Universiteit Leiden

Master Computer Science

Investigating the Interplay of Parameterization and Optimizer in Gradient-Free Topology Optimization: A Cantilever Case Study

Name:	Jelle Westra
Student ID:	s3607828
Date:	11/08/2025
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. Elena Raponi
2nd supervisor:	Dr. Niki van Stein

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Gradient-free black-box optimization (BBO) methods offer a flexible framework for topology optimization (TO), enabling the discovery of novel structural designs without requiring gradient information. However, their success critically dependents on geometric parameterization of the design space, and the selection of a suitable optimizer. This thesis investigates this optimizer–parameterization interplay in a structural TO case study: minimizing compliance in a horizontal cantilever beam subject to a connectivity constraint. The constraint, based on minimum connecting distance, effectively guides the search toward feasible, connected designs across all tested settings.

We evaluate the performance of three distinct geometric parameterizations: Movable Morphable Components (MMCs); Curved MMCs; and Honeycomb Tiling; combined with three distinct BBO algorithms: Differential Evolution (DE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and Heteroscedastic Evolutionary Bayesian Optimization (HEBO). These optimizers are benchmarked on the parameterizations defining design spaces of three different dimensionalities: 10D, 20D, and 50D.

For this case study, our central finding is that parameterization quality has a more significant impact on optimization outcomes than the choice of optimizer. A well-designed parameterization enables robust and competitive performance across all tested optimizers, whereas a poor parameterization creates a strong dependency on optimizer selection.

We further examine the applicability of Exploratory Landscape Analysis (ELA) for characterizing these real-world TO problems. Standard ELA feature sets, developed for unconstrained benchmarks, prove of limited use due to the sparsity of the feasible design space, limiting their value for both algorithm selection and for characterizing the different TO parameterizations. We have not found a way to gain insight into our parameterizations from ELA. This highlights the need for constraint-aware or domain-specific landscape analysis methods for practical TO applications.

Overall, this work provides new insights into the role of parameterization in real-world BBO problems and its implications for optimizer selection using a TO case-study.

Acknowledgements

First, I want to thank my first supervisor Elena Raponi for her excellent guidance throughout the project. I have to thank her for the opportunity, and learning quite a lot from her and this project. Second, Ivan Olarte Rodriguez, who helped me a lot, especially during the hectic last week of writing. I had many insightful discussions with both Elena and Ivan, which greatly contributed to my progress.

Finally, thanks to my family, friends, and girlfriend, I honestly couldn't do this without you.

Contents

1 Introduction

1.1 The New Age of Engineering

Traditionally the search for optimal design relied on expert knowledge and the centuries of engineering experience, however by utilizing black-box optimization (BBO) techniques such as Evolutionary Algorithms (EAs) [1] and Bayesian Optimization (BO) [2], it becomes possible to find novel and highly optimized designs without prior domain expertise. Using these algorithms, a design automatically and incrementally is improved according to the response of the system and an objective set by the engineer. In that sense: the engineer's task becomes to describe the problem as suitable and properly as possible for the optimization algorithm, opposed to coming up with designs themselves to be evaluated. The main rationale for this is that the objective driven designs are highly specialized for a specific task and are beyond what a trained engineer with domain knowledge could design in the same given time frame [3].

The increase in computational power has enabled the use of more advanced numerical methods capable of addressing increasingly complex physical models, thereby reducing the reliance on physical prototyping. Widely available Finite Element Method (FEM) tools discretize geometry into a mesh, enabling the simulation of a broad range of physical phenomena [4]. This (partially) removes the requirement for physical prototyping and use of scale models, which is costly material- and time-wise. There is however somewhat of an emphasis on partially here: still a model is a model. For example, it is challenging to model transitioning between physics regimes such as a rocket going transonic [5]. Yet it is not so challenging to imagine such a simulation is often much simpler and cheaper than sending an actual rocket to space. This resulted engineering in the last decades shifted towards these simulation-based approaches, accelerating the initial design phase.

