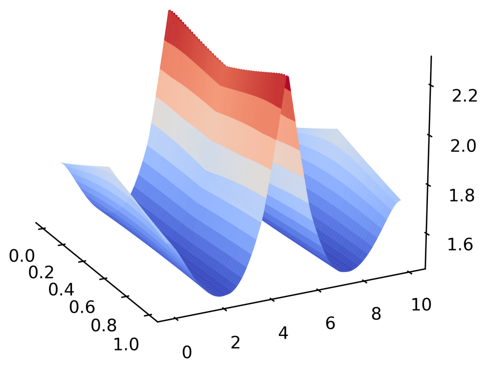
Figure 19 Measuring Δ of five 2D points: $[\mathbf{x}_i]_{i=1}^{4}$. The data can be understood as a projection of a 3D Pareto front onto a plane. For each point $\mathbf{x}_i$, we show the shortest distance d_i to other samples. Δ is smallest when $\mathbf{x}_0 = (0.5, 0.5)$, and is equidistant from the other points. Δ is high when $\mathbf{x}_0$ coincides with another point. Δ is moderately small when $\mathbf{x}_0$ is at the corners of the dotted square in (a).

For a given data $X = \{\mathbf{x}_i\}_{i=1}^n$ with n samples, the spacing metric Δ is defined as follows:

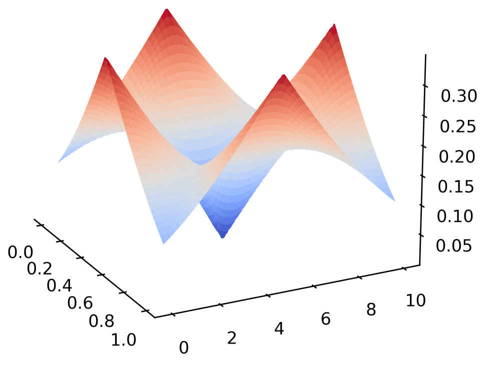
$$\begin{aligned} \Delta &= \mathrm{std}\,[d_i]_{i=1}^n, \\ \text{where} \quad d_i &= \min_{j=1}^{n} D(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \tag{6.4}$$

D is a distance measure for the given data, and std measures the standard deviation of the distance values. Typical choices for D are Euclidean distance or the Manhattan distance (L1-distance) in the objective space. For simplicity, std is calculated using the biased sample variance (second central moment of the sample), where the bias decreases with the increase in the number of distance values. $\Delta \in [0, \inf)$ and the higher the value, the lower the diversity. For evenly spread samples, $\Delta = 0$.

Figure 19 illustrates the definition of Δ using a small number of samples. $\Delta = 0$ when the shortest distance to other points does not vary, i.e., $d_i = d_j \ \forall \ i, j$. To better understand Δ, we can change the position of one of the points and measure Δ (Figure 19b), whose variation agrees with our intuition.



(a) Δ when the input has a large variation along an axis.



(b) Δ with standard scaling of input.

Figure 20 Δ is sensitive to the variation of input along different axes. When the input points of Figure 19 are scaled along the x-axis by 10 times, i.e., $d_0 = d_1 = d_3 = 5$ and $d_2 = d_4 = 0.5$, Δ is skewed along the x-axis. If the input is normalized, we can eliminate this problem.

One disadvantage of using Δ is that it is sensitive to the variation of input along different axes. For example, we can increase the distance between the points along the x-axis (Figure 19) and observe the variation of Δ in Figure 20a. To avoid this, we normalize the data before measuring Δ. Figure 20b shows that after normalization, we can treat variation along with different features (e.g., objective values) of the data equally. In this study, we use the Spacing metric with normalization, referred to as Δ' hereafter. Min-max scaling is used to normalize the data such that for any given sample $\mathbf{x} \in X$, each feature $x_i \in [0, 1]$.

Since score measures are defined such that higher values are preferred, we define the diversity score to be $-\Delta'$, which then lies in the range: $[-\infty, 0]$. Samples with low diversity have significant negative scores. If there are no samples or just one sample, we define the diversity score to be $-\infty$, since they are undesirable outcomes.

Other metrics worth investigating are diversity metrics using entropy (e.g., [138]) and metrics of the solution range (e.g., [134]).

Summary

In this chapter, we presented a cooperative framework called iSMO that generates diverse solutions similar to the preferred set of solutions identified using clustering. To this end, iSMO predicts the *weight-vectors* required by a multi-objective TO such as SEW-HCA to generate the *desired* solutions. A critical component of iSMO is an EA based on NSGA-II, which can identify diverse solutions in objective space. Since TO requires expensive simulations and EA requires multiple evaluations of objective functions and constraint equations, I proposed using metamodels with weights as input to predict objective values and the cluster label needed by the constraint equation. After initial training, metamodels are effectively used by EA to find the new desired weights. Finally, we described quantitative measures—silhouette and diversity scores—to evaluate the solutions obtained by iSMO.

Part III

Results

7. Evaluation of Geometric Features using Test Datasets

In this chapter, we evaluate the different feature extraction methods for geometry using the methods discussed in Chapter 5. An extraction method is regarded as successful here if the identified geometric feature-vector ($\mathbf{g}$), whose Euclidean distance (ED) shows the required invariance properties and is a meaningful metric of geometry. For this purpose, we design artificial datasets in Section 7.1. First, we use simple datasets, rather than more complex ones, to easily verify if the ED in $\mathbf{g}$ is sensitive or not to transformations such as rotation, elongation, and translation. Then, with more topologically complex design sets, different $\mathbf{g}$ are challenged and compared with proven reference metrics. Since we intend to cluster geometrically similar structures in a dataset, we evaluate next the suitability of different $\mathbf{g}$ vectors for clustering using an additional, more challenging test set. Finally, in Section 7.3, we explore topologically optimized designs using $\mathbf{g}$ vectors extracted with the Autoencoder, which had the best overall performance, followed by a discussion in Section 7.4.

7.1. Dataset Generation

First, we introduce simple datasets, each of which contains the designs that are generated by simple transformations of a template design. Then, we describe datasets with topologically complex designs and well-defined subclasses.

7.1.1. Ellipsoidal designs

We generate three datasets where the geometrical differences are easy to measure. Within a dataset, designs are obtained by transforming a template design using rotation, elongation, or translation. Hence, the reference metric (RM) for these designs is easy to define, e.g., for rotated designs, RM is the difference in the rotated angle. Although these datasets are simple, we can evaluate if the Euclidean distance with $\mathbf{g}$ is similar to RM. Furthermore, by testing the more complicated metrics discussed in Section 5.2, we can gain an intuition for different metrics.

The template for these datasets is an ellipsoid, which can be generated using a moving morphable component (MMC) with a form factor (or modeling exponent) $q = 2$ [43]. Using large even values for q, we can change the form of the MMC component, e.g., using $q = 2$ and $q = 6$ results in ellipsoidal and cuboidal MMCs, respectively. The reference MMC ellipsoid, also called beam here, can be transformed by varying the Euler angles ($\mathbf{E} \in \mathbb{R}^3$), lengths along the three principal axes ($\mathbf{L} \in \mathbb{R}^3$), and the position of the center of mass ($\mathbf{C} \in \mathbb{R}^3$), generating three different datasets.

MMC defines a design using a level-set function $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}$, where the surface is given by $\{\mathbf{x} \in \mathbb{R}^3 \mid \Phi(\mathbf{x}) = 0\}$ and the interior is given by $\{\mathbf{x} \in \mathbb{R}^3 \mid \Phi(\mathbf{x}) > 0\}$. Using level-sets, we can generate complex topologies along with the relevant voxel and pointcloud data. For a given dataset, to generate voxel data, a common domain containing all the designs is split into voxels (Section 5.1). If a common domain is not used, the voxel data cannot be directly compared. For each design, to verify if i-th voxel is occupied, we check if $\Phi(\mathbf{x}_i) \geq 0$ at the voxel center $\mathbf{x}_i$. Using the marching cubes algorithm [98, 99], the voxel data can be converted to a triangulated surface mesh. For pointcloud data, points are uniformly sampled on the triangulated surface mesh using the *trimesh* library [100, 101].

Beam-rotation dataset: An ellipsoidal beam is rotated by different angles along one of the principal axes to generate 20 different designs (Figure 21). RM measures the difference in the rotated angle. The rotation angle ranges from $0°$ to $90°$, otherwise, the difference in angle is not a good metric. For example, consider a beam B_1 rotated by a $\theta \in [0°, 90°]$ to give a beam B_2 and by an $\alpha = 180° - \theta \in [90°, 180°]$ to give a beam B_3. RM shows that B_1 is more similar to B_2 ($\Delta = \theta$) than to B_3 ($\Delta = \alpha$), which is incorrect since B_2 and B_3 fully overlap.

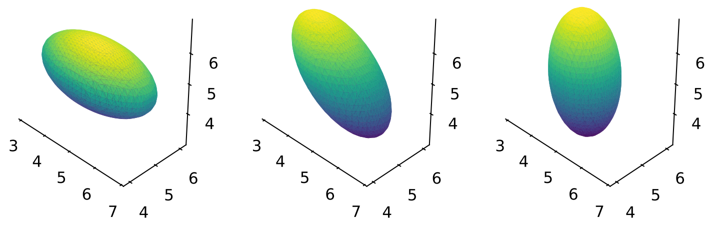


Figure 21 Beam-rotation dataset: An ellipsoidal beam is rotated by different angles.

Beam-elongation dataset: An ellipsoidal beam is elongated by different lengths along a fixed principal axis to generate 20 new designs (Figure 22). RM measures the difference in lengths along the fixed axis.

Beam-translation dataset: An ellipsoidal beam is translated by different amounts along a fixed principal axis to generate 20 new designs (Figure 23). RM measures the difference in the translated distance.

7.1.2. Topologically-complex designs

In the following, we discuss methods to generate complex truss-like designs by combining MMCs. Furthermore, we use different template frames to generate designs that distinctly belong to different subclasses.

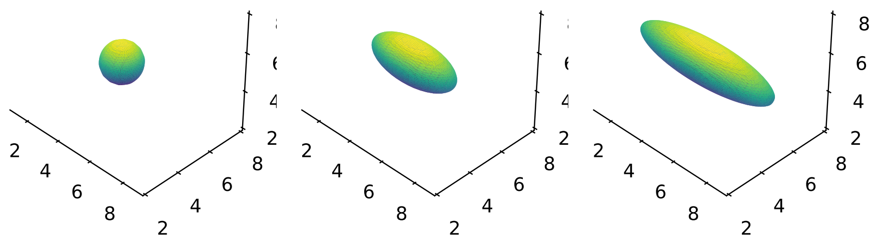


Figure 22 Beam-elongation dataset: An ellipsoidal beam is elongated by different lengths.

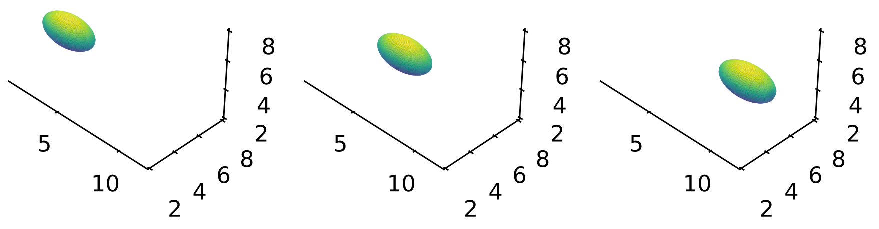


Figure 23 Beam-translation dataset: An ellipsoidal beam is translated by different amounts.

MMCs and other feature mapping techniques [139–142] are increasingly used in TO to construct complex structures using a few design variables. For example, Zhang et al. could generate arbitrarily curved beams by overlapping ellipsoids [139]. With a relatively small number of MMCs, Bujny et al. generated complex topologies using evolutionary algorithms [26, 43]. Even the optimal structures for highly nonlinear crash TO problems have been derived using a union of MMC beams [3].

Complex structures can be composed using MMCs: the interior of a design with n MMCs is defined by $\max\limits_{i=1,\ldots,n} \Phi_i > 0$. As discussed previously, using the level-set function, voxel, and pointcloud representations can be generated. In this study, we use ellipsoidal MMCs to construct designs with different topologies. The shapes of MMCs are not changed, since we are interested in the geometric structure rather than the subtle differences in shape. In the future, the effects of shape may be further studied.

In this thesis, I propose using 3D geometric graphs to generate datasets with well-defined subclasses. A 3D geometric graph contains nodes corresponding to points in 3D Euclidean space. The graph edges correspond to line segments connecting the 3D points. Figure 24 shows exemplary graphs based on cubes, where vertices of cubes indicate the graph nodes and the edges of cubes form the connections/edges in the graph. Given such a 3D graph, MMCs are positioned using their principal axis along a distinct edge of the graph (Figures 24, 25). By varying the thickness of beams, one can generate multiple designs with a similar structure. We can also truncate or remove a few of the beams to affect the topology of structures slightly. Using this method, we can generate multiple designs with the same underlying structure but with differences in topology, size, and shape. Repeating this process with distinct graphs yields a dataset with well-defined subclasses. In this study, we generate three such datasets with increasing complexity.

Three-cube trusses: Firstly, we construct 6 distinct *subgraphs* using a subset of nodes and edges of the *basegraph* shown in Figure 24. Each *subgraph* is used to generate a subclass of designs with similar structure. Figure 25 shows a sample design from each of the 6 subclasses. Note that the center of the underlying graphs (mean of the graph nodes) changes from subclass to subclass.

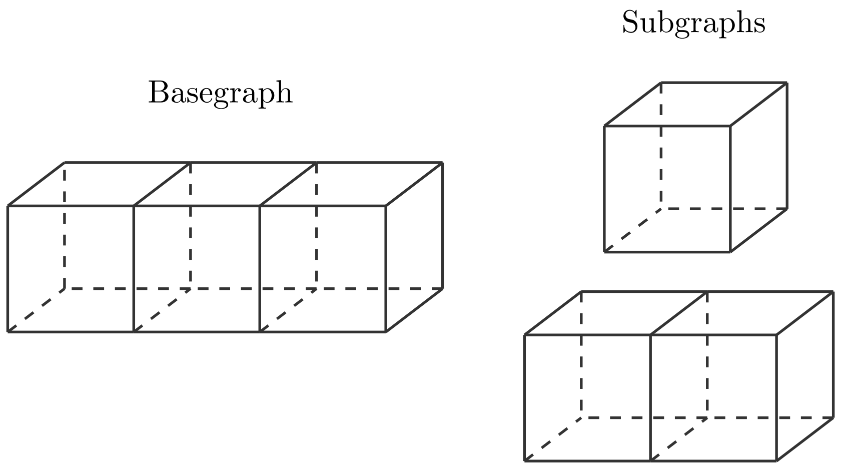


Figure 24 Three-cube basegraph and its subgraphs: The vertices of the cubes are graph nodes. Line segments joining any two vertices of the same cube are graph edges. For clarity, only some of the graph edges, i.e., the cube edges are shown in the figure. Hidden edges of the cubes are shown as dashed lines.

Single-cube trusses: This dataset is based on the basegraph shown in Figure 26. The basegraph is used to generate 11 different connected subgraphs, where each subgraph generates a subclass of designs. Figure 27 shows samples from six of the subclasses. The designs from different subclasses differ in orientation or/and topology.

Randomized topologies: This dataset consists of 50 subclasses with 20 designs per class, with complex variations in topology. Figure 24 shows the relevant basegraph but the subgraphs are constructed randomly using the following steps:

1. Construct a subgraph for each subclass.

    a. Randomly pick 5 to 10 vertices from the basegraph (Figure 24).

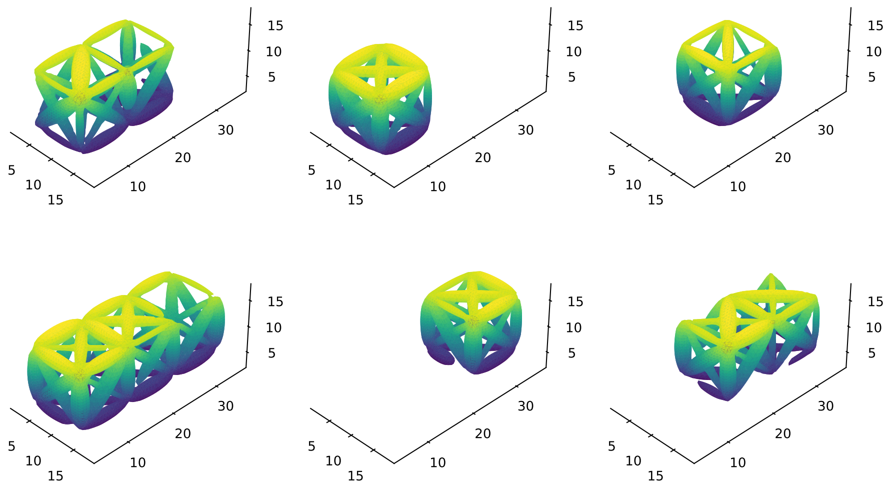


Figure 25 Three-cube truss dataset: Sample designs from 6 different subclasses.

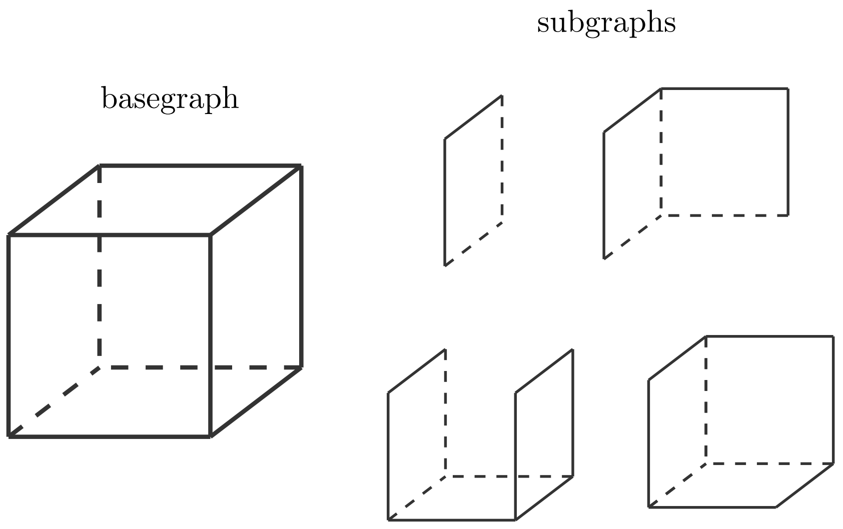


Figure 26 Single-cube basegraph and subgraphs: Graph nodes are the cube vertices and graph edges exist between every pair of nodes. Each subgraph of the basegraph yields a subclass of designs. Hidden edges in the graphs are shown as dashed lines.

   b. The subgraph then contains every possible edge connecting the vertices.

2. Generate multiple designs for a given subgraph.

   a. Align MMCs along the edges. Vary the edge thickness by sampling from a uniform distribution in [1.5, 4] (each cube edge is 10 units).

   b. Randomly remove a few of the beams with a low probability of 0.2 using a Bernoulli trial.

Note that for each subgraph, beams are removed with low probability since the resultant designs should still belong to the same subclass. Figure 28 shows 3 samples from 3 subgraphs.

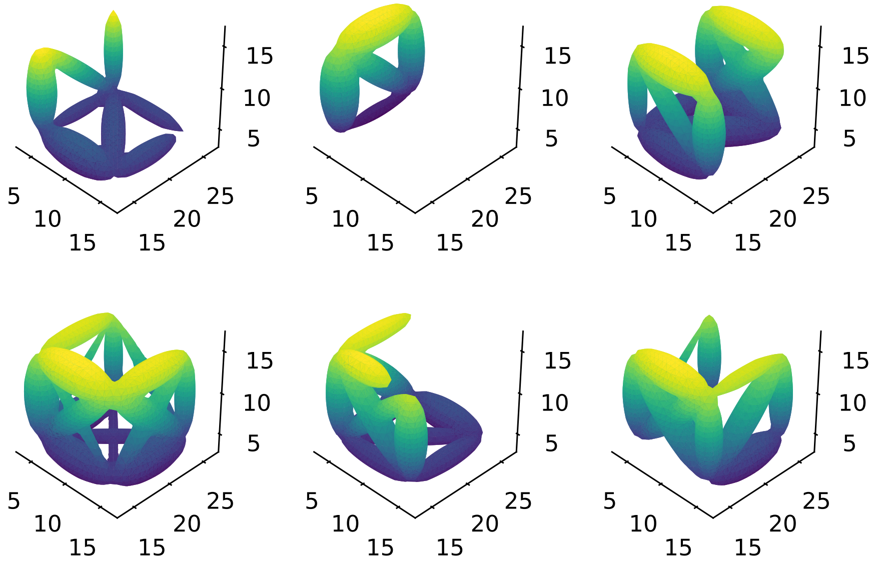


Figure 27 Single-cube truss dataset: sample designs from 6 different subclasses.