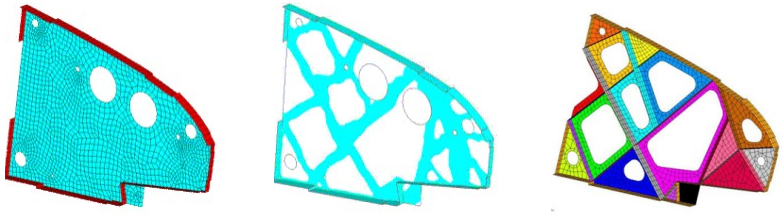
Secondly, by the innovation in fabrication techniques such as additive manufacturing and micro fabrication, highly complex designs such as interweaving meshes or organic structures can now be 3D printed in different types of materials with relative ease [6]. For example, highly customized low quantity parts are already fabricated with metal additive manufacturing in biomedical and aerospace industries [7, 8]. In previous decades these were infeasible with traditional casting or subtractive machining techniques. These opened the door for rapid prototyping as designs can be realized quickly on a desktop without the requirement for expensive molds or a trained machine shop workerforce.

1.2 Introducing: Topology Optimization

One design challenge one might face is to find an optimal material distribution in some bounded domain for weight- or material saving purposes; solving this problem is what is known as Topology Optimization (TO) [9]. This bounded domain may be a some sheet or block of material that functions for example as the connecting frame of some design. Design objectives might involve saving

as much material as possible under the constraint of some maximal bending threshold of the structure, or to maximize the stiffness given a predefined material budget. Although this framework is general to any problem, TO typically is applied to the minimization of compliance for static loading cases on structures, dubbed structural TO. Other problems one could think of are: improving fluid flows, vibro-acoustics, and heat transfer. The ultimate goal is to find a highly specialized, optimized even, material distribution for a specific scenario.

Figure 1 shows an example of structural TO: (A→B) the sheet is treated as canvas for which the amount of material is minimized. This is under satisfaction of structural feasibility cases set by the engineers. To make the TO result more suitable for fabrication, a Shape Optimization procedure can be performed (B→C). However, in this thesis we focus solely on the TO phase. To give a



(A) Original Sheet (B) Topology Optimization (C) Shape Optimization

Figure 1: Minimizing the weight of a sheet of metal (A) of a airplane wing rib by means of Topology Optimization (B). And, making the Optimized Topology production ready by means of Shape Optimization (C).

motivating industry application, corresponding to Figure 1: during a redesign campaign, structural TO was applied on various parts for weight saving purposes on the Airbus A380 [10], saving around 1000 kg of weight [11]. Clearly, such a redesign has quite a beneficial financial impact on the airliners operating the Airbus A380. Swapping five additional passengers for the saved 1000 kg, assuming one flight per day, and a ticket cost of €1000, results in 1.8M €/year additional revenue thanks to TO[1].

1.3 A Brief History of Topology Optimization

The homogenization-based approach was introduced and pioneered by Bendsøe and Kikuchi in 1988 [12], initiating the field of TO. The design domain is divided into a mesh of resizable micro structures (holes), the objective is to determine the optimal size for each individual hole such that the overall design performs optimally under static loading. Homogenization refers to the technique of approximating these microstructures with a less dense material for a given cell based on the size of its hole [13]. Note that, despite the approximation of the

[1] Take this calculation with a grain of salt, it is merely a guesstimate.

microstructures by filling the cell with less dense material, the underlying design still has a binary nature: it is either solid material or void.