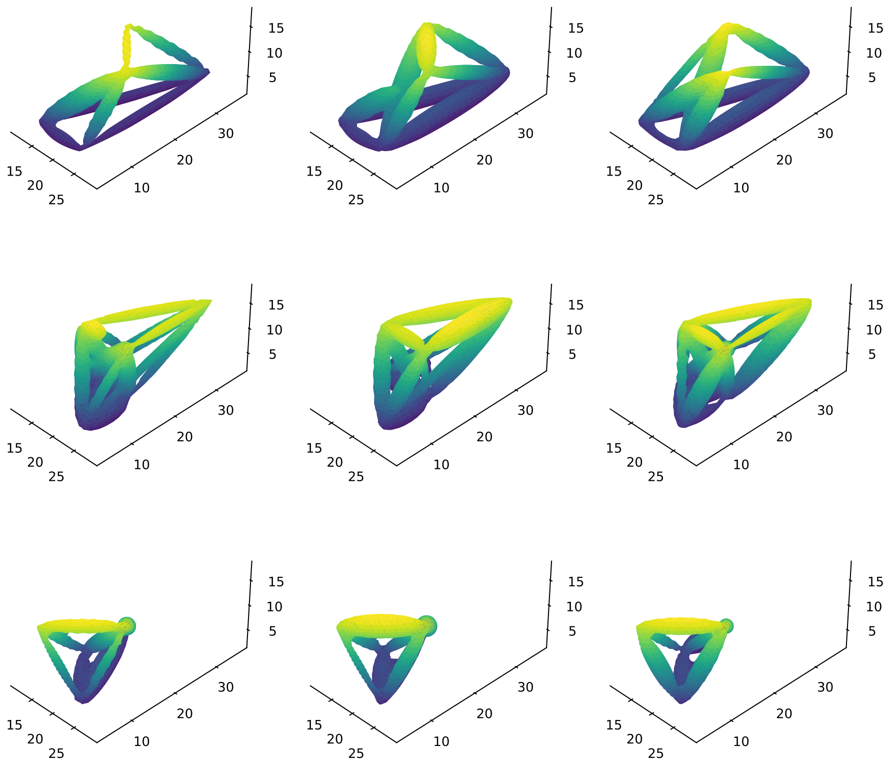


Figure 28 Randomized topologies: Each row shows designs from a subclass based on a subgraph. Beam thicknesses are varied as well some beams are removed with low probability.

7.2. Evaluation on Design Datasets

In this section, we evaluate the Euclidean distance (ED) with geometric feature-vectors (**g** vectors) as input using the methods and datasets from Section 5.4 and Section 7.1 respectively.

7.2.1. Naming convention

In general, a metric M may use a specific input data (I) which is processed by one or more dimensionality reduction methods before applying the metric M. So, in this study, we specify a metric along with its input data and the dimensionality reduction techniques for clarity. For brevity, the metrics are defined using a tuple: (I, D, M), which means that the input data I is reduced using a dimensionality reduction technique D whose output is analyzed using a metric M to yield the final distance measure. If D is not used, it may be skipped to give the metric: (I, M). Table 4 shows short names for different I, D, and M for further brevity. Table 5 shows the metrics used in this study.

Table 4 Short names used for input data, dimensionality reduction techniques, and metrics.

Description	Abbreviation	Input
MMC parameters	MMC par	-
Voxel data	Voxel	-
Principal Component Analysis	PCA	Voxel
Non-negative Matrix Factorization	NMF	Voxel
t-distributed Stochastic Neighbor Embedding	t-SNE	Voxel
Uniform Manifold Approximation and Projection	UMAP	Voxel
Pointcloud AutoEncoder	PCAE	Pointcloud
Euclidean distance	ED	-
Chamfer distance	CD	Pointcloud
Earth mover distance	EMD	Pointcloud

As an example metric, consider (MMC par, ED) which measures the Euclidean distance (ED) using the MMC parameters. For a dataset obtained using simple transformations of a template design, this results in a metric that measures the changes in a single MMC parameter, e.g., the changes in an Euler angle, or a single dimension. The metric (Voxel, ED) uses voxel data as input, whose Euclidean Distance (ED) gives the distance between designs. Since ED is used in this metric, voxel data can be considered as a geometric feature-vector (**g** vector). Other **g** vectors are possible; for example in metric (Voxel, PCA, ED), voxel data is processed using PCA to yield a different **g** vector. In this study, we use PCA, NMF, t-SNE, and UMAP reduction methods with voxel data to yield different **g** vectors which results in different metrics. We also consider the metrics (Pointcloud, CD) and (Pointcloud, EMD) with pointcloud data as input. Since ED is not involved, we do not refer to pointcloud data as a **g** vector. However, the metric (Pointcloud, PCAE, ED) uses ED with latent code obtained using PCAE. So the latent code will be referred to as a **g** vector.

7.2.2. Hyperparameters

To convert any design to the voxel data, we use a resolution of $25 \times 25 \times 25$ units. For PCA and NMF, we propose to reduce the voxel data using 10 components. UMAP and t-SNE use 2 components and hence, allow the 2D visualization of clusters in datasets. Each pointcloud contains 2048 points and is used as input to PCAE. The dimension of latent code used by PCAE is 128 for all the datasets except for the last one with TO designs, where 500 dimensions are used. For our datasets, the above parameters are set when a further increase did not seem to significantly improve the results.

Table 5 List of 9 metrics evaluated in this study. For simple datasets, (MMC par, ED) as RM can be used to evaluate the remaining 8 metrics (TMs). For complex data, (MMC par, ED) is not useful. So, we use (Pointcloud, CD) as RM to evaluate the other 7 metrics (TMs).

Metric	Remarks
(MMC par, ED)	RM for simple datasets.
(Pointcloud, CD)	RM for complex geometries.
(Voxel, ED)	Voxel distance is the simplest voxel metric.
(Pointcloud, EMD)	EMD is more expensive than CD.
(Pointcloud, PCAE, ED)	ED with latent code of autoencoder.
(Voxel, PCA, ED)	PCA reduces voxel data to 10 components.
(Voxel, NMF, ED)	NMF reduces voxel data to 10 components.
(Voxel, t-SNE, ED)	t-SNE reduces voxel data to 2 components.
(Voxel, UMAP, ED)	UMAP reduces voxel data to 2 components.

7.2.3. Metric correlations

As discussed in Section 5.4, we measure the correlation between distances measured by a given target metric (TM) with a proven reference metric (RM) of geometrical differences. For brevity, when the values measured by TM and RM are highly correlated, we say that the *metric correlation is high between TM and RM*. A valid metric is chosen as RM if it makes sense intuitively or is found useful in the literature. Therefore, if the metric correlation between RM and TM is high, TM is also meaningful at least for the given dataset. For more details on the following datasets, see Section 7.1.

Beam-rotation dataset: Figure 29 shows the metric correlations for this dataset. RM is the difference in rotated angle: so, RM is (MMC par, ED) for this dataset. Figure 29a shows that the correlation between voxel distance and RM is high. So, the voxel distance is meaningful, as expected. Since the correlation is nonlinear, the Pearson and the Spearman correlation

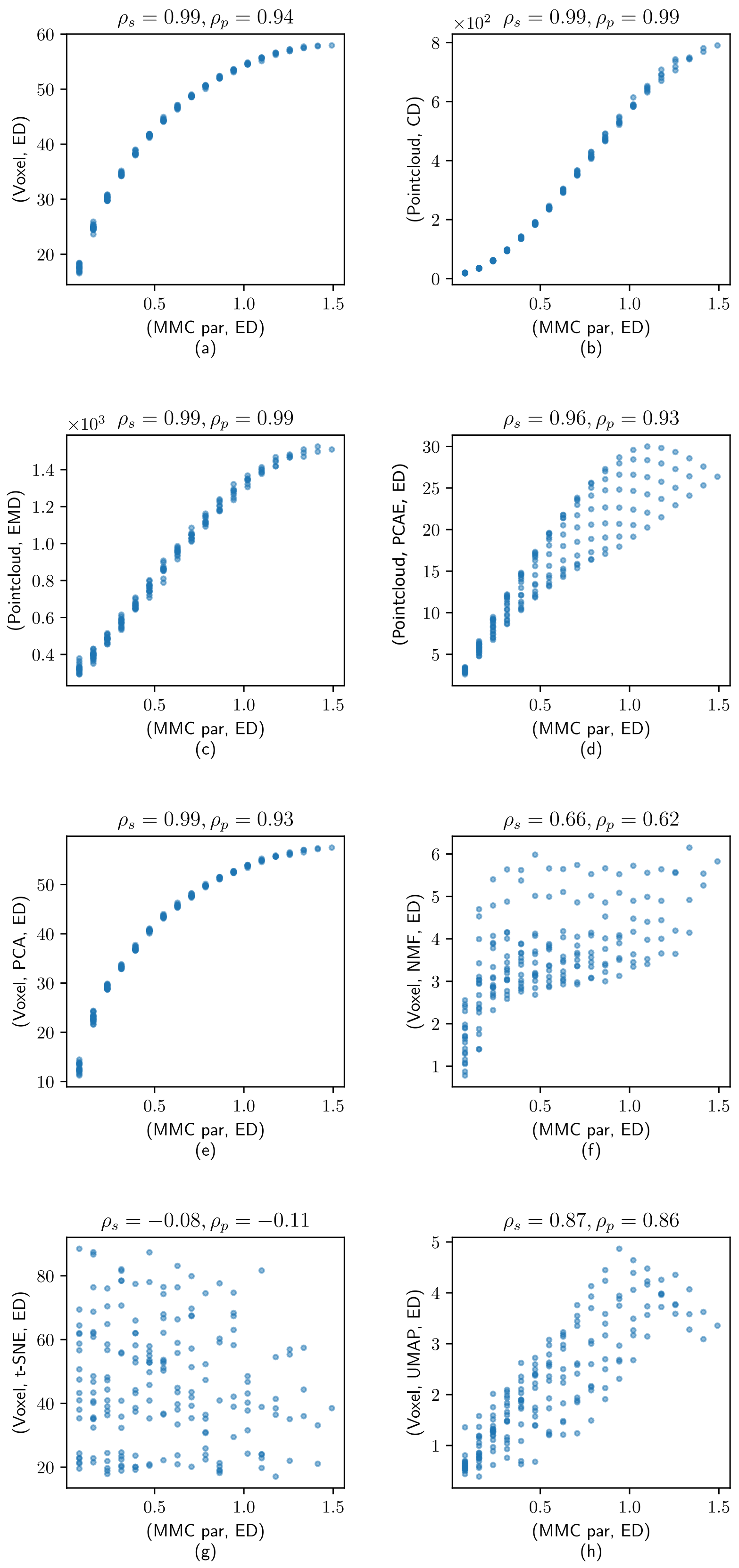


Figure 29 Metric correlations for the beam-rotation dataset: An ellipsoid is rotated by different amounts to yield 20 designs. Distance values measured by RM and TM, i.e., D_{RM} and D_{TM}, are shown along x- and y-axes respectively. Here, RM is (MMC par, ED), i.e., the difference in orientation.

values are different and we have $\rho_p < \rho_s$. Metrics using $\mathbf{g}$ obtained by NMF and t-SNE do not perform well since $\rho_s < 0.7$ with RM, while PCA and UMAP perform reasonably well ($\rho_s > 0.95$). Note that here the metrics with pointcloud as input (pointcloud metrics) have a high correlation with RM.

Interestingly, for certain values of D_{RM}, there are several values for D_{TM}, which results in a vertical segment of dotted points and can be explained as follows. In this dataset, one can find multiple pairs of designs with the same value of D_{RM}, i.e., the difference in orientation is the same for different pairs. Yet, D_{TM} can take different values due to the discretization errors in the voxel data and the randomization errors in the pointcloud data. Even if the relative angle between the two designs is held constant, the actual voxel data depends on the absolute orientation in the design domain. For pointcloud data, this effect occurs because the points are randomly sampled on the surface.

PCAE results in a meaningful TM with $\rho_s = 0.99$ (Figure 29d). PCA reduction of voxel data with 10 features is still useful even if the dimension of voxel data $\approx 1.5 \cdot 10^4$ is large (Figure 29e). Figure 29f shows that ED with t-SNE features is not very meaningful since it is designed only for the visualization of clusters. However, UMAP, although designed for the same application as t-SNE, leads to more meaningful ED with $\mathbf{g}$ (Figure 29g). Figure 29h shows NMF with slightly worse performance than PCA, possibly due to the additional non-negativity constraints on the reduced components. In summary, PCA and PCAE yield meaningful features despite being very low-dimensional compared to voxel data.

Beam-elongation dataset: Figure 30 shows the metric correlations for this dataset. Here, RM measures the difference in elongation lengths. The metric correlations are similar to those of the beam-rotation dataset (Figure 29). Measured values of Voxel distance, CD, and EMD have a high correlation ($\rho_s \geq 0.98$) with those of RM. Among the dimensionality reduction techniques, PCAE and PCA perform the best with $\rho_s = 0.99$. UMAP also performs reasonably well with $\rho_s = 0.95$.

Beam-translation dataset: Figure 31 shows the metric correlations for this dataset. Here, RM measures the difference in the translated distance. In Figure 31a, voxel distance (y-value) does not change when $D_{\mathrm{RM}} > 4$. This is because the designs stop overlapping for this range and the number of voxel differences (D_{TM}) does not change. So, voxel distance is disadvantageous compared to pointcloud metrics, which are sensitive to the position of the non-overlapping material. Other metric correlations are similar to previous results. Measured values of voxel distance, CD, and EMD still have a high correlation ($\rho_s \geq 0.98$) with those of RM. Among the dimensionality reduction techniques, PCAE and PCA perform the best with $\rho_s \geq 0.97$. UMAP also performs reasonably well with $\rho_s = 0.93$.

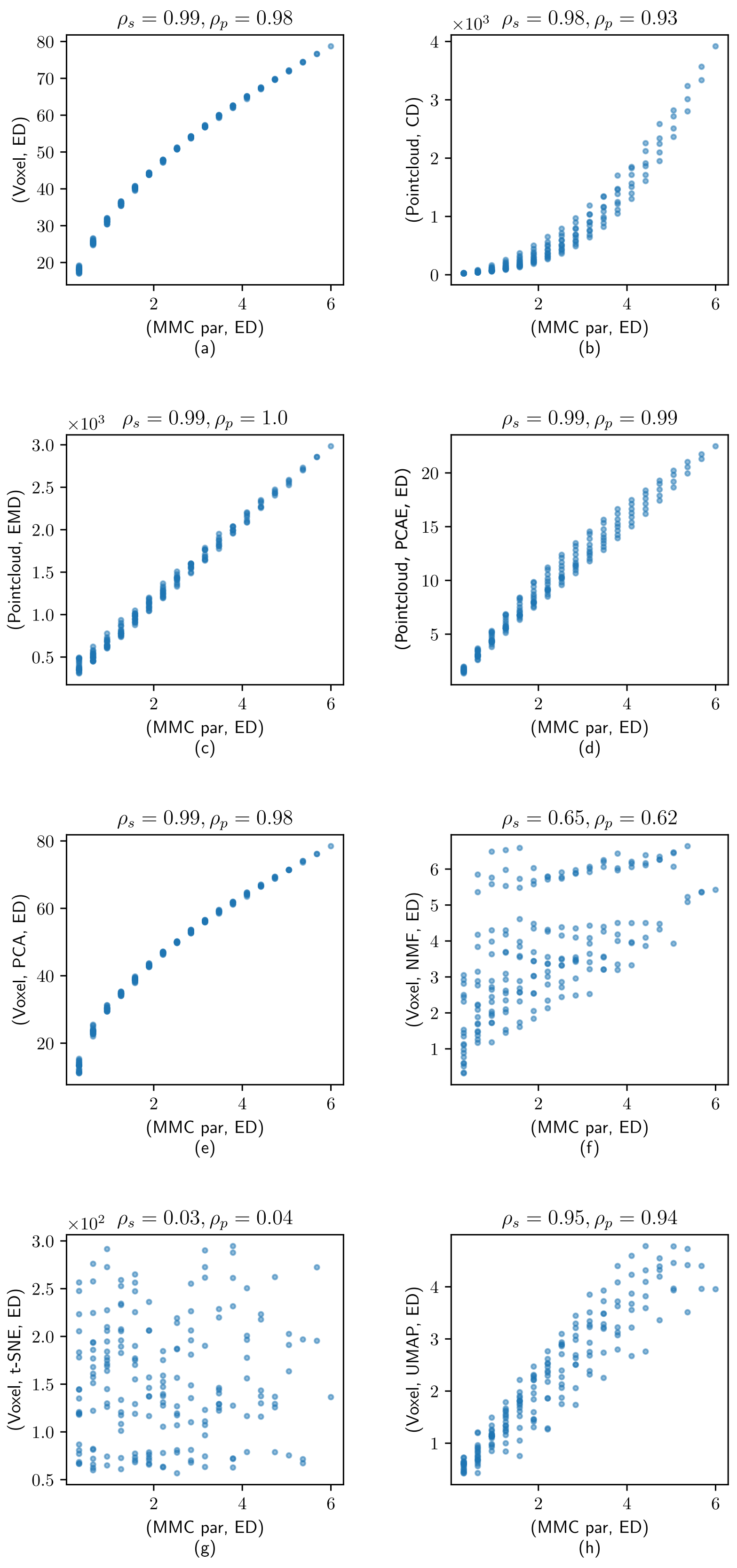


Figure 30 Metric correlations for the beam-elongation dataset: An ellipsoid is elongated by different amounts to yield 20 designs. So, each comparison plot above contains 190 points. Compare different metrics with RM: (MMC par, ED), i.e., the difference in elongation.