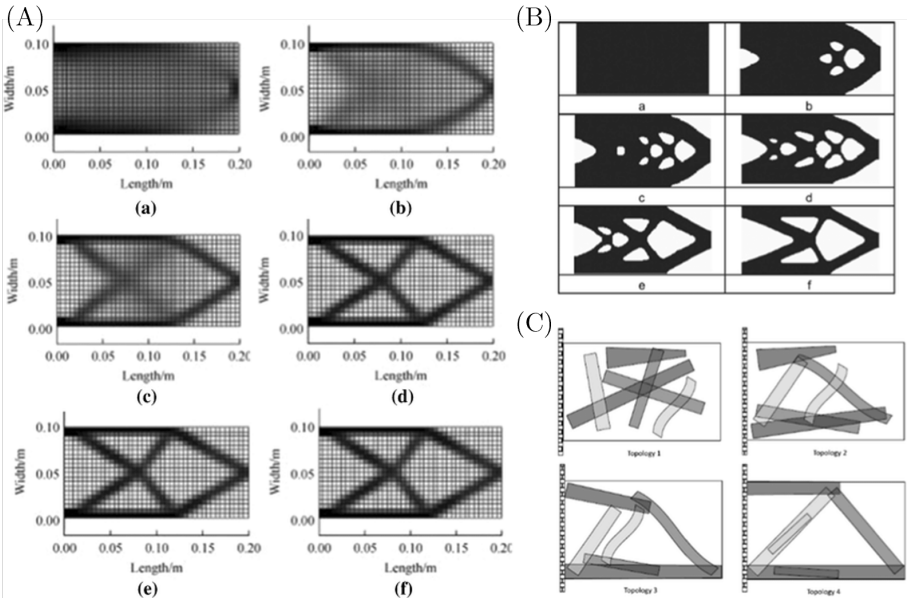


Figure 2: An overview of different methods TO applied on structural cantilever problems: (A) iterations of the SIMP method; (B) iterations of the ESO method; and (C) iterations of a MMC parameterized optimization. (Adapted from: [14])

A year later (1989) the Solid Isotropic Material with Penalization (SIMP) method was introduced [15]; SIMP directly optimizes the relative material density by attributing cells a solid isotropic material opposed to the previously used micro-structures (Figure 2A). This density is continuous in the range (0,1) opposed to a binary distribution, which is one that we can actually fabricate. The penalization in SIMP refers to way the stiffness is penalized with respect to the relative density: the stiffness is related to the density through a power law[2]. This promotes designs to not use intermediate densities as contributions of these cells are artificially weakened.

To this day SIMP is one of the most widely used methods in structural topology optimization due to its simplicity, effectiveness, and compatibility with gradient-based solvers, namely the Moving Morphable Asymptotes (MMA) method [16]. It is often used as a foundation or inspiration for extensions to other domains like multiphysics, heat transfer, and dynamics.

The Evolutionary Structural Optimization (ESO) method, introduced by Xie and Steven (1993), is a computational approach inspired by natural evolution

[2]For example using the conventionally used $p = 3$, a relative density of 50% gives the cell a relative stiffness of $(50\%)^3 = 12.5\%$.

to enhance structural performance [17]. In its basic form, ESO heuristically removes material with low stress from the initial design (Figure 2B). However, a key limitation of this one-way removal strategy is that once material is deleted, it cannot be restored, potentially leading to suboptimal designs. To address this, in a follow-up paper the Bi-directional ESO (BESO) method was developed (1998), allowing both the elimination of inefficient material and the reintroduction of beneficial elements during the optimization process [18]. Though this method is limited to structural TO problems of minimizing compliance.

The Level Set Method (LSM), proposed by Wang et al. (2003) and Allaire et al.(2004), instead of discretizing the domain in cells proposes to implicitly describe the void-material interface using level set functions [19, 20]. Therefore the level-set approach naturally maintains a crisp, well-defined boundary and handles topological changes such as merging or splitting of material regions. Direct optimization of this interface omits problems found in density-based approaches as: checkerboard patterns and regions with intermediate densities, so-called gray areas. However, this does mean that like in SIMP a one value per cell is optimized leading to high dimensional problems quickly for fine meshes.

Moreover, the original formulation of the LSM requires solving an advection (Hamilton–Jacobi) equation, which introduces additional numerical complexity and potential instability. Additionally, the resulting topology is often sensitive to the initial design, as the number and distribution of holes must typically be predefined, effectively warm-starting the optimization process and potentially biasing the final outcome.

Another variant that can attributed to the LSM category are Morphable Movable Components (MMCs), as proposed by Guo (2014) [21]. Instead of direct evolution of the boundary, the geometry is build out of multiple building blocks: beam-like MMCs (Figure 2C). The geometry of each MMC is controlled with a few shape- and size defining parameters. Also, as the name suggests, the MMC is free to move in the design domain. Relatively complex designs can be obtained with far less parameters as a single shape may occupy many elements of the mesh. This is results in far less design parameters have to optimized to the one parameter per element strategies found in SIMP and LSM. This also benefits fabrication, as the optimized design is already composed of beam-like components.