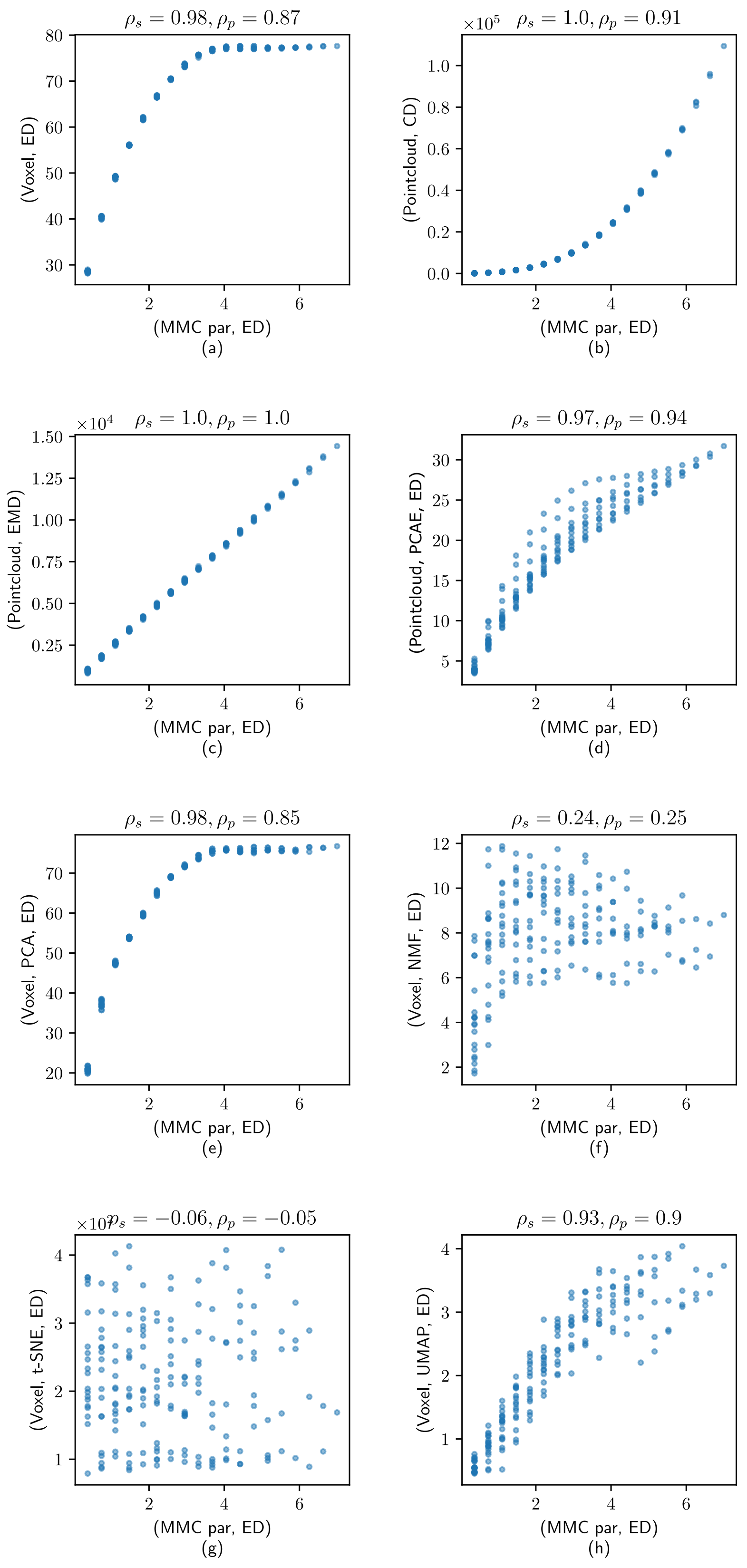


Figure 31 Metric correlation for beam-translation dataset: An ellipsoid is translated by different amounts to yield 20 designs. So, each comparison plot above contains 190 points. Compare different metrics with RM: (MMC par, ED), i.e., the difference in position.

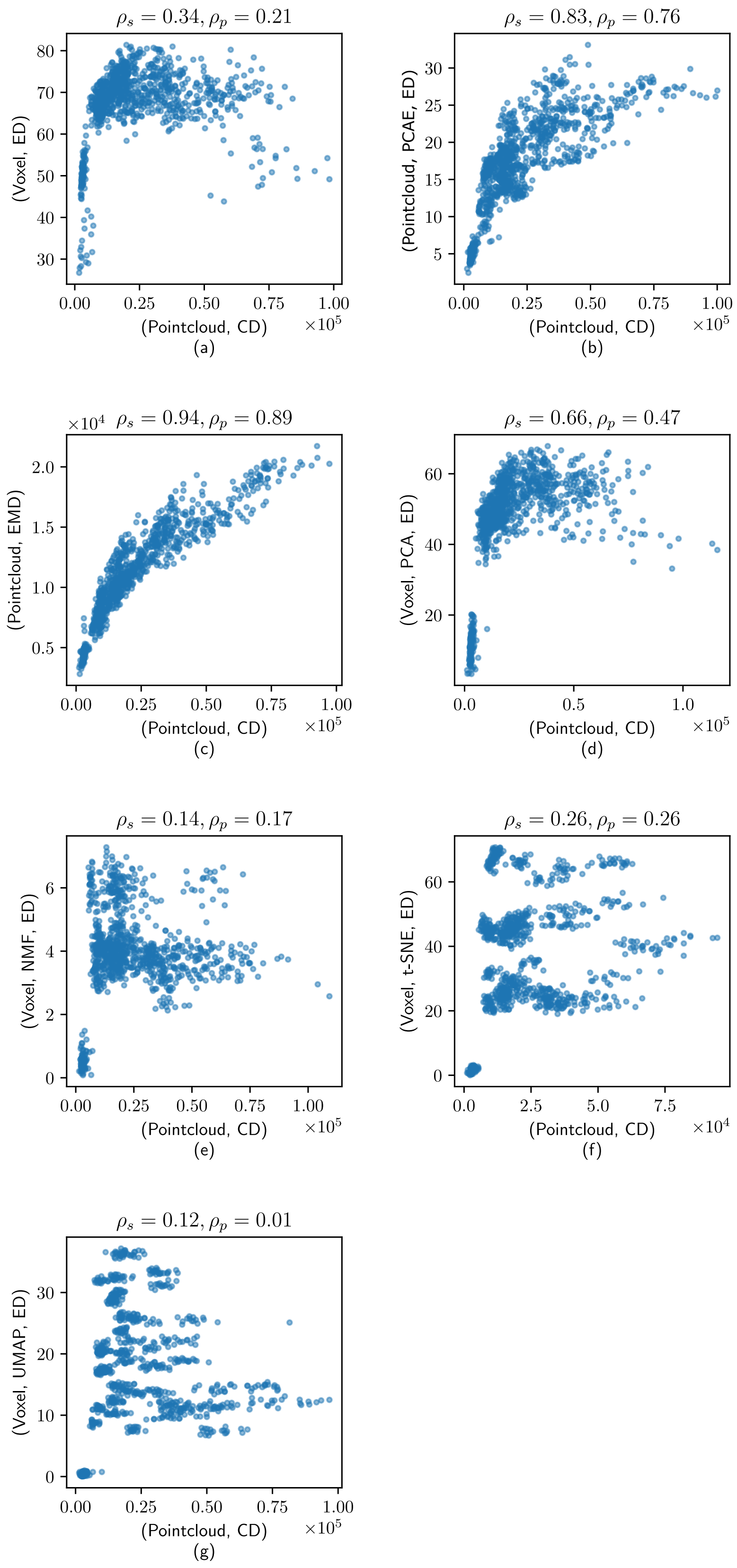


Figure 32 Metric correlations for random topologies dataset. Each correlation measure uses nearly 5×10^5 distance measures. Since (MMC par, ED) is not useful for complex datasets, we only have 7 TMs and 1 RM among the metrics (Table 5).

The three datasets discussed here use an MMC beam with a form factor $q = 2$. We repeated each of the experiments above with other MMC shapes with q ranging from 1 (bipyramids) to 6 (cuboids). For a given q value and dataset, the results are similar. CD (RM) and EMD values are strongly correlated ($\rho_s \geq 0.99$). (Pointcloud, PCAE, ED) is comparable to CD ($\rho_s \in [0.96, 1]$). Measured values of voxel distance and CD are highly correlated ($\rho_s \in [0.98, 1]$). Except for t-SNE, dimensionality reduction techniques can be used with voxel data to yield useful **g** ($\rho_s \in [0.95, 1]$). The metric correlation of RM and t-SNE is quite low ($\rho_s \in [0, 0.3]$) even for these simple datasets. Since the results do not strongly depend on the MMC form factors, we expect similar results with other shapes such as cones, plates, etc.

For complex datasets, geometrical differences cannot be quantified by using (MMC par, ED) since more than one MMC parameter is varying and MMC parameters may be redundant as discussed previously. So, for the following datasets, RM is (Pointcloud, CD) which is proven to be effective in 3D object recognition [30]. For brevity, we do not show any metric comparison plots for the single-cube truss dataset and three-cube truss dataset; we report only the metric correlation values. However, for the more complex dataset with random topologies, we show the metric comparison plots as well.

Single-cube truss dataset: Voxel distance has a low metric correlation with CD as RM ($\rho_s = 0.14$). By contrast, EMD and CD values are strongly correlated. ED with PCAE latent code also has a strong metric correlation with CD ($\rho_s = 0.96$). Other dimensionality reduction techniques with voxel data yield less useful **g** with $\rho_s \in [0.4, 0.6]$, which is still better than that of voxel data ($\rho_s = 0.14$).

Three-cube truss dataset: For this dataset, the metric correlation between voxel distance and CD is low ($\rho_s = 0.55$). Using dimensionality reduction techniques, we can extract **g** and improve the metric correlation with CD ($\rho_s \in [0.6, 0.8]$). Among these, ED with PCAE latent code has the strongest metric correlation with CD ($\rho_s = 0.93$).

Random topologies: Among the synthetic datasets, this dataset has the most complex and diverse topologies. As expected, voxel distance, unlike EMD ($\rho_s \approx 1$), differs from CD. Using dimensionality reduction methods with voxel data does not result in meaningful **g** whereas the autoencoder extracts meaningful **g** ($\rho_s = 0.88$). The results indicate that Euclidean distance in latent code is similar to CD, which is the loss function used to train PCAE for extracting the latent code.

Summary: The key finding from this section is that the **g** vector obtained by PCAE is the most useful compared to the **g** vectors obtained by other dimensionality reduction techniques such as PCA, NMF, t-SNE, and UMAP with voxel data. Metric correlation studies show the deficiencies of the voxel distance and Euclidean distance with **g** vectors obtained from voxel data using dimensionality reduction techniques. By contrast, the pointcloud metrics CD and EMD can effectively identify geometrical differences. Furthermore, PCAE can reduce pointcloud data to yield a latent code whose Euclidean distance values strongly correlates with those obtained by CD. Since the last few datasets have complex and diverse topologies with similar metric correlations, we expect similar results with TO designs as demonstrated in Chapter 8.

7.2.4. Using clustering performance

For evaluating the clustering performance of different metrics, we use the dataset with random topologies (Section 7.1). As discussed previously, the dataset contains diverse topologies from 50 subclasses, where each subclass has 20 designs. The clustering methods (OPTICS and k-means) and evaluation measures (precision, AMI, F1-score) used here are already mentioned in Section 5.4.2.

First, we use k-means clustering to identify the different classes in the dataset. Since the k-means method implicitly assumes ED of its input as the metric, we cannot use the k-means method with general metrics such as CD and EMD. However, as discussed previously, dimensionality reduction methods can yield **g** vectors which can be used as input to k-means clustering. For example, (Voxel, PCA, ED) reduces voxel data using PCA to yield a **g** vector which can be used as input for the k-means method since (Voxel, PCA, ED) uses ED to compare **g** vectors. Since the dataset contains 50 subclasses, we set $k = 50$ in the k-means method.

Table 6 compares the clustering performance of the metrics that use ED in the last stage. PCA yields better input/**g** vectors than NMF since it yields higher clustering scores. Using PCAE, UMAP, or t-SNE, we can identify all the subclasses accurately using the k-means method. The dimensions of resulting **g** vectors of PCAE, t-SNE, and UMAP are 128, 2, and 2 respectively. So, t-SNE and UMAP coordinates can be used to visualize the clusters in a 2D plane. Furthermore, the clusters in PCAE latent code can also be visualized in 2D, if UMAP or t-SNE is used to reduce the latent code from 128 dimensions to 2. For example, Figure 33 shows the clusters using UMAP reduction of PCAE latent code.

Table 7 shows the classification accuracy of different metrics using OPTICS clustering, which is a density-based algorithm that identifies the appropriate number of clusters automatically. For this dataset, the k-means method has better clustering performance than OPTICS for a given **g**. However, the order of performance is the same as before for NMF, Voxels (Voxel, ED), PCA, UMAP, and PCAE (Table 6). CD and EMD perform better than UMAP, which outperforms both PCA and t-SNE. PCAE latent code is still the best input for clustering.

Table 6 Classification accuracy of different metrics with k-means clustering ($k = 50$).

Metric	**Precision**	**F1 score**	**AMI score**
(Voxel, NMF, ED)	0.88	0.86	0.92
(Voxel, ED)	0.96	0.95	0.98
(Voxel, PCA, ED)	0.97	0.96	0.98
(Voxel, UMAP, ED)	1.00	1.00	1.00
(Voxel, t-SNE, ED)	1.00	1.00	1.00
(Pointcloud, PCAE, ED)	1.00	1.00	1.00

Table 7 Classification accuracy of different metrics with OPTICS as the clustering method.

Metric	**Precision**	**F1 score**	**AMI score**
(Voxel, NMF, ED)	0.60	0.68	0.61
(Voxel, ED)	0.66	0.64	0.80
(Voxel, t-SNE, ED)	0.69	0.77	0.67
(Voxel, PCA, ED)	0.72	0.78	0.75
(Voxel, UMAP, ED)	0.74	0.82	0.68
(Pointcloud, CD)	0.86	0.88	0.90
(Pointcloud, EMD)	0.86	0.88	0.90
(Pointcloud, PCAE, ED)	0.94	0.95	0.95

7.3. TO Results

As discussed in Section 7.2, PCAE can extract latent code, which is low-dimensional and can be used to identify designs with similar geometrical structures. As demonstrated in the previous section, PCAE is very useful since ED with the latent code is still meaningful. In this section, we qualitatively evaluate it using dataset 3 discussed in Section 3.3.3. The dataset uses a cubic design space for TO with randomized load and boundary conditions.

Since Euclidean distance with latent code is meaningful, k-means clustering can be used with PCAE latent code as input. The corresponding clusters can be visualized using UMAP with PCAE latent code as input (Figure 34). Furthermore, since ED in the UMAP space still makes sense (see previous results), close points in the UMAP space indicate similar designs (Figure 34). Here, the clusters are not well-separated in the UMAP space, indicating that the designs gradually change in topology across the dataset. In the plot, a few samples of each cluster are located in a faraway cluster since it is not always possible to reduce high dimensional data to

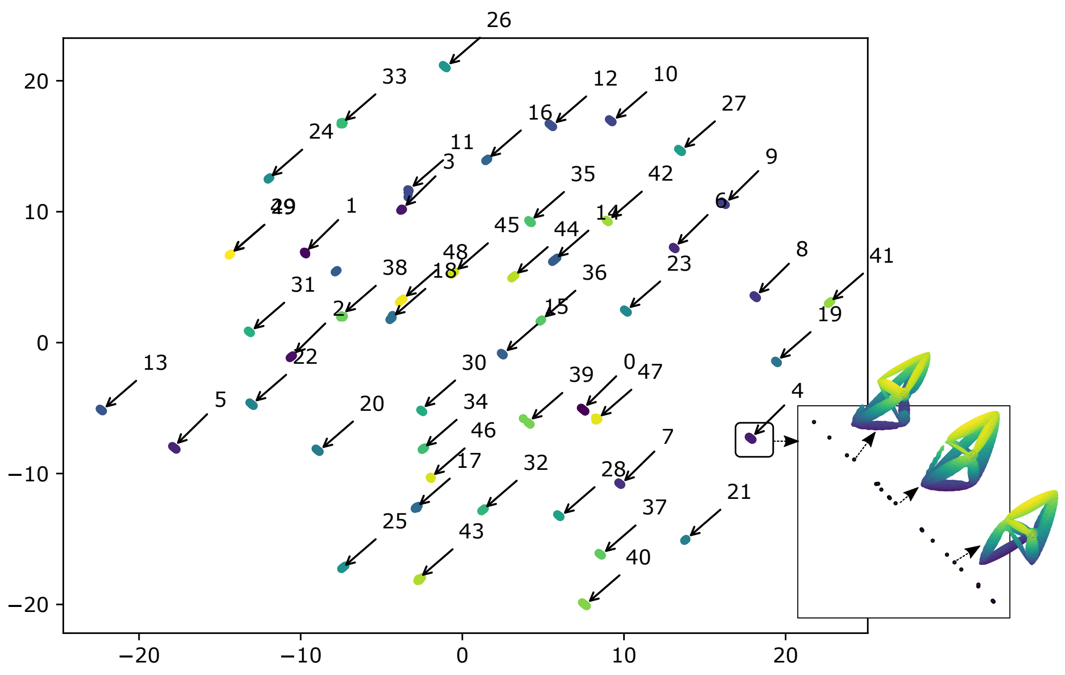


Figure 33 A UMAP visualization of clusters. Each cluster is a set of designs, which are represented here as points in a 2D plane. The clusters are numbered from 0 to 49 and are annotated in the figure using arrows. The inset shows cluster 4 and three sample designs from it. Designs and clusters are colored to ease visualization.

low dimensions without loss of information [85, 86].

Figure 35 shows samples from each cluster in each row. Each cluster has a medoid which is defined as the cluster representative ($\mathbf{d}_r$) [23]. Given a metric, the design in the cluster with the smallest (or largest) distance to $\mathbf{d}_r$ is referred to as the closest (or farthest) design in the figure. For selected clusters, we also show the corresponding $\mathbf{d}_r$ along with its closest and farthest design using (Pointcloud, PCAE, ED) as the metric. For brevity, (Pointcloud, PCAE, ED) is referred to as AD here. In each cluster, the closest design has a similar material distribution as $\mathbf{d}_r$, unlike the farthest design. However, the load conditions of similar designs are not related.

We also indicate the *relative distance* between the designs and $\mathbf{d}_r$ measured using CD and AD. For a given cluster C, the relative distance between any two designs $\mathbf{d}_1$ and $\mathbf{d}_2$ is the actual distance between $\mathbf{d}_1$ and $\mathbf{d}_2$ divided by the farthest distance, i.e., the distance between farthest design in C and $\mathbf{d}_r$. The figure also shows that the distances measured by (Pointcloud, PCAE, ED) and (Pointcloud, CD) are similar.

Table 8 summarizes the benefits of the metrics studied in this chapter. According to clustering accuracy, CD, EMD, UMAP, t-SNE, and latent code are ranked high. According to the quality of geometric features, (Pointcloud, PCAE, ED) is ranked the highest, since Euclidean distance with latent code is similar to the reference metric CD. Metrics such as CD and EMD do not

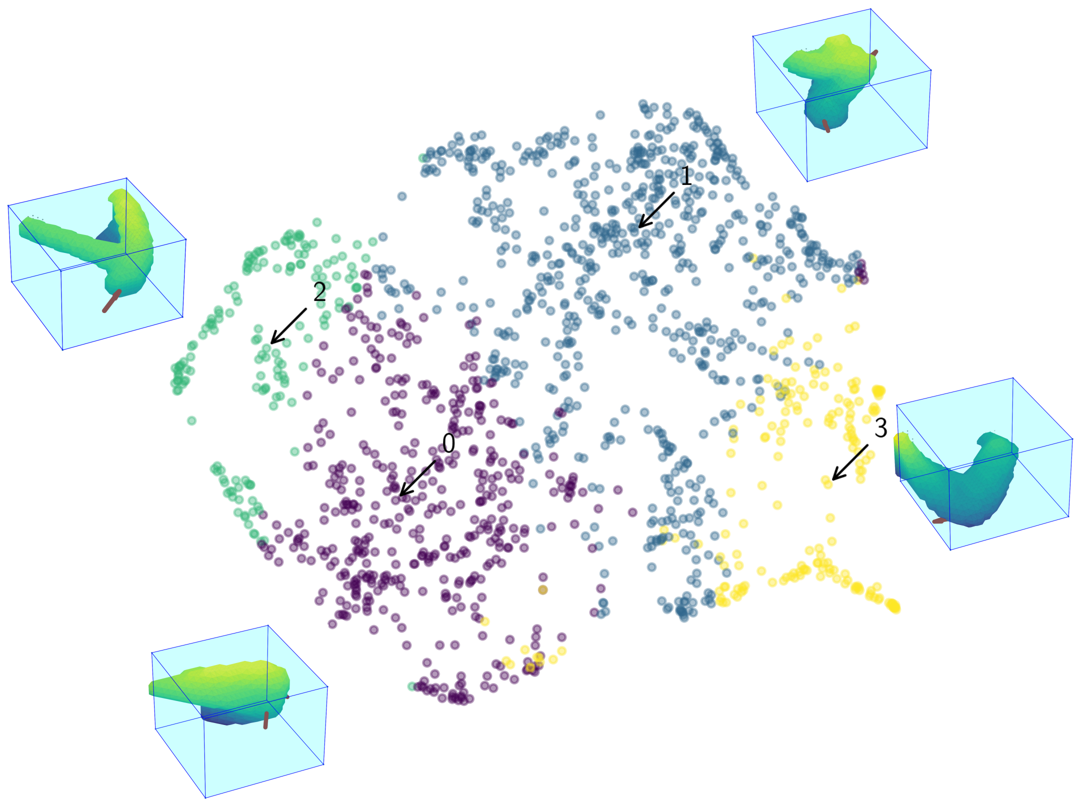


Figure 34 Clusters in the TO design-set are visualized in 2D using UMAP reduction of PCAE latent code. UMAP values are not shown along axes since they do not yield additional insights. k-means method identifies 4 clusters (labeled 0-3). The representative design for each cluster is shown near the respective cluster. The identified clusters are not separated, which is often the case with engineering data.

yield $\mathbf{g}$ and are hence ranked low for this criterion. According to ease of calculation (i.e., computational effort), voxel distance ranks the highest but it cannot be used for clustering or as a meaningful $\mathbf{g}$ vector. CD and EMD are moderately expensive. Similar to PCA or NMF, PCAE requires a training stage to extract the latent code, but once the models are trained, the feature extraction is inexpensive. According to clustering ease, metrics that use ED at the last stage rank the highest, e.g., the k-means method can be used to quickly calculate clusters using latent code as input. Clustering with CD and EMD requires that all pair-wise distances are calculated and more expensive clustering algorithms such as OPTICS need to be used. Considering these criteria, latent code is an ideal choice for clustering and analyzing TO results.