1.4 Challenges in Topology Optimization

The study of compliance minimization in TO resulted in highly specialized procedures that rely on the gradients of the objective with respect to the parameterization of the design and problem domain, dubbed sensitivity analysis [22]. To counter checkerboard patterns, considered disconnected structures, and gray areas in SIMP, which blur the material/void distinction, specialized filtering techniques are employed. An example of these gray areas can be seen in the first couple iterations of Figure 2(A).

In multiphysics domains like vibroacoustics and viscothermal problems, tailored physics interpolation schemes or specialized (problem-specific) optimiza-

tion methods are required [23]. Developing these demands significant mathematical expertise and is typically feasible only for experienced simulation-optimization domain specialists.

Yet, in BBO methods such a sensitivity analysis is not required. These methods treat the underlying simulation as an oracle, querying it iteratively to explore and exploit designs. This makes them particularly attractive for problems where gradients are unavailable, unreliable, or prohibitively expensive to compute. For instance, in crashworthiness optimization, the objective function may stem from complex nonlinear simulations with discontinuities and contact events, making gradient-based methods impractical [24].

The use of non-gradient methods—particularly EAs—in TO has faced criticism, mainly due to early applications using dense grid-based representations [25, 26], which led to high-dimensional, inefficient searches. Additionally, their stochastic nature means results can vary significantly between runs, with near-optimal solutions found in fewer than 10% of cases for simple benchmarks. To justify their use, Sigmund [27] proposed two criteria, at least one of which should be satisfied.

- **Discretization**: Must handle models with $\geq$1000 elements, match or outperform gradient-based methods with numerical sensitivities, and avoid artifacts like checkerboards.
- **Problem type**: Should address problems intractable by standard gradient-based methods.

Since the strength of BBO methods lie in generality: they can handle complex, multi-modal, discontinuous, and noisy problems. We support the use of EA and BO for TO based on the second criterion, as they provide a general optimization framework applicable to a wide range of problems, especially where gradient methods are infeasible.

1.5 Exploratory Landscape Analysis

To characterize differences among BBO problems, Mersmann et al. in 2011 proposed a Exploratory Landscape Analysis (ELA) prior to optimization runs [28]. The goal of the ELA phase is to capture the problem description in a feature set. The rationale for this is that these features are correlated with optimizer performance. For example, BO might perform well on problems with certain characteristics whereas EAs perform better on others. This problem is known as Algorithm Selection (AS), particularly prevalent in Automated Machine Learning (AutoML) [29].

The No Free Lunch (NFL) theorem for optimization [30], further supports the motivation for problem characterization through ELA. It states that, averaged over all possible problems, all optimization algorithms perform equally. In other words, if an algorithm performs exceptionally well on a subset of problems, it must perform worse on others. This implies that there is no universally superior optimizer, and algorithm performance is inherently problem-dependent.

Consequently, the NFL principle reinforces the need for data-driven algorithm selection strategies, where knowledge about the structure of a specific problem (captured via ELA features) can inform the choice of an optimizer, rather than relying on fixed or default strategies.

1.6 Challenges and Objectives

Gradient-free approaches offer a flexible framework for solving BBO problems. However besides the underlying problem, their effectiveness depends heavily on the representation of the design space. Specifically for TO, that is the way geometry is parameterized. Poor parameterizations can mislead the search, cause premature convergence, or result in infeasible or low-quality designs. Thus, a key research objective is to investigate how different parameterizations interact with BBO methods and affect performance in TO settings.

Furthermore, ELA typically is studied with benchmark problems. In the initial proposal, the BBO Benchmarking (BBOB) problems [31], are successfully segmented into expert-designed classes based on their ELA features [28]. The question if this methodology transfers to a real-world problem and how well the feature-optimizer performance correlates to these benchmark problems.

This thesis is mainly an exploratory study, BBO methods are not adapted to suit problems, we purely focus on the interplay between out-of-the-box optimizers and parameterizations. The main objectives are:

- To investigate how different geometric parameterizations influence the performance and effectiveness of BBO methods in a structural TO case-study.
- To investigate whether ELA can be used to meaningfully characterize real-world TO problems and inform algorithm selection.
- To investigate the relationship between ELA features and optimizer performance, both within the TO setting and in comparison to established benchmark problems (e.g., BBOB).