7.4. Discussion

In this chapter, we addressed the problem of finding geometric features ($\mathbf{g}$) that can be used to explore TO results with varying topology, size, shape, and orientation in a design space. We investigated different machine learning methods to generate features that can be accurately and inexpensively used to cluster designs based on geometrical structure. Furthermore,

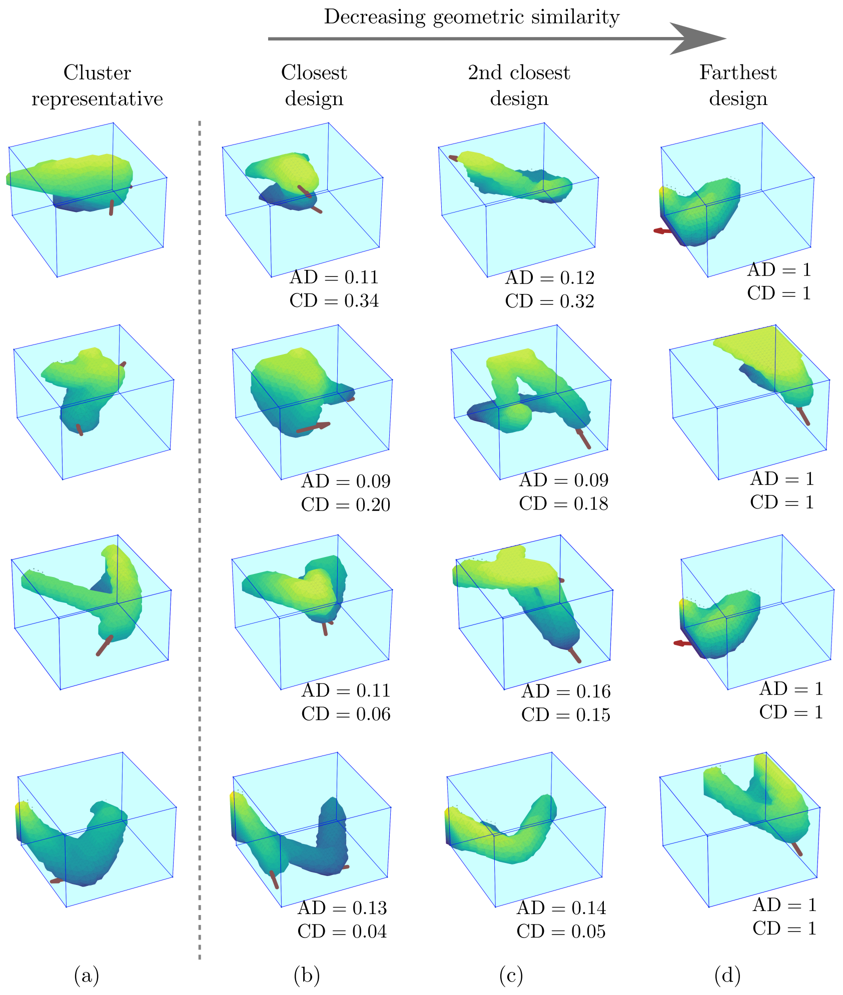


Figure 35 TO designs similar to cluster representatives found using latent code (Figure 34). The first column (a) contains the four design representatives. For each cluster representative, three designs from the same cluster are shown in a row with decreasing similarity in columns b-d. For each design, we show the corresponding loads as brown arrows. In each row, we measure the relative distance from the representative (a) to each design in columns b-d using latent code (AD) and chamfer distance (CD).

we evaluated if a particular **g** is meaningful, i.e., if the Euclidean distance with **g** measures geometrical differences.

One of the challenges is to evaluate different **g** obtained using PCA, NMF, t-SNE, UMAP, or PCAE. We proposed to choose the **g** based on two properties: Euclidean distance with **g** is meaningful and the clustering with **g** as input is accurate. Furthermore, we proposed new test datasets that are topologically more complex than the public datasets such as those provided in ShapeNet [127]. The new datasets are publicly available at Zenodo [143]. To validate ED with **g**, we propose a novel method to compare it with a demonstrated reference metric such as chamfer distance (CD). For different labeled datasets, the clustering performance is evaluated using the AMI score and by measuring the classification errors using precision and F1-score (after label remapping).

Table 8 Advantages and disadvantages of different metrics. For each attribute column, values are put into three categories: low (L), medium (M), and high (H).

Metric	Clustering accuracy	Quality of *geometric features*	Metric calculation ease	Clustering ease
(Voxel)	L	L	H	H
(Voxel, NMF)	M	M	M	H
(Voxel, PCA)	M	M	M	H
(Voxel, t-SNE)	H	L	M	H
(Voxel, UMAP)	H	M	M	H
(Pointcloud, CD)	H	L	L	L
(Pointcloud, EMD)	H	L	L	L
(Pointcloud, PCAE, ED)	H	H	M	H

In the experiments with different datasets, we derive the following conclusions. More meaningful features are obtained by reducing voxel data using NMF or more preferably by using PCA. Euclidean distance with t-SNE is not useful, unlike the UMAP features. Euclidean with latent code is the most meaningful among the reduced features. For clustering, UMAP, t-SNE, and latent code have high accuracy.

Autoencoder (PCAE) can extract meaningful geometric features as latent code (or latent code), which is good for clustering. Furthermore, measured values of CD and Euclidean distance with latent code are strongly correlated. Since CD is used to train PCAE, the latter learns to imitate its loss function (CD in our case). It would be interesting to see if an autoencoder can learn useful features for any metric. From our experiments, we conclude that latent code can be used to address our two objectives: cluster similar structures in TO results as well as build metamodels of a design optimization method.

8. Cooperative Topology Optimization

In this chapter, we demonstrate the iSMO method (Chapter 6) using a simple MOP and two MTO problems: a static compliance problem with a cantilever plate and a crashworthiness optimization problem with a simply supported beam. For each of the problems, clusters are identified in the initial set of solutions and iSMO generates new solutions in each of the clusters. Furthermore, we investigate the effect of different regressor and classifier metamodels on iSMO. The hyperparameters of the metamodels are as described in Section 6.1.4.

iSMO can be used with any MOP that is based on a weighted-sum approach. For example, the first problem (Section 8.1) has only two decision variables and the weighted-sum method can directly combine the two objectives to yield a single scalarized objective, which can be solved using any global optimizer. By contrast, the MTO problems—the subject of this thesis—have millions of decision variables and are efficiently solved using SEW-HCA [27] for a given set of weights for objectives. For each MTO, we cluster the initial solutions either based on performance or geometry. For each cluster $\mathbb{C}_i$, iSMO then generates new solutions that are similar to $\mathbb{C}_i$ by controlling the input weight-vectors to the SEW-HCA.

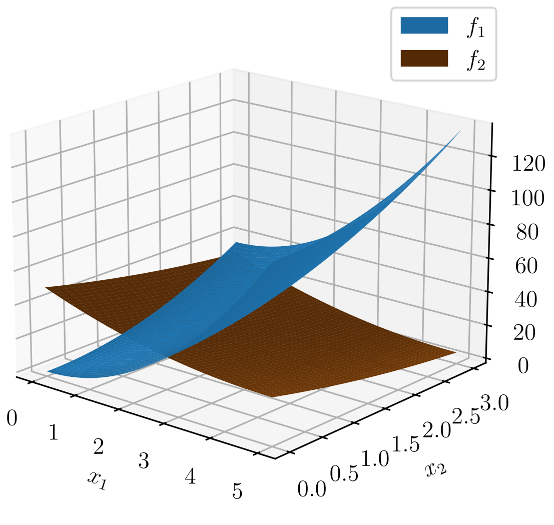


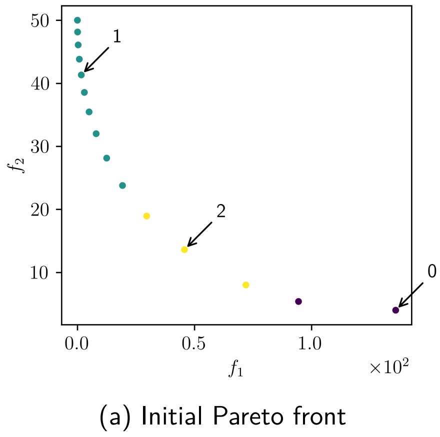
Figure 36 BNH problem: 3D surface plots showing the effect of decision variables on the two objective functions f_1 and f_2.

8.1. Simple Example: BNH Problem

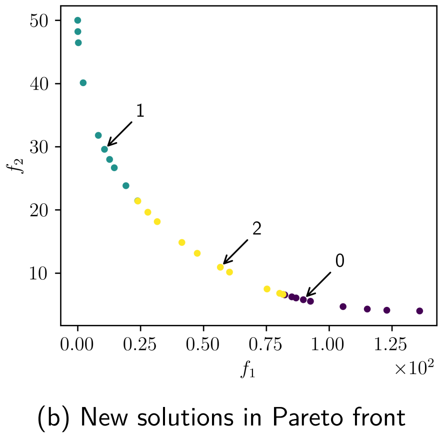
The BNH [144] problem has two decision variables $\mathbf{x} = [x_1, x_2]$, two minimization objectives $\mathbf{f} = [f_1, f_2]$ and two constraints. For simplicity, we remove the constraints in the BNH problem to yield the following test MOP for iSMO. Figure 36 plots the objective functions f_1, f_2.

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}) = 4x_1^2 + 4x_2^2, \\ \text{Minimize} \quad & f_2(\mathbf{x}) = (x_1 - 5)^2 + (x_2 - 5)^2, \\ \text{subject to} \quad & 0 \le x_1 \le 5, \\ & 0 \le x_2 \le 3. \end{aligned} \tag{8.1}$$

For a given set of weights $\mathbf{w} = [w_1, w_2]$, where $w_1 \ge 0$ and $w_2 = 1 - w_1 \ge 0$, the scalarized objective $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{f} = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x})$ is minimized using the differential evolution (DE) to find the optimal $\mathbf{x}$ [145]. We use the DE algorithm implemented in SciPy Python library with the following hyperparameters: DE strategy is "best1bin", maximum iteration limit is 100, population size is 15, mutation value is uniformly sampled from 0 to 2, crossover probability is 0.7, and Latin hypercube sampling is used to initialize the population [145, 146].



(a) Initial Pareto front



(b) New solutions in Pareto front

Figure 37 iSMO generates new solutions (b) for each of the clusters in (a). Solutions are colored according to their cluster ids (labels).

The initial set of solutions is obtained using a set of uniformly sampled weights, $\mathbf{W} = \{\mathbf{w}_i\}_{i=1}^N$, obtained using the Das-Dennis approach [57]. The corresponding Pareto set (Figure 37a), given by $\mathbf{F} = \{\mathbf{f}(\mathbf{w}_i)\}_{i=1}^N$, is slightly non-uniform in the objective space since the range of the two objective functions is different. These initial solutions are clustered based on objective values using the k-means method ($k = 3$). Since the data samples are contiguous in this example, k-means partitions the dataset instead of separating the data into well-defined clusters. iSMO can now generate new and diverse solutions in any preferred cluster.

In this example, iSMO uses metamodels that predict the objectives and cluster labels from weights using a regressor and a classifier respectively. iSMO then ensures that the new solutions belong to the selected cluster according to the metamodels. For example, Figure 37b shows the new solutions in the Pareto front obtained using a Gaussian process regressor and a k-nearest neighbors ($k = 1$) classifier. While training the metamodels, the redundancy in input weights is removed by normalizing the weights and excluding one of the weights before training the metamodels (Section 6.1).

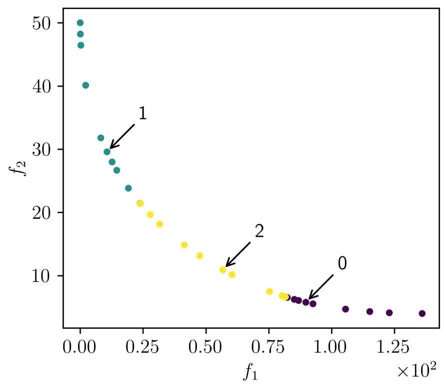
The regression and classification algorithms need to be chosen based on the given data. Here, we experiment with multiple regression models: Gaussian process regressor (GPR), random forests regressor, k-nearest neighbors (KNN) regressor, and linear regressor [18]. Table 9 shows the accuracy of the regressors as well as the *preferredness* and *diversity* of the resultant solutions (Section 6.2). The R2 score is used to measure the k-fold cross-validation (CV) score as well as the training score of regressors. The preferredness and diversity of new solutions are measured using the silhouette score (s_{sil}) and the diversity score (s_{div}) respectively (Section 6.2). Since the clustering is based on Euclidean distance in the objective space, the same metric is used to evaluate s_{sil} and s_{div}. Furthermore, each set of new solutions yields a different s_{sil} (or s_{div}) score; we report only the average scores for s_{sil} (or s_{div}) across different preferred clusters.

Table 9 A comparison of different regressor models

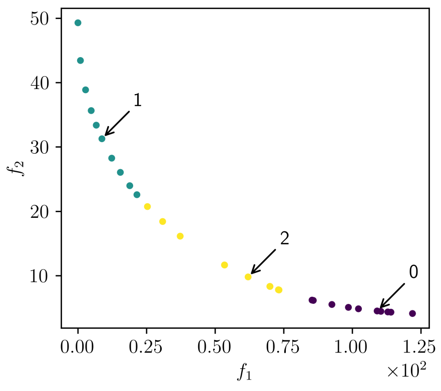
Regressors	**R2 score (CV)**	**R2 score (training)**	s_{sil}	s_{div}
GPR	0.99 ± 0.02	1.00	0.51	-0.05
Random forests regressor	0.92 ± 0.08	0.99	0.59	-0.08
KNN regressor	0.52 ± 0.75	0.90	0.29	-0.07
Linear regressor/least squares	-0.25 ± 2.11	0.84	0.24	-0.07

For each regressor, the resultant new solutions in all the clusters are shown in Figure 38. In this experiment, a k-nearest neighbors classifier ($k = 1$) is used with all the regressors. As shown in Table 9, the Gaussian process regressor (GPR) has the best model performance since the CV score and the training score are high. Other models have reduced performance and the gap between CV and training score is higher which indicates overfitting. GPR also has the best diversity score. Since the clusters seem contiguous, any $s_{sil} \in [0, 1]$ is acceptable so that the solutions are either on the border or interior of clusters. So, all the models seem to perform well based on the preferredness of solutions. Based on these results, GPR is the best choice for this example problem.

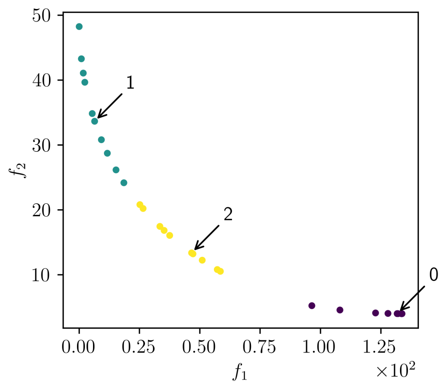
We also investigate if it is better to remove the redundancy of weights before using them as input for the different regressors. For GPR, if the weights are only normalized and no weights are removed, the CV score drops to 0.85 and s_{div} drops to -0.08. For other regressors, s_{sil}



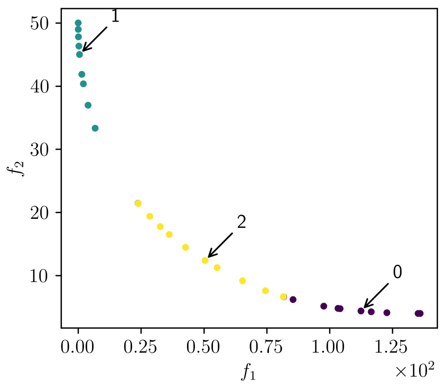
(a) Gaussian process regressor



(b) Random forests regressor



(c) k-nearest neighbors regressor



(d) Linear regressor

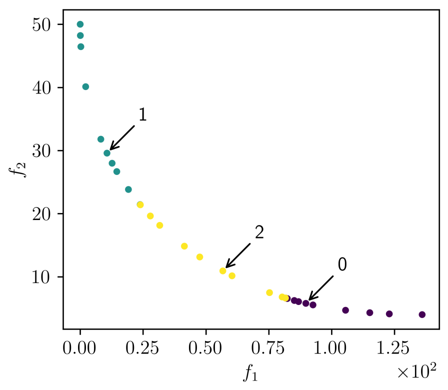
Figure 38 iSMO yields different solutions when different regressor models are used.

scores do not change but s_{div} is even lower. So, it is beneficial to remove the redundancy in weights before training a regressor.

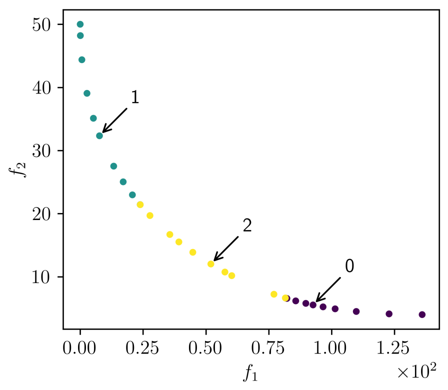
Table 10 A comparison of different classifier models

Classifiers	**F1 score (CV)**	**F1 score (training)**	s_{sil}	s_{div}
k-nearest neighbors classifier	0.89 $\pm$ 0.14	1.00	0.51	-0.05
Random forests classifier	0.86 $\pm$ 0.19	1.00	0.47	-0.06
SVC	0.73 $\pm$ 0.23	0.93	0.49	-0.08
Logistic regression classifier	0.53 $\pm$ 0.00	0.53	-0.03	-inf
Linear SVC	0.56 $\pm$ 0.05	0.64	0.35	-inf

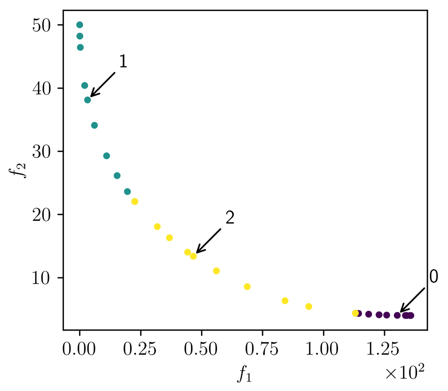
Next, we fix the regressor model to GPR in iSMO and compare in Table 10 the use of different classifier models: k-nearest neighbors (KNN) classifier with $k = 1$, random forests classifier, support vector machine classifier (SVC) with radial basis functions (RBF) as kernel, logistic



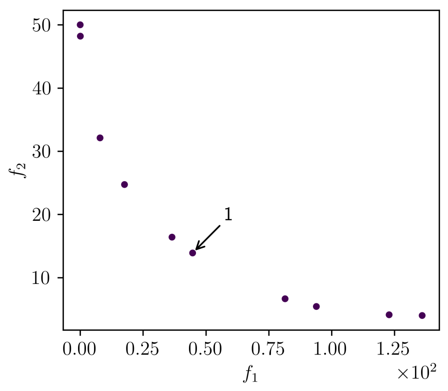
(a) k-nearest neighbors classifier



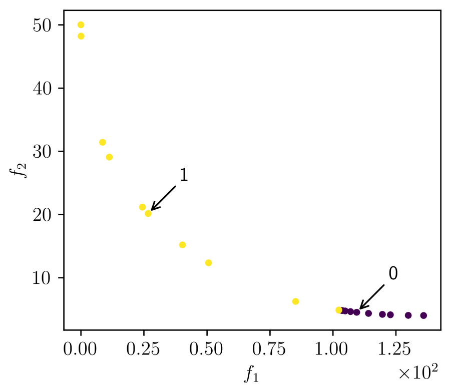
(b) Random forests classifier



(c) SVC



(d) Linear regression classifier



(e) Linear SVC

Figure 39 iSMO yields different solutions when different classifier models are used.

regression classifier (regression to labels), and linear SVC. Note that the random forests classifier is a different algorithm from the random forests regressor. The same is true for the KNN regressor and the KNN classifier. Figure 39 shows the resultant new solutions in all the clusters. The classifier performance is measured using an F1 score with CV and training sets. Here, we treat each class equally and report the average F1 score without weighing each class with its sample size. This is called the "F1 macro" score in scikit-learn [130]. KNN and random forests have the highest accuracy as well as the best s_{sil} and s_{div} scores. Logistic regression and linear SVC yield no new solutions for some of the clusters (Figures 39d,e); hence, $s_{div} = -\infty$.