2 Background

2.1 Topology Optimization

As introduced the goal of TO is to optimize the material distribution in a bounded design domain D according to an objective function f. The objective in our case is to minimize the structure's compliance. Intuitively minimizing the compliance is equivalent to maximizing the stiffness of the structure.

Since we only consider 2D problems: $D \subset \mathbb{R}^2$. In essence we optimize a binary distribution of the design domain described by some material indicator function $\alpha : D \to \{0, 1\}$, where 0 represents void and 1 represents material. That is the material subset:

$$\Omega = \{\mathbf{u} \in D : \alpha(\mathbf{u}) = 1\} \subseteq D \tag{1}$$

and void $D \backslash \Omega$. This means the parameterization must implicitly or explicitly imply the material indicator.

For example in LSM [32], this material indicator is implied by level-set functions; material regions are implicitly described by level-set functions: $\phi_j : D \to \mathbb{R}$. Moreover, the boundary of the j-th geometry Γ_j is described by the 0th iso-contour of ϕ_j. That is, the j-th geometry occupies the region in D for which

$$\Omega_j = \{\mathbf{u} \in D : \phi_j(\mathbf{u}) \geq 0\}. \tag{2}$$

And, the boundary of the level-set imposed geometry:

$$\Gamma_j = \partial\Omega_j = \{\mathbf{u} \in D : \phi_j(\mathbf{u}) = 0\}. \tag{3}$$

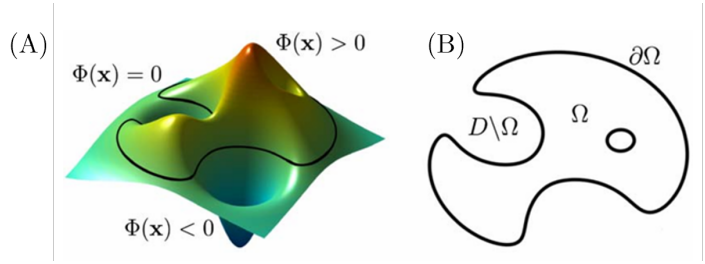


Figure 3: Level-set function parameterization: (A) a level-set ϕ describing material domain Ω; and (B) the distinction between corresponding material domain Ω, void $D \backslash \Omega$, and boundary $\partial\Omega$ [24].

In the case of SIMP [15], the design domain D is partitioned into fixed cells, specifically a regular grid. By "turning on" the j-th cell, the region corresponding to Ω_j is added to the total geometry.

We define the total geometry set as,

$$\Omega_{\text{geo}} = \{\Omega_j\} \tag{4}$$

with cardinality $|\Omega_{\text{geo}}| = n$, where n is the number of level-set functions, or the number of individual shapes like cells if you will.

We let $\mathbf{x} \in \mathbb{R}^d$ be the design vector describing the geometry through some parameterization $\mathcal{P} : \mathbb{R}^d \to \{\Omega_j\}$. From here the full material domain $\Omega \subseteq D$ is parameterized to:

$$\Omega(\mathbf{x}) = \bigcup_{\Omega_j \in \mathcal{P}(\mathbf{x})} \Omega_j. \tag{5}$$

and material indicator is parameterized as:

$$\alpha(\mathbf{u}; \mathbf{x}) = [\mathbf{u} \in \Omega(\mathbf{x})]. \tag{6}$$

Using the material indicator, implied from Ω, the volume of the material

$$\mathcal{V}(\mathbf{x}) = \int_{\Omega(\mathbf{x})} d\mathbf{u} = \int_D \alpha(\mathbf{u}; \mathbf{x}) d\mathbf{u}. \tag{7}$$

Similarly any objective is a functional of the parameterized material indicator:

$$f(\mathbf{x}) = \int_D c(\alpha(\mathbf{u}; \mathbf{x})) d\mathbf{u}. \tag{8}$$

In our case, compliance is used as the objective. This involves first solving for the displacement field corresponding to the material distribution, and then using this displacement to compute the compliance. However, since the compliance calculation depends on the underlying physics, we abstract it using an arbitrary function c.

To ensure numerical stability in the FEM calculations, a strictly binary material distribution (0/1) is avoided. Instead, void elements (0) are assigned a small stiffness: 10^{-9} of the original material stiffness in our case. This approach is commonly used in topology optimization and is known as the *ersatz material* method [9].