Based on these results, we choose the Gaussian process regressor and the KNN classifier ($k = 1$) to build the metamodels in iSMO, especially if the initial solutions are not many. For larger and/or more complex datasets, one may choose more accurate models based on cross-validation scores.

8.2. Topology Optimization using iSMO

In this section, we test iSMO using two exemplary MTO problems. In a real-world application, the user chooses one of the clusters and iSMO generates new solutions for the preferred cluster. Here, we use different clusters as input to iSMO and recommend new solutions. For each MTO problem, we perform clustering either based on performance or geometrical structure and iSMO generates new solutions for each of the possible clusters.

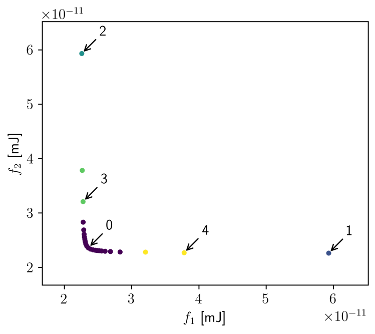
Performance clustering
The solutions can be clustered in the objective space. If the objective values are used as input to clustering algorithms such as k-means or OPTICS, Euclidean distance in the objective space is used implicitly as the metric for clustering.

Geometrical clustering
As discussed in Section 4.3.1, a pointcloud autoencoder (PCAE) is useful in finding *geometrical clusters*, i.e., clusters of designs with similar geometrical structure. PCAE converts each design into a pointcloud and extracts a latent code, which can be inexpensively used for clustering. For the following datasets, the latent code is extracted using the following process. Each optimized design is defined by a set of elements with relative densities $\rho \in [0, 1]$. The design is converted to a surface mesh using the marching cubes algorithm [98] with a threshold of $\rho = 0.1$. The mesh is then converted to a 2048-dimensional pointcloud. Using an initial set of pointclouds, we train a PCAE model to extract a 128-dimensional latent code, which is used as input for clustering.

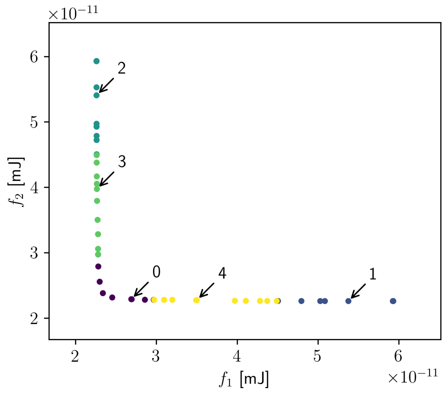
8.2.1. Test Case 1: Cantilever Plate with Two Static Loads

As described in the first example in Section 3.3, a cantilever plate is optimized for two objectives using SEW-HCA. A set of weight-vectors are uniformly sampled and an initial set of solutions is generated. Using the initial solutions, iSMO trains a Gaussian process regressor to predict the objectives $\mathbf{f} = [f_1, f_2]$ from a given set of positive weights $\mathbf{w} = [w_1, w_2]$, where $w_1 + w_2 = 1$. A k-nearest neighbor classifier with $k = 1$ is trained to predict cluster labels p from $\mathbf{w}$. k-fold cross-validation ($k = 5$) for the regressor yields a CV score $R^2 = 0.83 \pm 0.14$; the training score is 0.99. Although there is some overfitting, this model gives the best score and the least amount of overfitting compared to other regressors: random forests, a linear regressor, and a support vector regressor (RBF kernel).

Performance clusters are obtained using k-means method with objective values as input (Figure 40a). For each performance cluster, iSMO is used to generate more solutions within the cluster. Figure 40b shows that the new solutions seem to belong to their preferred clusters. The average silhouette score $s_{sil} = 0.48$ and the diversity score s_{div} = -0.09, which are lower compared to the best scores for the BNH case (previous section). The lower scores can be attributed to the lower accuracy of the metamodels. However, the solutions still seem to be well-distributed in the objective space (Figure 40b).



(a) Initial set of solutions. Solutions are colored based on their cluster.

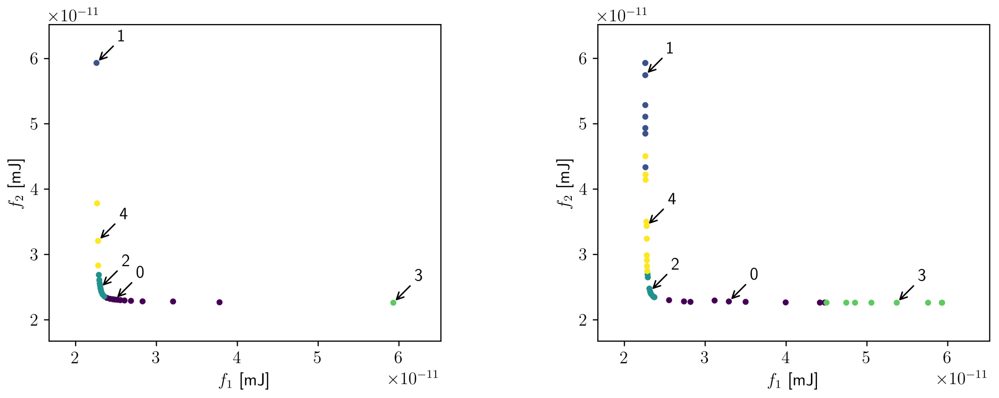


(b) New solutions for each cluster in (a)

Figure 40 For each **performance cluster** (a), iSMO generates new solutions (b). Subfigure b shows only the new solutions.

Geometrical clusters are obtained using the k-means method with the PCAE latent code as input (Figure 41). Once again, iSMO generates new solutions in each cluster with an average silhouette score $s_{sil} = 0.41$ and a diversity score $s_{div} = -0.11$, which are lower compared to performance clusters. Although the metamodel accuracy is similar in this case, the preferred clusters are closer to each other, and hence are more challenging for iSMO.

Figure 42 shows the representative solutions in each geometrical cluster and the corresponding new solutions generated by iSMO. In fact, for a preferred cluster, we show the most dissimilar design to the cluster representative as measured by the chamfer distance; we observe that the structural similarity between them is very high. Note that each optimized design consists of high-density elements with $\rho \in [0.1, 1]$ and low-density elements with $\rho < 0.1$. Even after convergence, a few optimized designs have unsupported high-density (colored brown) parts embedded in the low-density material since the latter can still support loads, albeit with high



(a) Initial set of solutions. Solutions are colored based on their cluster.

(b) New solutions for each cluster in (a)

Figure 41 For each **geometric cluster** (a), iSMO generates new solutions (b). Subfigure b shows only the new solutions.

compliance. In practice, one may interpret such designs as multi-material composites or post-process designs to remove low-density material and any unsupported high-density parts. In this thesis, we interpret such designs as composites, and the pointcloud representation uses the boundary of high-density material.

Figure 42 contains pairs of designs that are almost mirror images of each other along the horizontal axis. For example, prototypes of clusters 1 and 3 are one such pair. These two clusters are on the opposite ends of the Pareto front in Figure 41 and hence, contain designs that are preferentially optimized for the top load or the bottom load. Since these two loads are symmetric w.r.t the horizontal axis, the designs from one cluster are almost mirror images of the other. For each geometric cluster, Figure 42 shows the most dissimilar solution generated by iSMO, which typically lies on the boundary of that cluster. Therefore, the new solution may be similar to the designs in the neighboring cluster. For example, the new solution for cluster 0 is very similar to the cluster 3 representative.

For this dataset, iSMO recommends useful new solutions for both performance and geometrical clusters. Even for sparse input clusters, the new solutions belong to the preferred cluster and are well-distributed, especially in comparison to the initial solutions. Next, we discuss a more complex MTO example with a crash load.

8.2.2. Test Case 2: Simply Supported Beam with a Crash and a Static load

As described in the second example in Section 3.3, a simply supported beam is optimized for two objectives using SEW-HCA. Once again, a set of weight-vectors are uniformly sampled and an initial set of solutions are generated using SEW-HCA. Since SEW-HCA is a heuristic algorithm, a few solutions are not Pareto-optimal but are near the Pareto front. Using the initial solutions, a Gaussian process regressor and a k-nearest neighbor classifier ($k = 1$) are trained to predict objectives and cluster labels respectively. Gaussian process regressor yields

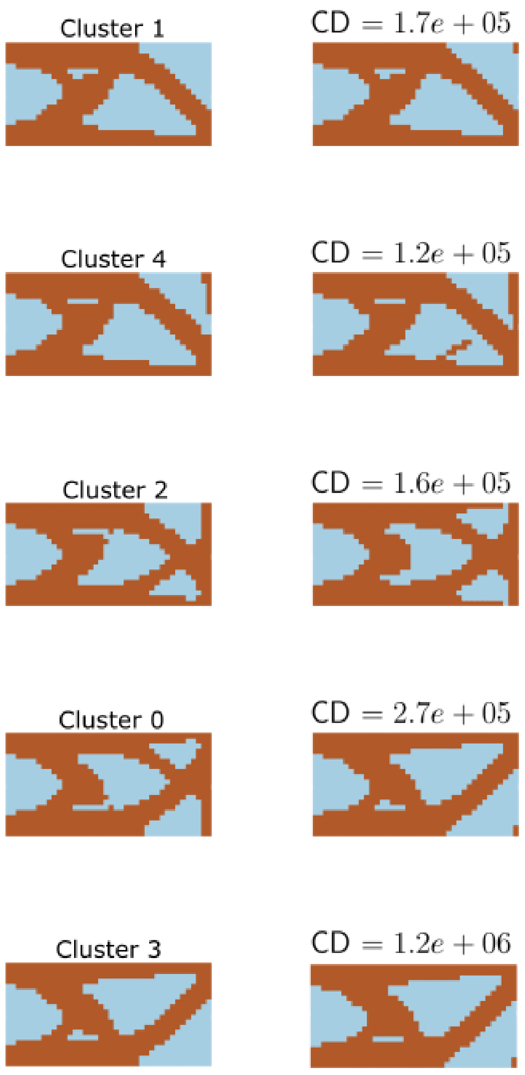
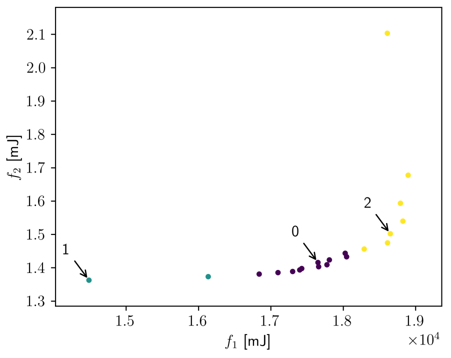


Figure 42 Comparing the geometrical structures of initial and new solutions generated by iSMO. In each design, low-density and high-density elements are indicated by blue and brown color respectively. In each row, we show a cluster representative in the initial solutions (left image), followed by its most dissimilar design—as measured by chamfer distance (CD)—in the new solutions generated for that cluster (right image).

the best CV score of 0.90 ± 0.04 and a training score of 0.99 when compared to random forests, a linear regressor, and a support vector machine (RBF kernel).

Similar to test-case 1, k-means clustering is used to find performance and geometrical clusters. Figure 43a shows the performance clusters in the objective space. Figure 43b shows the new solutions generated by iSMO which seem to belong to their preferred clusters in the objective space, which is confirmed by the values of $s_{sil} = 0.53$ and $s_{div} = -0.07$.

Figure 44a shows the geometrical clusters obtained using PCAE (Section 4.3.1). Figure 44b


(a) Initial set of solutions. Solutions are colored based on their cluster.

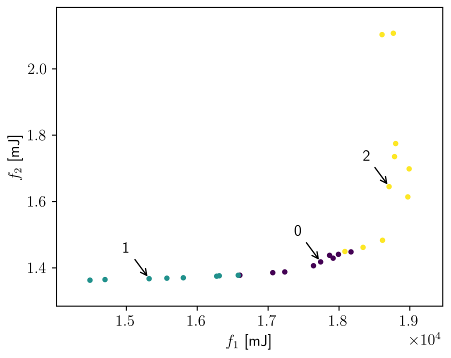
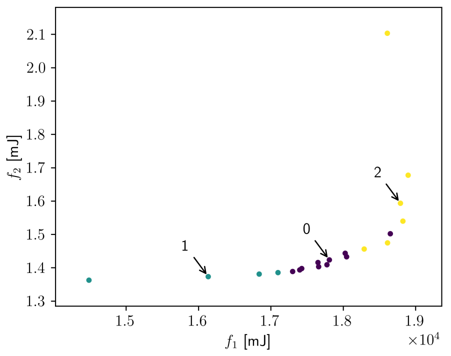

(b) New solutions for each cluster in (a)

Figure 43 For each **performance cluster** in test case 2 (Figure 4), iSMO generates new solutions. Note that since we are maximizing f_1 and minimizing f_2, the Pareto front is towards the bottom-right corner.


(a) Initial set of solutions colored according to the geometric cluster.

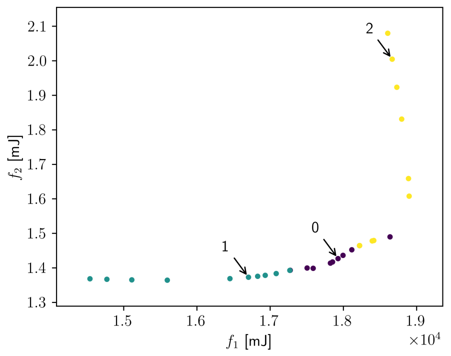

(b) New solutions for each cluster in (a)

Figure 44 For each **geometric cluster** in test case 2 (Figure 4), iSMO generates new solutions.

shows that the new solutions generated by iSMO seem to belong to their preferred clusters, at least as seen in the objective space. Compared to the new solutions in the performance clusters, the score for new solutions in geometric clusters is lower with $s_{sil} = 0.34$ and $s_{div} = -0.08$. The silhouette score is worse because of the slight mixing of clusters 0 and 2 at their boundary (Figure 44b), while the diversity score is relatively similar.

For this dataset, similar structures tend to have similar performance. Even when cluster "0" is split in the objective space (Figure 44), the new solutions are similarly split in the Pareto front. In Figure 45, we compare the geometrical structure of the initial and new solutions.

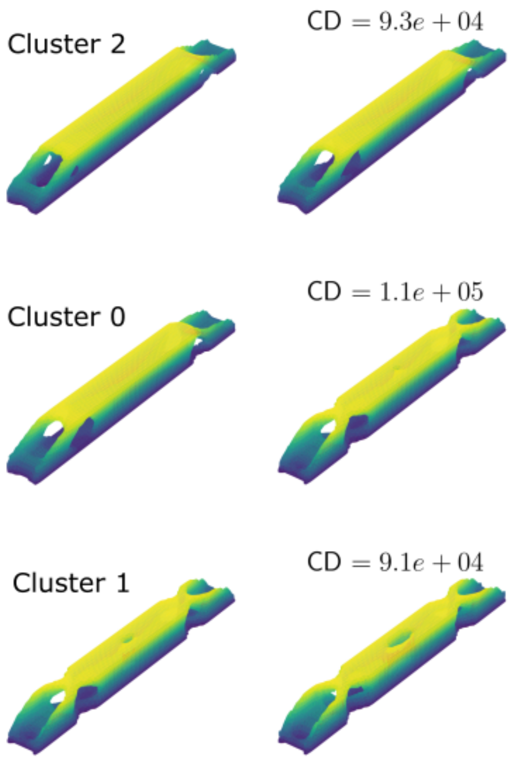


Figure 45 Comparing the geometrical structures of initial and new solutions generated by iSMO. Each design is shown using the surface mesh obtained using the threshold 0.1. Similar to Figure 42, we compare the representative in each preferred cluster with its most dissimilar, new solution. Note that in cluster 0 (middle row), the most dissimilar design has a different structure since it is on the boundary.

8.3. Discussion

In this chapter, we addressed the challenge of minimizing expensive simulations in multi-objective topology optimization using iSMO (Chapter 6). As an example, we consider SEW-HCA, which optimizes the design given a set of weights for multiple objectives as input. Each run of SEW-HCA requires expensive simulations. By contrast, iSMO iteratively generates new solutions in the preferred regions using metamodels, which are much cheaper to evaluate than simulations.

We analyzed the performance of iSMO with 2-objective problems using quantitative measures, namely, silhouette and diversity scores (Section 6.2). Furthermore, we evaluate the results qualitatively through visual inspection of the 2D Pareto front. The initial solutions generated by SEW-HCA for uniformly sampled weights are reasonably uniform owing to its preference scaling strategy. We can then choose a preferred cluster among the performance/geometrical clusters in the sparse initial dataset and generate new solutions using iSMO in the chosen cluster. The *preferredness* of the new solutions is measured using the silhouette score and is found to be reasonably high. From diversity scores, the new solutions are found to be well-distributed along the Pareto front. Since we used two-objective MTO problems in this chapter, the new solutions can be visually evaluated on the Pareto front. Interestingly, given a clustering of initial solutions, if new solutions are generated for all the clusters using iSMO, the resulting solutions are better distributed on the Pareto front than the initial solutions.

For each cluster in the objective space (i.e., performance cluster), it is easy to see from the Pareto front that the new solutions generated by iSMO belong to the preferred cluster. This is reflected in the relatively high silhouette score ($s_{sil} \in [0.5, 0.6]$). We also have silhouette scores in the same range for geometric clusters, indicating that the new solutions belong to the preferred clusters. The diversity scores s_{div} range from $[-0.05, -0.11]$, where 0 score indicates the ideal diversity. From visual inspection of the Pareto front, this range for diversity scores seems to be good. However, in practice, the desired diversity score is dependent on the application.

From the results, we believe iSMO is an interesting method to efficiently generate new solutions in the preferred regions for solutions to multi-objective optimization problems. Furthermore, iSMO can avoid expensive optimization runs by building metamodels (or surrogate models). In the following chapter, we explore the solutions for complex optimization problems using iSMO.

9. Engineering Example: Hood Optimization

In this chapter, iSMO is used to cooperatively generate preferred solutions for an engineering example. A hood model is optimized for three crash objectives using SEW-HCA. Once again, a set of weight-vectors are uniformly sampled and an initial set of solutions near the Pareto front are generated using SEW-HCA. Given the initial solutions, performance and geometrical clusters are found using the methods discussed in Chapter 4. For each selected cluster, we demonstrate the use of iSMO to generate new solutions.