Traditionally the compliance is minimized under restriction of a maximum material budget $V_{\max}$ which leads to the following optimization problem:

$$\begin{aligned} \text{minimize:} \quad & f(\mathbf{x}) \text{ for } \mathbf{x} \in \mathcal{X} \\ \text{subject to:} \quad & \mathcal{V}(\mathbf{x}) - V_{\max} \leq 0 \end{aligned} \tag{9}$$

In here f is the objective, $\mathcal{X}$ is the space representing designs that can be parameterized, and $\mathcal{V}$ is the volume correponding to design $\mathbf{x} \in \mathcal{X}$.

As introduced, the question now is how parameterization, i.e. mathematical formulation of describing the geometry, affects the optimization of the TO problem. We now consider a few parameterizations.

2.2 Parameterizations

In this thesis we focus on the effect of three different ways of parameterization of the geometry in the design domain:

- Honeycomb Tiling;
- Movable Morpable Components (MMC); and
- Curved MMCs.

the Honeycomb tiling is a binary parameterization with cells that can be activated, MMCs represent straight beams, and the Curved MMCs add deformation parameters to give MMCs more flexibility to form curved beams.

2.2.1 Honeycomb Tiling

Like in SIMP [15], the design domain can be partitioned into cells. Using the SIMP grid, this leads to hundreds to thousands of elements which all are attributed their own design variable x_j. The dimension of this design space is equal to the number of elements in the grid. To keep the discrete material/void distinction the cell can be activated once its design variable exceeds a certain threshold:

$$x_j > \tau \implies \Omega_j \in \Omega_{\text{geo}}. \tag{10}$$

In here τ is a global threshold, and Ω_j represents the j-th cell in the grid. The number of cells in the grid dictate the dimension of such a parameterization.

Contrary to the gradient-based procedure in SIMP, BBO methods struggle on parameterizations with hundreds of variables. To facilitate this gradient-based approach, in SIMP the material/void is relaxed to be a continuous density. Previously it was attempted to directly optimize the discrete grid parameterization using an EA method but resulted in tens of thousands of iterations [26]. This was achieved by progressively subdividing the elements in the grid, starting from a coarse mesh to one that represent the full mesh (Figure 4A).

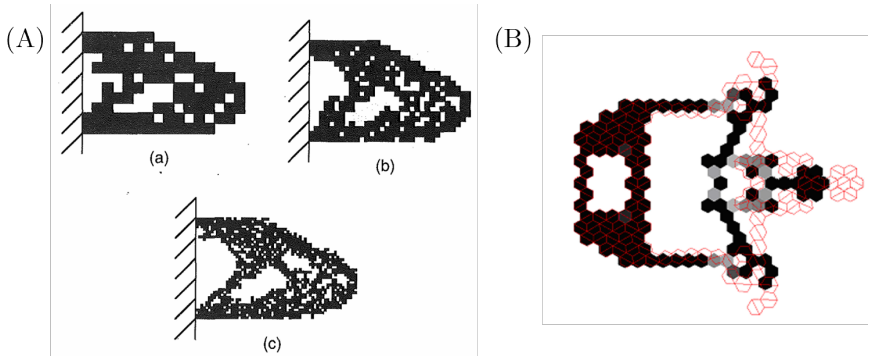


Figure 4: Grid-based parameterizations: (A) a rectangular grid solved by EA, where cells progressively getting sub-dived [26]; and (B) Honeycomb parameterization solved using gradient-based methods [33].

Opposed to rectangular grids, a Honeycomb parameterization has some benefits [33]. The hexagonal tiles in the Honeycomb pattern ensures continuous

edge connectivity and eliminates checkerboard patterns and stiffness singularities, providing a well-defined material layout for efficient topology synthesis. The former problems are prevalent in rectangular grids when pixels connect diagonally, only connecting the corners (Figure 4B). Such an individual cell could be used to activate multiple elements of a rectangular grid while keeping the dimension low, forming a potential entry point for EA-based optimization.