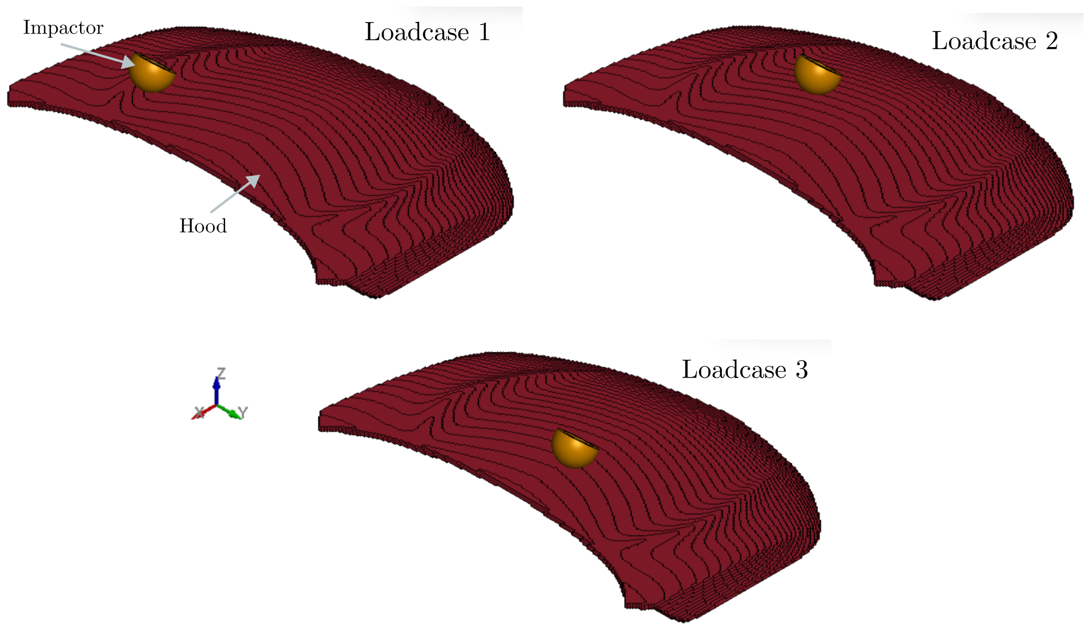


Figure 46 A hood model is optimized for three crash load configurations where an impactor crashes into the hood in three different locations. MOP is to maximize the crash energy absorbed for the loadcases.

9.1. Problem Description

A hood design, modeled using a non-linear elastoplastic material, is optimized for three crash objectives, where each objective function f_i ($i \in \{1,2,3\}$) is the negative of total internal energy absorbed for the crash loadcase i (Figure 46). For a given crash load i, we simulate the crash for a certain number of time steps. For each element, we find the maximum internal energy absorbed across the time steps. By summing the maximal internal energies of all the elements, we obtain f_i. SEW-HCA minimizes the objective $f = \sum_{i=1}^{3} w_i \; f_i = \mathbf{w} \cdot \mathbf{f}$ for a given weight-vector $\mathbf{w} = [w_i]_{i=0}^{3}$ by trying to homogenize the energy absorbed, i.e., $-f$. Fixed nodes in the model are shown in Figure 47. The absorbed internal energies are calculated using an explicit LS-DYNA solver. For a given set of weights for the objectives, SEW-HCA iteratively optimizes the design by calculating the objective values for each load case separately. For more details on SEW-HCA, see Chapter 3.

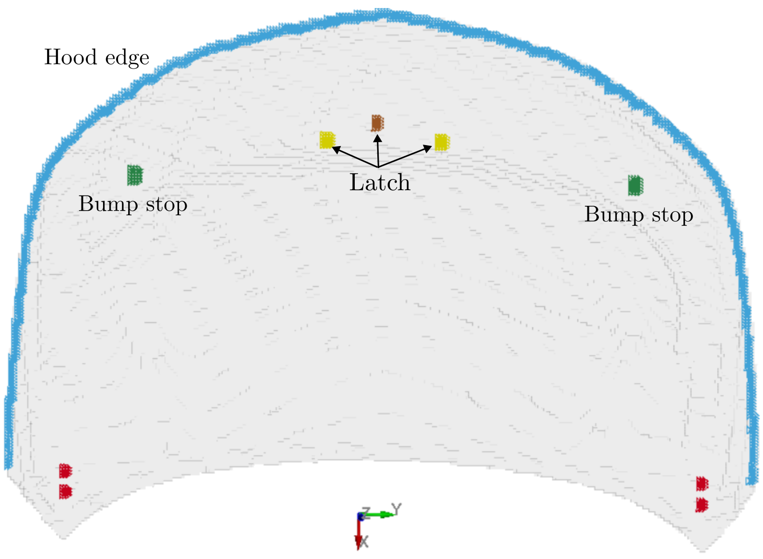


Figure 47 Boundary conditions: multiple nodes in the colored locations (blue, red, green, yellow, brown) are constrained to have zero displacements.

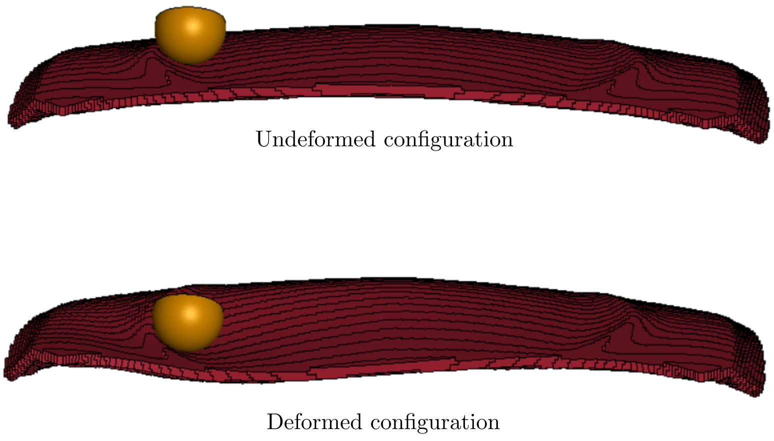


Figure 48 Deformation in the hood with crash loadcase 1 at the last time step ($t = 0.01\text{s}$).

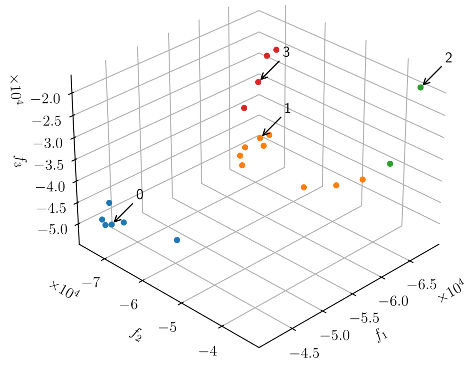
The hood model has an approximate span of $1.73\ \text{m} \times 1.25\ \text{m}$ along x- and y-axes respectively, and contains $195,830$ solid elements. The thickness of the hood skin is approximately 25mm with an average of 4 elements across the skin. The material properties and hyperparameters for SEW-HCA are as in test-case 1 (Section 8.2.1) except for the following changes. A smaller move limit of 0.05 is chosen for stabilizing the optimizer. A volume fraction of 0.3 is chosen with a filter radius of $50\ \text{mm}$. The number of iterations of SEW-HCA is limited to 25, given that a complete optimization run lasts 24 hrs with our computing resources and we have reasonable convergence in objectives after 25 iterations. A uniform gravity of $9.8\ \text{m/s}^2$ is used. The crash load is applied using a hemispherical rigid impactor with a diameter of $150\ \text{mm}$ and a mass of $0.5\ \text{kg}$. The impact velocity is $9.9\ \text{m/s}$ in the z direction and the simulation time is 0.01s. Figure 48 shows a sample deformation for the first crash loadcase.

We first build metamodels needed by iSMO for predicting objective values from weight-vectors. Gaussian process regressor is trained to predict objective values from weight-vectors, which yields a training score of 0.98 and a CV score $s_{CV} = 0.57$. The low CV score is because of very few training samples, where the CV split strongly affects the score. Gaussian process regressor performs better compared to the scores obtained for other models: random forests ($s_{CV} = 0.36$), a linear regressor ($s_{CV} = -0.11$), and a support vector machine with RBF kernel ($s_{CV} = -0.08$).

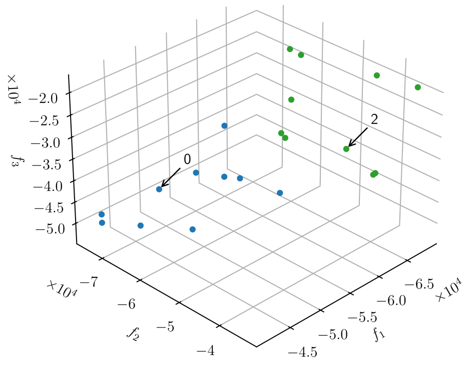
Next, we use iSMO to generate new solutions in selected performance and geometric clusters. Performance clusters are obtained by clustering solutions using objective values as input. Geometrical clustering uses the autoencoder latent code as input. For more information on these two kinds of clustering, see Chapter 4.

9.2. Performance Clustering

Figure 49a shows the performance clusters in the objective space. Note that even for a uniform sampling of weight-vectors using the Das-Dennis approach [57], SEW-HCA is unable to generate a uniform set of solutions. For illustrating the iSMO method, two of the clusters, labeled 0 and 2 are chosen and new solutions are generated (Figure 49b). For the generation of new solutions, iSMO uses metamodels with weight-vectors as input to predict objective-vector and cluster labels using a Gaussian process regressor and k-nearest neighbor classifier ($k = 1$) respectively. The objective predictor has a training accuracy of 0.99, a CV score of 0.62 ± 0.54 with a median of 0.80. The label predictor has a training accuracy of 1, a CV score of 0.51 ± 0.17 with a median of 0.5. The scores indicate that the metamodels overfit the data, which results in the spread of new solutions outside their preferred clusters to some extent (Figure 49b).



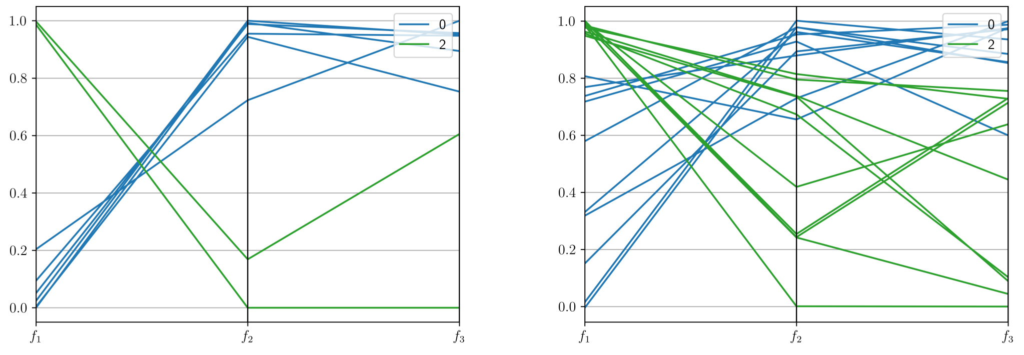
(a) Initial set of solutions



(b) New set of solutions

Figure 49 For selected performance clusters 0 and 2 in initial solutions (Subfigure a), new solutions are generated using iSMO (Subfigure b).

The silhouette score $s_{sil} = 0.15$ of the new solutions is low while the diversity score $s_{div} = -0.15$ is close to 0 as desired. Even if the new solutions seem to be in the correct neighborhood of the reference set of solutions, the spread of solutions into other clusters and the resultant low silhouette score might be due to the inadequacy of SEW-HCA to find the global minimum for the given weight-vector.



(a) Initial set of solutions (b) New set of solutions

Figure 50 Parallel coordinates plot of initial and new solutions. Normalized objective values are shown along the vertical axes. For a given cluster label, the connecting lines have similar changes in slope, and the intersection points along each objective share a common region.

Figure 50 compares the objective values for the initial and new solutions using a visualization tool called parallel coordinates plot [88]. Before plotting, values for each objective f_i are normalized using the min/max values of f_i in the initial solutions. This allows us to compare the two plots since the values for each objective are scaled and shifted by the same pair of values. The initial and new solutions have a similar pattern in their plots even if the latter occupy a wider region for each objective.

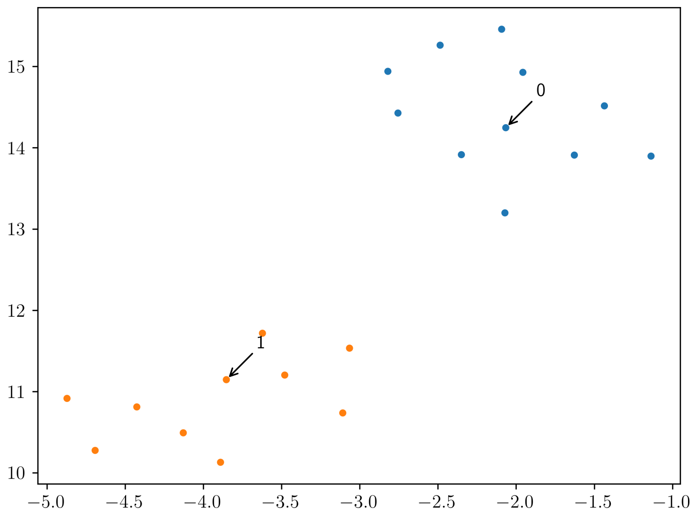
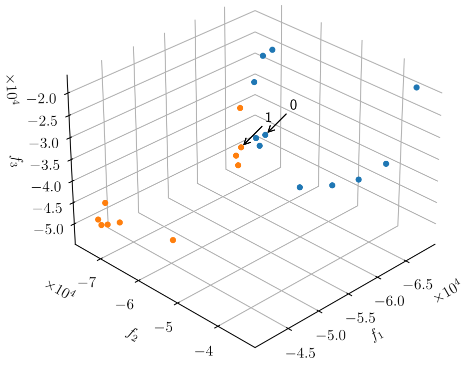


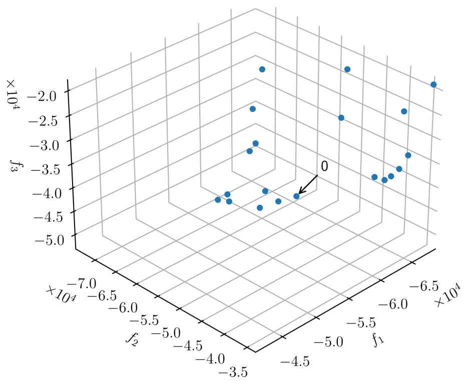
Figure 51 UMAP visualization of the geometrical clusters shows that there are two well-separated clusters. Each solution is shown here using its two UMAP coordinate values.

9.3. Geometric Clustering

The initial set of solutions is clustered based on the geometric structure using an autoencoder, as discussed in Chapter 4. Pointcloud autoencoder (PCAE) is used to extract a low-dimensional latent code of size 128 from a 3D pointcloud of size 2048 used to represent the surface of each geometry. The latent code can then be used as input for clustering based on geometrical structure. Here, we use the initial set of solutions as input for training PCAE. An alternative training approach is to use a larger dataset such as the publicly available Carhoods10k dataset [147] and then extract the latent code. This might avoid overfitting to the given dataset and lead to a better PCAE model, which as mentioned before can be verified by using CV score (Section 6.1.4).



(a) Initial set of solutions



(b) New set of solutions

Figure 52 Given an initial set of solutions and its geometric clusters (Subfigure a), iSMO is used to generate new solutions in a selected cluster 0 (Subfigure b).

UMAP visualization of the clusters shows that there are two well-separated clusters in the data, as shown in Figure 51. For UMAP visualization, we use UMAP to reduce the latent code (128 dimensions) to 2D data. We use UMAP instead of t-SNE or PCA because it preserves inter-cluster distance as well as achieves a non-linear reduction of data while preserving the cluster relations in the data.

Since we expect two clusters in the data, we use k-means clustering method with $k = 2$ and latent code as input to find the two geometrical clusters, which are shown in Figure 52a. Samples of geometries from each of the two clusters are shown in Figure 53. The optimized geometry is shown here using only the elements with relative density $\rho \geq 0.05$. Note, that some of the geometries contain groups of elements that are connected through low-density elements. So, such concepts are to be post-processed to remove unconnected parts and obtain contiguous structures before further development in the design process. Figure 53 shows that the solutions in the two clusters have different structures and topologies.

To illustrate the use of iSMO, the cluster 0 is chosen as the preferred set of solutions, and iSMO is then used to generate new solutions in cluster 0, which are shown in Figure 52b. iSMO

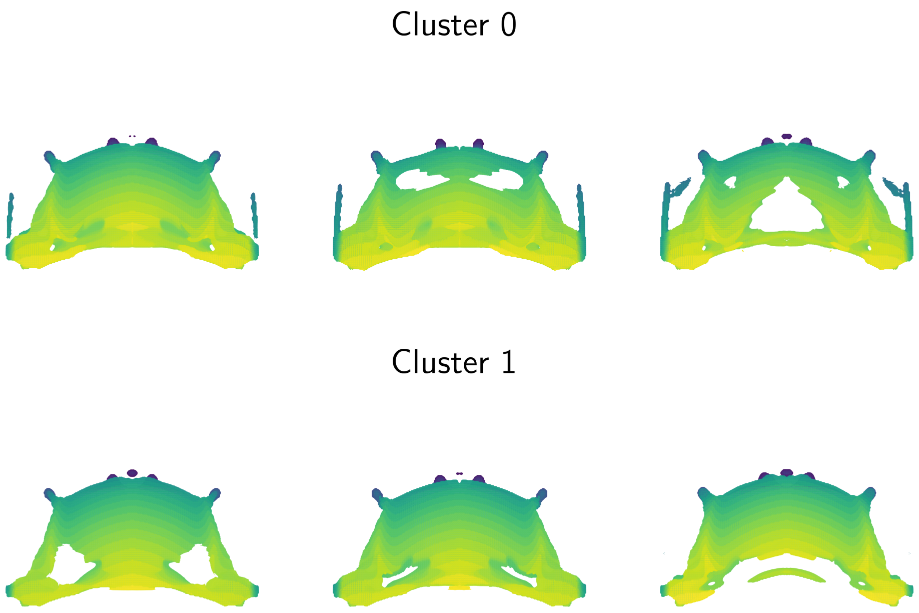


Figure 53 Top view of the optimized designs in the initial set of solutions. In the top row, samples in cluster 0 are shown, which have an arm-like structure on the left/right sides of the subfigures. In the bottom row, samples in cluster 1 are shown, which have empty holes at the bottom side.

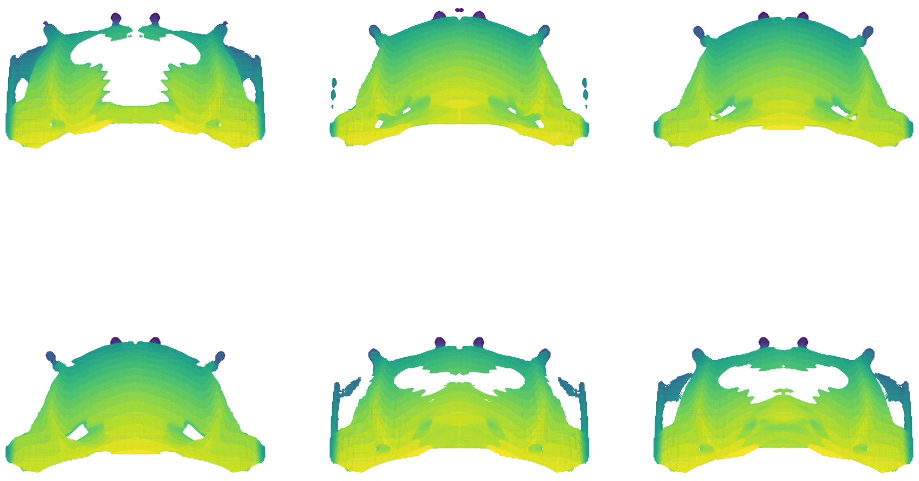


Figure 54 Top view of the optimized designs in the new set of solutions generated by iSMO in cluster 0. Most of the samples (4 of 6 subfigures) tend to have arm-like structures to the left/right sides similar to samples in cluster 0, shown in Figure 53.

uses the same metamodel to predict the objective-vector from weight-vector. Since geometric cluster labels are different from performance clusters, a new metamodel is built to predict cluster labels from weight-vector. The *preferredness* and *diversity* scores for the new solutions are $s_{sil} = 0.11$ and $s_{div} = -0.08$ respectively. Figure 54 shows that the new solutions have a similar structure to the initial solutions in the selected cluster 0. A few of the samples on the cluster boundary have a structure similar to the other cluster (label $= 1$), which is indicated by the lower silhouette score.

9.4. Discussion

Similar to the previous chapter, iSMO is used again to minimize expensive simulations while using SEW-HCA. We analyzed the performance of iSMO with a 3-objective hood optimization problem using silhouette and diversity scores (Section 6.2). Since we only have 3 objectives, we can evaluate the results qualitatively through visual inspection of the 2D Pareto front. The initial solutions generated by SEW-HCA for uniformly sampled weights are not uniform despite the preference scaling strategy of SEW-HCA. However, we can still choose a preferred cluster among the performance/geometrical clusters in the sparse initial dataset and generate new solutions using iSMO in the chosen cluster.

For each cluster in the objective space (i.e., performance cluster), it is easy to see from the Pareto front that the new solutions generated by iSMO are near the preferred cluster, with some solutions spread into the nearby clusters. This is reflected in the relatively low average silhouette score ($s_{sil} = 0.10$) of the new solutions in the selected performance clusters. When a geometric cluster is chosen, new solutions yield a slightly better score (0.15). Since there are only two geometric clusters, it is less challenging for iSMO to generate new solutions compared to performance clustering. The diversity scores s_{div} range from $[-0.15, -0.8]$, where 0 score indicates the ideal diversity. From visual inspection of the Pareto front, this range for diversity scores seems to be good.

From the results, we believe iSMO can efficiently generate new solutions in the preferred regions for solutions to multi-objective optimization problems. iSMO could successfully generate solutions near the selected cluster since we could build metamodels with high accuracy for predicting objectives and cluster labels. However, some of the solutions spread beyond the selected clusters, which might be due to SEW-HCA finding a local minimum for a given weight-vector. Possibly, solutions could be restricted to the preferred clusters by using a smaller move limit. Due to time constraints, we did not investigate this further.

10. Conclusion

Design generation methods such as topology optimization (TO) [1–6] and multiobjective optimization methods [7, 24, 54, 55, 148–150] can be used to yield multitudes of promising concepts for designing and developing products in automotive and aerospace industries. While significant advances in high-performance computing and simulation tools have been made, expensive simulations still hinder the process of TO, which typically requires several simulations to iteratively optimize structural solutions. This problem is exacerbated when multiple objectives need to be simultaneously considered for optimization. Furthermore, each optimization method tends to have hyperparameters such as filter radius in TO that need to be tuned according to user requirements. So, typically, an engineer needs to tune the optimizer and generate numerous solutions, of which a few solutions are selected for further development. In this thesis, we addressed two of the challenges in multi-objective TO: design exploration and expensive simulations.

Designs can be easily explored using clustering, an unsupervised machine learning method that can group solutions into classes, which can be more easily analyzed by an engineer. In Chapter 4, we discussed how some of the methods available in literature can be used to analyze solutions obtained using TO. While several clustering methods are available in literature [22, 75, 151], the metric used to cluster still needs to be tailored based on the application and design requirements. In this thesis, we investigated the use of pointcloud autoencoders developed by Achlioptas et al. [30] to cluster solutions based on geometrical structure. Furthermore, we compared multiple metrics of geometry based on their ability to cluster complex topologies obtained from structural optimization. We published some of this work already [93, 113, 152]. As discussed in Chapters 5 and 7, autoencoders show promise in geometrical clustering and enable engineers to find interesting concepts in the solutions.

In the literature, two general strategies are available to avoid expensive simulations in design generation methods: (i) generate only the preferred solutions [64, 153–155], and (ii) build metamodels (surrogate models) to predict the behavior of an optimizer [33, 150]. We propose a novel method called iSMO which builds metamodels for SEW-HCA [27], a weight-based approach for multiobjective TO. Given a reference set of solutions, iSMO uses an evolutionary algorithm to generate weight-vectors that will result in diverse solutions in the objective space. Furthermore, iSMO can restrict the weight-vectors using metamodels to predict if they will result in a solution *similar* to the reference set of solutions. In this work, the reference set is chosen using one of the clusters obtained by clustering solutions with a given metric. This allows us to simultaneously tackle the challenges of finding and generating desired solutions while minimizing expensive simulations. Finally, we propose new methods for evaluating *preferredness* and *diversity* of new solutions based on the silhouette score [79] and the spacing metric [137] respectively. The proposed method iSMO has been partially published by us

[143].

The initial solutions generated by SEW-HCA for uniformly sampled weights are reasonably spread owing to its preference scaling strategy. Using iSMO, we could choose a preferred cluster among the performance or geometrical clusters in the sparse initial dataset and successfully generate preferred solutions. The *preferredness* of the new solutions is found to be high, provided the optimization method and the metamodels are accurate. From diversity scores, the new solutions are found to be well-distributed in the objective space. Since we used two-objective and three-objective MTO problems in this thesis, the new solutions can be visually evaluated on the Pareto front as well. Interestingly, given a clustering of initial solutions, if new solutions are generated in all the clusters using iSMO, the resulting solutions are better distributed on the Pareto front than the initial solutions obtained by the uniform sampling of weight-vectors.

From the results, we believe clustering is a valuable tool for identifying interesting subsets of solutions without supervision. We propose the use of iSMO to efficiently generate new solutions in the preferred regions for solutions to multi-objective optimization problems. Furthermore, iSMO can avoid expensive optimization runs by building metamodels (or surrogate models). Although we use iSMO with a weight-based approach for multiobjective optimization, it can be easily extended to handle any other optimizer with different inputs and hyperparameters. Furthermore, the reference set of solutions could be obtained by methods other than clustering. We only need to build a classifier to judge if the solutions are desirable or not. Such cooperative optimizers with surrogate models could greatly improve the applicability of topology optimization in the industry.

List of Figures

List of Tables

Bibliography

[1] M. P. Bendsøe and O. Sigmund. *Topology optimization*. Springer Berlin Heidelberg, 2004. DOI: 10.1007/978-3-662-05086-6.

[2] K. Liu and A. Tovar. "An efficient 3D topology optimization code written in Matlab". In: *Structural and Multidisciplinary Optimization* 50.6 (2014), pp. 1175–1196. DOI: 10.1007/s00158-014-1107-x.

[3] M. Bujny, N. Aulig, M. Olhofer, and F. Duddeck. "Identification of optimal topologies for crashworthiness with the evolutionary level set method". In: *International Journal of Crashworthiness* 23.4 (2017), pp. 395–416. DOI: 10.1080/13588265.2017.1331493.

[4] E. Raponi, M. Bujny, M. Olhofer, N. Aulig, S. Boria, and F. Duddeck. "Kriging-assisted topology optimization of crash structures". In: *Computer Methods in Applied Mechanics and Engineering* 348 (2019), pp. 730–752. DOI: 10.1016/j.cma.2019.02.002.

[5] D. Zeng and F. Duddeck. "Improved hybrid cellular automata for crashworthiness optimization of thin-walled structures". In: *Structural and Multidisciplinary Optimization* 56.1 (2017), pp. 101–115. DOI: 10.1007/s00158-017-1650-3.

[6] F. Duddeck, S. Hunkeler, P. Lozano, E. Wehrle, and D. Zeng. "Topology optimization for crashworthiness of thin-walled structures under axial impact using hybrid cellular automata". In: *Structural and Multidisciplinary Optimization* 54.3 (2016), pp. 415–428. DOI: 10.1007/s00158-016-1445-y.

[7] F. Duddeck. "Multidisciplinary optimization of car bodies". In: *Structural and Multidisciplinary Optimization* 35.4 (2008), pp. 375–389. DOI: 10.1007/s00158-007-0130-6.

[8] T. Borrvall and J. Petersson. "Topology optimization of fluids in Stokes flow". In: *International Journal for Numerical Methods in Fluids* 41.1 (2002), pp. 77–107. DOI: 10.1002/fld.426.

[9] S. Sanogo and F. Messine. "Topology optimization in electromagnetism using SIMP method". In: *COMPEL - The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* 37.6 (2018), pp. 2138–2157. DOI: 10.1108/compel-04-2017-0170.

[10] G. H. Yoon, J. S. Jensen, and O. Sigmund. "Topology optimization of acoustic-structure interaction problems using a mixed finite element formulation". In: *International Journal for Numerical Methods in Engineering* 70.9 (2007), pp. 1049–1075. DOI: 10.1002/nme.1900.

[11] J. Matejka, M. Glueck, E. Bradner, A. Hashemi, T. Grossman, and G. Fitzmaurice. "Dream Lens". In: *Proc. of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, pp. 1–12. DOI: 10.1145/3173574.3173943.

[12] G. Fender, S. Marburg, and F. Duddeck. “Identification of a Set of Candidate Solutions for Optimal Positioning of Damping Layers”. In: *SAE International Journal of Passenger Cars - Mechanical Systems* 9.3 (2016), pp. 987–994. DOI: 10.4271/2016-01-1778.

[13] O. Sigmund and J. Petersson. “Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima”. In: *Structural Optimization* 16.1 (1998), pp. 68–75. DOI: 10.1007/BF01214002.

[14] S. Ramnath, N. Aulig, M. Bujny, S. Menzel, I. Gandikota, and K. Horner. “Load Case Preference Patterns based on Parameterized Pareto-Optimal Vehicle Design Concept Optimization”. In: *12th European LS-DYNA Conference*. 2019, pp. 1–9.

[15] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, and O. Sigmund. “Efficient topology optimization in MATLAB using 88 lines of code”. In: *Structural and Multidisciplinary Optimization* 43.1 (2011), pp. 1–16. DOI: 10.1007/s00158-010-0594-7.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009. 745 pp. ISBN: 0387848576.

[17] S. Bandaru, A. H. C. Ng, and K. Deb. “Data mining methods for knowledge discovery in multi-objective optimization: Part A - Survey”. In: *Expert Systems with Applications* 70 (2017), pp. 139–159. DOI: 10.1016/j.eswa.2016.10.015.

[18] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly UK Ltd., 2019. ISBN: 9781492032649.

[19] M. S. Yousaf, M. Bujny, N. Zurbrugg, D. Detwiler, and F. Duddeck. “Similarity control in topology optimization under static and crash loading scenarios”. In: *Engineering Optimization* 53.9 (2021), pp. 1523–1538. DOI: 10.1080/0305215x.2020.1806257.

[20] W. Zhang, Y. Wang, Z. Du, C. Liu, S.-K. Youn, and X. Guo. “Machine-learning assisted topology optimization for architectural design with artistic flavor”. In: *Computer Methods in Applied Mechanics and Engineering* 413 (2023), p. 116041. DOI: 10.1016/j.cma.2023.116041.

[21] K. Miettinen, J. Hakanen, and D. Podkopaev. “Interactive Nonlinear Multiobjective Optimization Methods”. In: *Multiple Criteria Decision Analysis*. Springer New York, 2016, pp. 927–976. DOI: 10.1007/978-1-4939-3094-4_22.

[22] P. Berkhin. “A Survey of Clustering Data Mining Techniques”. In: *Grouping Multidimensional Data*. Springer-Verlag, 2006, pp. 25–71. DOI: 10.1007/3-540-28349-8_2.

[23] A. Hagg, A. Asteroth, and T. Bäck. “Prototype Discovery Using Quality-Diversity”. In: *Parallel Problem Solving from Nature – PPSN XV*. Springer International Publishing, 2018, pp. 500–511. DOI: 10.1007/978-3-319-99253-2_40.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[25] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. “Multiobjective evolutionary algorithms: A survey of the state of the art”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 32–49. DOI: 10.1016/j.swevo.2011.03.001.

[26] M. Bujny, M. Olhofer, N. Aulig, and F. Duddeck. “Topology Optimization of 3D-printed joints under crash loads using Evolutionary Algorithms”. In: *Structural and Multidisciplinary Optimization* 64.6 (2021), pp. 4181–4206. DOI: 10.1007/s00158-021-03053-4.

[27] N. Aulig, E. Nutwell, S. Menzel, and D. Detwiler. “Preference-based topology optimization for vehicle concept design with concurrent static and crash load cases”. In: *Structural and Multidisciplinary Optimization* 57.1 (2018), pp. 251–266. DOI: 10.1007/s00158-017-1751-z.

[28] D. W. Kelly and M. W. Tosh. “Interpreting load paths and stress trajectories in elasticity”. In: *Engineering Computations* 17.2 (2000), pp. 117–135. DOI: 10.1108/02644400010313084.

[29] L. Song, F. Duddeck, and J. Fender. “Commonality Optimization for Components in Vehicle Families with respect to Crashworthiness Design”. In: *VII Europ. Congr. on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*. 2016, pp. 1–12.

[30] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. “Learning Representations and Generative Models for 3D Point Clouds”. In: *Proc. of the 35th Int. Conf. on Machine Learning*. Vol. 80. Proc. of Machine Learning Research (PMLR). 2018, pp. 40–49.

[31] S. Saha, S. Menzel, L. L. Minku, X. Yao, B. Sendhoff, and P. Wollstadt. “Quantifying The Generative Capabilities Of Variational Autoencoders For 3D Car Point Clouds”. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020. DOI: 10.1109/ssci47803.2020.9308513.

[32] C. H. Chuang, R. J. Yang, G. Li, K. Mallela, and P. Pothuraju. “Multidisciplinary design optimization on vehicle tailor rolled blank design”. In: *Structural and Multidisciplinary Optimization* 35.6 (2007), pp. 551–560. DOI: 10.1007/s00158-007-0152-0.

[33] M. Kiani, I. Gandikota, A. Parrish, K. Motoyama, and M. Rais-Rohani. “Surrogate-based optimisation of automotive structures under multiple crash and vibration design criteria”. In: *International Journal of Crashworthiness* 18.5 (2013), pp. 473–482. DOI: 10.1080/13588265.2013.805294.

[34] A.-B. Ryberg, R. D. Bäckryd, and L. Nilsson. “A metamodel-based multidisciplinary design optimization process for automotive structures”. In: *Engineering with Computers* 31.4 (2014), pp. 711–728. DOI: 10.1007/s00366-014-0381-y.

[35] C. Li and I. Y. Kim. “Topology, size and shape optimization of an automotive cross car beam”. In: *Proc. of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 229.10 (2014), pp. 1361–1378. DOI: 10.1177/0954407014561279.

[36] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

[37] D. Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[38] O. M. Querin, G. P. Steven, and Y. M. Xie. "Evolutionary structural optimisation (ESO) using a bidirectional algorithm". In: *Engineering Computations* 15.8 (1998), pp. 1031–1048. DOI: 10.1108/02644409810244129.

[39] A. Tovar, N. M. Patel, G. L. Niebur, M. Sen, and J. E. Renaud. "Topology optimization using a hybrid cellular automation method with local control rules". In: *Journal of Mechanical Design, Transactions of the ASME* 128.6 (2006), pp. 1205–1216. DOI: 10.1115/1.2336251.

[40] N. P. van Dijk, K. Maute, M. Langelaar, and F. van Keulen. "Level-set methods for structural topology optimization: a review". In: *Structural and Multidisciplinary Optimization* 48.3 (2013), pp. 437–472. DOI: 10.1007/s00158-013-0912-y.

[41] M. Bujny, N. Aulig, M. Olhofer, and F. Duddeck. "Evolutionary level set method for crashworthiness topology optimization". In: *ECCOMAS Congress 2016 - Proc. of the 7th European Congress on Computational Methods in Applied Sciences and Engineering*. Vol. 1. 2016, pp. 309–322. ISBN: 9786188284401. DOI: 10.7712/100016.1814.11054.

[42] G. Allaire, F. Jouve, and A.-M. Toader. "Structural optimization using sensitivity analysis and a level-set method". In: *Journal of Computational Physics* 194.1 (2004), pp. 363–393. DOI: 10.1016/j.jcp.2003.09.032.

[43] M. Bujny. "Level set topology optimization for crashworthiness using evolutionary algorithms and machine learning". PhD thesis. Munich: Technical University of Munich, Germany, 2020. URL: http://mediatum.ub.tum.de/doc/1540709/document.pdf.

[44] K. Svanberg. "The method of moving asymptotes—a new method for structural optimization". In: *International Journal for Numerical Methods in Engineering* 24.2 (1987), pp. 359–373. DOI: 10.1002/nme.1620240207.

[45] R. J. Yang. "Multidiscipline topology optimization". In: *Computers & Structures* 63.6 (1997), pp. 1205–1212. DOI: 10.1016/s0045-7949(96)00402-6.

[46] C. B. W. Pedersen. "Topology optimization design of crushed 2D-frames for desired energy absorption history". In: *Structural and Multidisciplinary Optimization* 25.5-6 (2003), pp. 368–382. DOI: 10.1007/s00158-003-0282-y.

[47] N. M. Patel, B. S. Kang, J. E. Renaud, and A. Tovar. "Crashworthiness design using topology optimization". In: *Journal of Mechanical Design, Transactions of the ASME* 131.6 (2009). DOI: 10.1115/1.3116256.

[48] A. Tovar, N. M. Patel, A. K. Kaushik, and J. E. Renaud. "Optimality Conditions of the Hybrid Cellular Automata for Structural Optimization". In: *AIAA Journal* 45.3 (2007), pp. 673–683. DOI: 10.2514/1.20184.

[49] S. Hunkeler. “Topology Optimisation in Crashworthiness Design via Hybrid Cellular Automata for Thin Walled Structures”. PhD thesis. Queen Mary University of London, UK, 2013.

[50] D. Zeng. “Enhanced Hybrid Cellular Automata Method for Crashworthiness Topology Optimization of Thin-walled Structures”. PhD thesis. Technical University of Munich, Germany, 2019.

[51] Y. M. Xie and G. P. Steven. “A simple evolutionary procedure for structural optimization”. In: *Computers & Structures* 49.5 (1993), pp. 885–896. DOI: 10.1016/0045-7949(93)90035-c.

[52] K. Deb and H. Jain. “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints”. In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), pp. 577–601. DOI: 10.1109/tevc.2013.2281535.

[53] H. Jain and K. Deb. “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach”. In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), pp. 602–622. DOI: 10.1109/tevc.2013.2281534.

[54] I. Y. Kim and O. L. de Weck. “Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation”. In: *Structural and Multidisciplinary Optimization* 31.2 (2005), pp. 105–116. DOI: 10.1007/s00158-005-0557-6.

[55] Y. Sato, K. Izui, T. Yamada, and S. Nishiwaki. “Pareto frontier exploration in multiobjective topology optimization using adaptive weighting and point selection schemes”. In: *Structural and Multidisciplinary Optimization* 55.2 (2016), pp. 409–422. DOI: 10.1007/s00158-016-1499-x.

[56] N. Ryu and S. Min. “Multiobjective optimization with an adaptive weight determination scheme using the concept of hyperplane”. In: *International Journal for Numerical Methods in Engineering* 118.6 (2019), pp. 303–319. DOI: 10.1002/nme.6013.

[57] I. Das and J. E. Dennis. “Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems”. In: *SIAM Journal on Optimization* 8.3 (1998), pp. 631–657. DOI: 10.1137/s1052623496307510.

[58] LIVERMORE SOFTWARE TECHNOLOGY CORPORATION (LSTC). *LS-DYNA Software*. Version R10.0. Feb. 19, 2010. URL: lsdyna.ansys.com.

[59] Y. Wu. “Skeleton Analysis of Topology Optimization Results”. Master’s thesis. Technical University of Munich, Germany, 2021.

[60] E. Hutapea, N. Dommaraju, M. Bujny, and F. Duddeck. “Clustering Topologically Optimized Designs Based on Structural Deformation”. In: *Proc. of the Munich Symposium on Lightweight Design 2021*. Springer Berlin Heidelberg, 2022, pp. 104–114. DOI: 10.1007/978-3-662-65216-9_10.