2.2.2 Movable Morphable Component Beams

In the case for the MMCs [21], the design vector $\mathbf{x}$ controls the shapes of the beam. Such a beam is parameterized using a quintuple of parameters: a beam of length l and thickness t is placed at position $(x_0, y_0) \in D$, and has an orientation of θ (Figure 5B). The level-set function is defined as:

$$\phi_j(u,v) = -\left[\left(\frac{\cos\theta_j(u-x_{0,j}) + \sin\theta_j(v-y_{0,j})}{l_j/2}\right)^m + \left(\frac{-\sin\theta_j(u-x_{0,j}) + \cos\theta_j(v-y_{0,j})}{t_j/2}\right)^m - 1\right] \tag{11}$$

In here j represents the j-th geometry, u and v correspond to x- and y coordinates inside the design domain D, and m is a shape controlling parameter set globally at $m = 6$. This shape is also known as hyperellipse ($m > 2$), as $m \to \infty$ the shape approaches a rectangle. For lower powers provides a smooth continuous shape (Figure 5A), which was initially required to drive the gradient-based optimization [21].

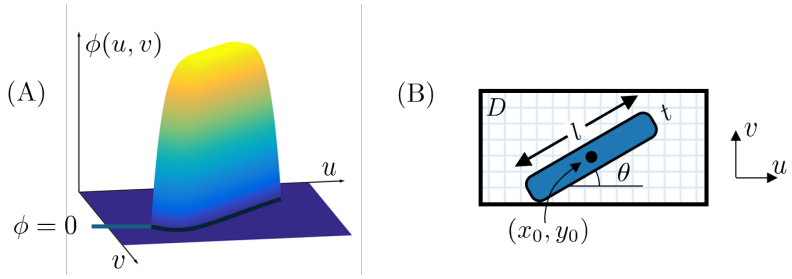


Figure 5: A MMC based parameterization: (A) the level-set function of Equation 11 (adapted from [34]); and (B) the projection onto the FEM mesh showing the beam spanning multiple elements.

We explicitly denote the level-set with index j to correspond to the j-th level-set function. This means a d-dimensional design vector is split-up in parts of five-dimensional design vector $\mathbf{x}_j$ such that $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_j = (x_{0,j}, y_{0,j}, \theta_j, l_j, t_j)$, and n is the number of beams. This means in this case, the dimension d of the parameterization of n beams is a multiple of five $d = 5n$.

Bujny et al. have shown promising application of this parameterization using an EA approach [24]. They first validated on simple static load cases to compare with the gradient-based SIMP. And then applied it to complex crash scenarios, for which no gradient-based optimization method is available. The EA used for this research was the CMA-ES method which will be discussed in subsubsection 2.6.2. In a follow-up paper by Raponi et al., this was extended by comparing the EA approach with a special-purpose BO variant [35]. Showing both BO and EA as promising candidates for TO problems, especially ones where gradients are unavailable.

2.2.3 Curved Movable Morphable Component Beams

Curved MMCs add five additional parameters per beam to the original MMC parameterization to deform the beam [36]. First the length of the beam is split-up in l_L and l_R, the left- and right length respectively (still from center (x_0, y_0)). The thickness t is split into three parameters t_L, t_M, and t_R, denoting the left, middle, and right thickness respectively. The intermediate thickness of the beam is interpolated using these three thickness parameters (this will be discussed in more detail later in subsubsection 3.4.3). Additional to this, a sinusoidal deformation is added to the centerline of the beam in the form of:

$$f(u') = a \sin(bu'). \tag{12}$$

In here a and b control the deformation, and u' is the u coordinate in the θ-rotated coordinate frame, corresponding to the beams orientation (Figure 6).

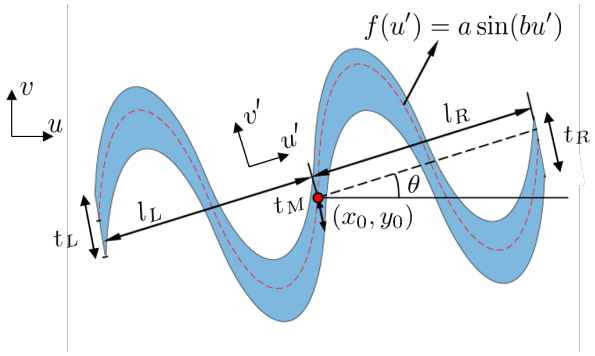


Figure 6: A schematic overview showing the deformation imposed by the additional parameters introduced in the curved MMC parameterization (adapted from [36]).