[61] Y. Shimizu, N. Dommaraju, M. Bujny, S. Menzel, M. Olhofer, and F. Duddeck. "Deformation Clustering Methods for Topologically Optimized Structures under Crash Load based on Displacement Time Series". In: *World Congress on Computational Mechanics*. 2022. DOI: 10.23967/wccm-apcom.2022.037.

[62] J. C. Ferreira, C. M. Fonseca, and A. Gaspar-Cunha. "Methodology to select solutions from the pareto-optimal set: a comparative study". In: *Proc. of the 9th annual conference on genetic and evolutionary computation - GECCO '07*. ACM Press, 2007, pp. 789–796. DOI: 10.1145/1276958.1277117.

[63] H. K. Singh, T. Ray, T. Rodemann, and M. Olhofer. "Identifying solutions of interest for practical many-objective problems using recursive expected marginal utility". In: *Proc. of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2019, pp. 1734–1741. DOI: 10.1145/3319619.3326804.

[64] D. Cvetkovic and I. C. Parmee. "Preferences and their application in evolutionary multiobjective optimization". In: *IEEE Transactions on Evolutionary Computation* 6.1 (2002), pp. 42–57. DOI: 10.1109/4235.985691.

[65] Y. Sato, K. Izui, T. Yamada, and S. Nishiwaki. "Data mining based on clustering and association rule analysis for knowledge discovery in multiobjective topology optimization". In: *Expert Systems with Applications* 119 (2019), pp. 247–261. DOI: 10.1016/j.eswa.2018.10.047.

[66] R. Mcgill, J. W. Tukey, and W. A. Larsen. "Variations of Box Plots". In: *The American Statistician* 32.1 (1978), pp. 12–16. DOI: 10.1080/00031305.1978.10479236.

[67] J. L. Hintze and R. D. Nelson. "Violin Plots: A Box Plot-Density Trace Synergism". In: *The American Statistician* 52.2 (1998), pp. 181–184. DOI: 10.1080/00031305.1998.10480559.

[68] P. Kampstra. "Beanplot: A Boxplot Alternative for Visual Comparison of Distributions". In: *Journal of Statistical Software* 28.Code Snippet 1 (2008). DOI: 10.18637/jss.v028.c01.

[69] K. R. Gabriel. "The biplot graphic display of matrices with application to principal component analysis". In: *Biometrika* 58.3 (1971), pp. 453–467. DOI: 10.1093/biomet/58.3.453.

[70] M. Friendly. "Mosaic Displays for Multi-Way Contingency Tables". In: *Journal of the American Statistical Association* 89.425 (1994), pp. 190–200. DOI: 10.1080/01621459.1994.10476460.

[71] W. S. Cleveland. "Coplots, Nonparametric Regression, and Conditionally Parametric Fits". In: *Lecture Notes-Monograph Series* 24 (1994), pp. 21–36. URL: http://www.jstor.org/stable/4355791.

[72] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[73] G. J. McLachlan and K. E. Basford. *Mixture models: Inference and applications to clustering*. Vol. 38. New York: Dekker, 1988.

[74] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining*. 1996, pp. 226–231.

[75] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. "OPTICS: Ordering Points to Identify the Clustering Structure". In: *SIGMOD Record (ACM Special Interest Group on Management of Data)* 28.2 (1999), pp. 49–60. DOI: 10.1145/304181.304187.

[76] A. K. Jain. "Data clustering: 50 years beyond K-means". In: *Pattern Recognition Letters* 31.8 (2010), pp. 651–666. DOI: 10.1016/j.patrec.2009.09.011.

[77] M. Meilă. "The Uniqueness of a Good Optimum for K-Means". In: *Proc. of the 23rd Int. Conf. on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 625–632. ISBN: 1595933832. DOI: 10.1145/1143844.1143923.

[78] L. Kaufman and P. J. Rousseeuw. "Partitioning Around Medoids (Program PAM)". In: *Finding Groups in Data*. John Wiley & Sons, Inc., 1990, pp. 68–125. DOI: 10.1002/9780470316801.ch2.

[79] P. J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. DOI: 10.1016/0377-0427(87)90125-7.

[80] F. Lanfermann, S. Schmitt, and S. Menzel. "An Effective Measure to Identify Meaningful Concepts in Engineering Design optimization". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020. DOI: 10.1109/ssci47803.2020.9308484.

[81] H. Hotelling. "Analysis of a complex of statistical variables into principal components". In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441. DOI: 10.1037/h0071325.

[82] Z. Bozakov, L. Graening, S. Hasler, H. Wersing, and S. Menzel. "Unsupervised extraction of design components for a 3D parts-based representation". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 2009–2016. DOI: 10.1109/ijcnn.2008.4634074.

[83] E. Ulu, R. Zhang, and L. B. Kara. "A data-driven investigation and estimation of optimal topologies under variable loading configurations". In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 4.2 (2015), pp. 61–72. DOI: 10.1080/21681163.2015.1030775.

[84] D. Feldman, M. Schmidt, and C. Sohler. "Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering". In: *SIAM Journal on Computing* 49.3 (2020), pp. 601–657. DOI: 10.1137/18m1209854.

[85] L. van der Maaten and G. Hinton. “Visualizing data using t-SNE”. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2625.

[86] L. McInnes, J. Healy, N. Saul, and L. Großberger. “UMAP: Uniform Manifold Approximation and Projection”. In: *Journal of Open Source Software* 3.29 (2018), p. 861. DOI: 10.21105/joss.00861.

[87] T. Kohonen. “The self-organizing map”. In: *Proc. of the IEEE* 78.9 (1990), pp. 1464–1480. DOI: 10.1109/5.58325.

[88] A. Inselberg. “The plane with parallel coordinates”. In: *The Visual Computer* 1.2 (1985), pp. 69–91. DOI: 10.1007/bf01898350.

[89] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Chapman and Hall/CRC, 2018. DOI: 10.1201/9781351072304.

[90] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. “DNA visual and analytic data mining”. In: *Proc.. Visualization ’97 (Cat. No. 97CB36155)*. IEEE, 1997. DOI: 10.1109/visual.1997.663916.

[91] J. LeBlanc, M. O. Ward, and N. Wittels. “Exploring N-dimensional databases”. In: *Proc. of the First IEEE Conference on Visualization: Visualization ‘90*. IEEE Comput. Soc. Press, 1990. DOI: 10.1109/visual.1990.146386.

[92] M. O. Ward. “Multivariate Data Glyphs: Principles and Practice”. In: *Handbook of Data Visualization*. Springer Berlin Heidelberg, 2008, pp. 179–198. DOI: 10.1007/978-3-540-33037-0_8.

[93] N. Dommaraju, M. Bujny, S. Menzel, M. Olhofer, and F. Duddeck. “Identifying Topological Prototypes using Deep Point Cloud Autoencoder Networks”. In: *2019 Int. Conf. on Data Mining Workshops (ICDMW)*. IEEE, 2019, pp. 761–768. DOI: 10.1109/icdmw.2019.00113.

[94] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 77–85. DOI: 10.1109/cvpr.2017.16.

[95] Y. Yang, C. Feng, Y. Shen, and D. Tian. “FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 206–215. DOI: 10.1109/cvpr.2018.00029.

[96] T. Rios, B. Van Stein, S. Menzel, T. Back, B. Sendhoff, and P. Wollstadt. “Feature Visualization for 3D Point Cloud Autoencoders”. In: *Proc. of the International Joint Conference on Neural Networks*. 2020, pp. 1–9. ISBN: 9781728169262. DOI: 10.1109/IJCNN48605.2020.9207326.

[97] S. Saha, T. Rios, L. L. Minku, et al. “Exploiting Generative Models for Performance Predictions of 3D Car Designs”. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021. DOI: 10.1109/ssci50451.2021.9660034.

[98] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. "Efficient Implementation of Marching Cubes' Cases with Topological Guarantees". In: *Journal of Graphics Tools* 8.2 (2003), pp. 1–15. DOI: 10.1080/10867651.2003.10487582.

[99] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, et al. "Scikit-image: image processing in Python". In: *PeerJ* 2 (2014), e453. DOI: 10.7717/peerj.453.

[100] E. W. Weisstein. *Triangle point picking*. 1999. URL: https://mathworld.wolfram.com/TrianglePointPicking.html (visited on 09/01/2022).

[101] M. Dawson-Haggerty. *trimesh (3.2.0)*. 2019. URL: http://trimsh.org (visited on 09/01/2022).

[102] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *32nd Int. Conf. on Machine Learning, ICML 2015* 1 (2015), pp. 448–456.

[103] K. Marhadi and S. Venkataraman. "Comparison of Load Path Definitions in 2-D Continuum Structures". In: *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2009. DOI: 10.2514/6.2009-2354.

[104] Y. Zhou and A. W. Toga. "Efficient skeletonization of volumetric objects". In: *IEEE Transactions on Visualization and Computer Graphics* 5.3 (1999), pp. 196–209. DOI: 10.1109/2945.795212.

[105] P. K. Saha, G. Borgefors, and G. S. di Baja. "A survey on skeletonization algorithms and their applications". In: *Pattern Recognition Letters* 76 (2016), pp. 3–12. DOI: 10.1016/j.patrec.2015.04.006.

[106] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. "3D Skeletons: A State-of-the-Art Report". In: *Computer Graphics Forum* 35.2 (2016), pp. 573–597. DOI: 10.1111/cgf.12865.

[107] T. C. Lee, R. L. Kashyap, and C. N. Chu. "Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms". In: *CVGIP: Graphical Models and Image Processing* 56.6 (1994), pp. 462–478. DOI: 10.1006/cgip.1994.1042.

[108] S. Skylar, I.-T. Rodrigo, G. Jochen, A. Nikola, and W. Patricia. "A Compact Spectral Descriptor for Shape Deformations". In: *European Conference on Artificial Intelligence*. Vol. 325. IOS Press, 2020, pp. 1930–1937. DOI: 10.3233/FAIA200311.

[109] J. Garcke and R. Iza-Teran. "Machine Learning Approaches for Data from Car Crashes and Numerical Car Crash Simulations". In: *Int. Conf. Simulation Process & Data Management (SPDM)*. 2017.

[110] R. R. Coifman and S. Lafon. "Diffusion maps". In: *Applied and Computational Harmonic Analysis* 21.1 (2006), pp. 5–30. DOI: 10.1016/j.acha.2006.04.006.

[111] M. Müller. *Dynamic Time Warping*. Springer Berlin Heidelberg, 2007, pp. 69–84. DOI: 10.1007/978-3-540-74048-3_4.

[112] S. Oh, Y. Jung, S. Kim, I. Lee, and N. Kang. "Deep Generative Design: Integration of Topology Optimization and Generative Models". In: *Journal of Mechanical Design* 141.11 (2019). DOI: 10.1115/1.4044229.

[113] N. Dommaraju, M. Bujny, S. Menzel, M. Olhofer, and F. Duddeck. "Evaluation of geometric similarity metrics for structural clusters generated using topology optimization". In: *Applied Intelligence* (2022). DOI: 10.1007/s10489-022-03301-0.

[114] G. L. López, A. P. P. Negrón, A. D. A. Jiménez, J. R. Rodríguez, and R. I. Paredes. "Comparative analysis of shape descriptors for 3D objects". In: *Multimedia Tools and Applications* 76.5 (2016), pp. 6993–7040. DOI: 10.1007/s11042-016-3330-5.

[115] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris. "Deep Learning Advances in Computer Vision with 3D Data". In: *ACM Computing Surveys* 50.2 (2017), pp. 1–38. DOI: 10.1145/3042064.

[116] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. "O-CNN: Octree-Based Convolutional Neural Networks for 3D Shape Analysis". In: *ACM Transactions on Graphics* 36.4 (2017), pp. 1–11. DOI: 10.1145/3072959.3073608.

[117] H. Zhang, O. V. Kaick, and R. Dyer. "Spectral Mesh Processing". In: *Computer Graphics Forum* 29.6 (2010), pp. 1865–1894. DOI: 10.1111/j.1467-8659.2010.01655.x.

[118] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements Continua and Structures*. John Wiley & Sons, Dec. 27, 2013. 834 pp. ISBN: 1118632702.

[119] M. Maimaitimin, K. Watanabe, and S. Maeyama. "Stacked convolutional auto-encoders for surface recognition based on 3d point cloud data". In: *Artificial Life and Robotics* 22.2 (2017), pp. 259–264. DOI: 10.1007/s10015-017-0350-9.

[120] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. "Variational Autoencoders for Deforming 3D Mesh Models". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018. DOI: 10.1109/cvpr.2018.00612.

[121] G. Peyré and M. Cuturi. "Computational Optimal Transport: With Applications to Data Science". In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607. DOI: 10.1561/2200000073.

[122] I. T. Jolliffe. *Principal component analysis*. Springer-Verlag, 2002. ISBN: 9780387224404. DOI: 10.1007/b98835.

[123] D. D. Lee and H. S. Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755 (1999), pp. 788–791. DOI: 10.1038/44565.

[124] P. Schober, C. Boer, and L. A. Schwarte. "Correlation coefficients: Appropriate use and interpretation". In: *Anesthesia & Analgesia* 126.5 (2018), pp. 1763–1768. DOI: 10.1213/ane.0000000000002864.

[125] S. Godbole and S. Sarawagi. "Discriminative Methods for Multi-labeled Classification". In: *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2004, pp. 22–30. DOI: 10.1007/978-3-540-24775-3_5.

[126] N. X. Vinh, J. Epps, and J. Bailey. “Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance”. In: *J. Mach. Learn. Res.* 11 (2010), pp. 2837–2854.

[127] A. X. Chang, T. Funkhouser, L. Guibas, et al. *ShapeNet: An Information-Rich 3D Model Repository.* Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[128] L. Breiman. “Random forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/a:1010933404324.

[129] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proc. of the Thirteenth Int. Conf. on Artificial Intelligence and Statistics*. Ed. by Y. W. Teh and M. Titterington. Vol. 9. Proc. of Machine Learning Research. PMLR, 2010, pp. 249–256. URL: https://Proc..mlr.press/v9/glorot10a.html.

[130] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. “Scikit-learn: Machine learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[131] M. P. Hansen and A. Jaszkiewicz. *Evaluating the quality of approximations to the non-dominated set.* IMM, Department of Mathematical Modelling, Technical University of Denmark, 1994.

[132] J. Knowles and D. Corne. “On metrics for comparing nondominated sets”. In: *Proc. of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600).* IEEE, 2002. DOI: 10.1109/cec.2002.1007013.

[133] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. “Performance assessment of multiobjective optimizers: an analysis and review”. In: *IEEE Transactions on Evolutionary Computation* 7.2 (2003), pp. 117–132. DOI: 10.1109/tevc.2003.810758.

[134] H. Ishibuchi and Y. Shibata. “Mating Scheme for Controlling the Diversity-Convergence Balance for Multiobjective Optimization”. In: *Genetic and Evolutionary Computation – GECCO 2004.* Springer Berlin Heidelberg, 2004, pp. 1259–1271. DOI: 10.1007/978-3-540-24854-5_121.

[135] K. C. Tan, T. H. Lee, and E. F. Khor. “Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons”. In: *Artificial Intelligence Review* 17.4 (2002), pp. 251–290. DOI: 10.1023/a:1015516501242.

[136] N. Srinivas and K. Deb. “Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”. In: *Evolutionary Computation* 2.3 (1994), pp. 221–248. DOI: 10.1162/evco.1994.2.3.221.

[137] T. Okabe, Y. Jin, and B. Sendhoff. “A critical survey of performance indices for multi-objective optimisation”. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.* Vol. 2. IEEE, 2003, pp. 878–885. DOI: 10.1109/cec.2003.1299759.

[138] A. Farhang-Mehr and S. Azarm. “Diversity assessment of Pareto optimal solution sets: an entropy approach”. In: *Proc. of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No.02TH8600)*. Vol. 1. IEEE, 2002, pp. 723–728. DOI: 10.1109/cec.2002.1007015.

[139] W. Zhang, J. Yuan, J. Zhang, and X. Guo. “A new topology optimization approach based on Moving Morphable Components (MMC) and the ersatz material model”. In: *Structural and Multidisciplinary Optimization* 53.6 (2016), pp. 1243–1260. DOI: 10.1007/s00158-015-1372-3.

[140] X. Lei, C. Liu, Z. Du, W. Zhang, and X. Guo. “Machine Learning-Driven Real-Time Topology Optimization Under Moving Morphable Component-Based Framework”. In: *Journal of Applied Mechanics* 86.1 (2018). DOI: 10.1115/1.4041319.

[141] J. Bai and W. Zuo. “Hollow structural design in topology optimization via moving morphable component method”. In: *Structural and Multidisciplinary Optimization* 61.1 (2020), pp. 187–205. DOI: 10.1007/s00158-019-02353-0.

[142] F. Wein, P. D. Dunning, and J. A. Norato. “A review on feature-mapping methods for structural optimization”. In: *Structural and Multidisciplinary Optimization* 62.4 (2020), pp. 1597–1638. DOI: 10.1007/s00158-020-02649-6.

[143] N. Dommaraju, M. Bujny, S. Menzel, M. Olhofer, and F. Duddeck. “Cooperative Multi-objective Topology Optimization Using Clustering and Metamodeling”. In: *2022 IEEE congress on evolutionary computation (CEC)*. IEEE. 2022.

[144] T. B. To and B. Korn. “MOBES: A Multiobjective Evolution Strategy for Constrained Optimization Problems”. In: *The Third Int. Conf. on Genetic Algorithms (Mendel 97)*. Vol. 25. 1997, p. 27.

[145] K. V. Price. “Differential Evolution”. In: *Handbook of Optimization*. Springer Berlin Heidelberg, 2013, pp. 187–214. DOI: 10.1007/978-3-642-30504-7_8.

[146] P. Virtanen, R. Gommers, T. E. Oliphant, et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[147] P. Wollstadt, M. Bujny, S. Ramnath, J. J. Shah, D. Detwiler, and S. Menzel. “Car-Hoods10k: An Industry-grade Data Set for Representation Learning and Design Optimization in Engineering Applications”. In: *IEEE Transactions on Evolutionary Computation* (2022), pp. 1–1. DOI: 10.1109/tevc.2022.3147013.

[148] J. Lin, Z. Luo, and L. Tong. “A new multi-objective programming scheme for topology optimization of compliant mechanisms”. In: *Structural and Multidisciplinary Optimization* 40.1-6 (2009), pp. 241–255. DOI: 10.1007/s00158-008-0355-z.

[149] S. Doi, H. Sasaki, and H. Igarashi. “Multi-Objective Topology Optimization of Rotating Machines Using Deep Learning”. In: *IEEE Transactions on Magnetics* 55.6 (2019), pp. 1–5. DOI: 10.1109/tmag.2019.2899934.

[150] P. A. Pour, T. Rodemann, J. Hakanen, and K. Miettinen. “Surrogate assisted interactive multiobjective optimization in energy system design of buildings”. In: *Optimization and Engineering* (2021). DOI: 10.1007/s11081-020-09587-8.

[151] D. Arthur and S. Vassilvitskii. “K-means++: The advantages of careful seeding”. In: *Proc. of the Annual ACM-SIAM Symposium on Discrete Algorithms*. Ed. by H. Gabow. 2007, pp. 1027–1035. ISBN: 9780898716245.

[152] N. Dommaraju, M. Bujny, S. Menzel, M. Olhofer, and F. Duddeck. “Simultaneous Exploration of Geometric Features and Performance in Design Optimization”. In: *16th International LS-DYNA Conference*. 2020, p. 12.

[153] K. Deb and J. Sundar. “Reference point based multi-objective optimization using evolutionary algorithms”. In: *Proc. of the 8th annual conference on Genetic and evolutionary computation - GECCO ’06*. ACM Press, 2006. DOI: 10.1145/1143997.1144112.

[154] M. Luque, K. Miettinen, P. Eskelinen, and F. Ruiz. “Incorporating preference information in interactive reference point methods for multiobjective optimization”. In: *Omega* 37.2 (2009), pp. 450–462. DOI: 10.1016/j.omega.2007.06.001.

[155] N. Aulig and M. Olhofer. “Evolutionary computation for topology optimization of mechanical structures: An overview of representations”. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 1948–1955. ISBN: 9781509006229. DOI: 10.1109/CEC.2016.7744026.