Yet, these additions do not lend itself easily to a single closed-form expression as the original, for a mathematical treatment see the original paper: [36]; explanation on how we handle the deformation is discussed later (subsection 3.4). Since the additional five deformation parameters per beam, now the dimension d of this parameterization of n beams is a multiple of 10 $d = 10n$.

2.3 Black-Box Optimization

In BBO the goal is to minimize, or optimize in general, some cost function f which is called the objective. This cost function relates a design vector $\mathbf{x} \in \mathcal{X}$ to a performance metric $f : \mathcal{X} \to \mathbb{R}$. Here we denote $\mathcal{X}$ as the design space; the set that contains all configurations that can be evaluated. The absence of a closed-form expression or insight into the internal workings of f is what characterizes it as a black-box. Analogous to a mountainous landscape on earth, moving within the search space $\mathbf{x} \in \mathcal{X}$, heights $f(\mathbf{x})$ are attributed to each point, forming a so-called fitness landscape.

It could be the case a subset of designs are infeasible, i.e. f can be not be evaluated, or it makes no sense to spend compute on it. For example in TO, disconnected designs are infeasible since it is impossible to fabricate floating structures. Still conducting the simulation with such a design could lead to spurious a response. Yet a design exceeding its volume target could be feasible in the sense of fabrication but is not in terms of volume.

To describe infeasibility, optimization problems include constraint functions $g : \mathcal{X} \to \mathbb{R}$. Whenever a constraint is activate $g(\mathbf{x}) > 0$, the design deemed infeasible and the objective function f is not evaluated. Incorporating such a constraint can be done using several tactics, each with implications for the optimization algorithm's performance and the nature of the solution found.

These constraints can be binary in nature: active $\mathbf{x} \mapsto 1$ or inactive $\mathbf{x} \mapsto 0$. Optimizers benefit from expressing how far a design is from feasibility. For example in TO when posing a material budget of 50%, a design exceeding the budget by 10% is closer to feasibility than a design exceeding the budget by 25%. Hence it is intuitive to penalize the latter more to create a valley in the fitness landscape towards feasible regions.

Conventionally, in general the optimization framework is written as:

$$\begin{aligned} \text{minimize:} \quad & f(\mathbf{x}) \text{ for } \mathbf{x} \in \mathcal{X} \\ \text{subject to:} \quad & g_i(\mathbf{x}) \leq 0 \quad \forall i \end{aligned} \tag{13}$$

In here i is a index for constraint functions indicating that all constraints should be met to be feasible. This means $\mathcal{X}$ can be partitioned into infeasible- ($\times$) and feasible ($\circ$) regions:

$$\mathcal{X}_\circ = \{\mathbf{x} \in \mathcal{X} : \forall g_i(\mathbf{x}) \leq 0\}; \qquad \mathcal{X}_\times = \mathcal{X} \backslash \mathcal{X}_\circ. \tag{14}$$

Then the optimal solution(s) $\mathbf{x}^\star \in \mathcal{X}_\circ$ are denoted with

$$\mathbf{x}^\star \in \underset{\mathbf{x} \in \mathcal{X}_\circ}{\arg\min} f(\mathbf{x}). \tag{15}$$

That is $f(\mathbf{x}^\star) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}_\circ$. Note it may be the case that multiple solutions that yield the same minimal achievable evaluation, for example due to symmetries in the design space. In practice for real-world optimization problems, finding this optimal solution $\mathbf{x}^\star$ is extremely hard, if not impossible. Fundamentally it means either:

1. all feasible designs should be evaluated in a total enumeration; or

2. the underlying structure or behavior of f should be known.

In continuous spaces, the first is impossible. The second is never true since it is a black-box problem, there is no knowledge about the intrinsics of the evaluations. Based on observation of evaluations alone it is impossible to say a solution $\mathbf{x}'$ is optimal or not, rather it is the best-found. That is, often it can be said some solution is a local optimum. It means within a bounded neighborhood $N_\epsilon(\mathbf{x}) \subset \mathcal{X}$ the solution $\mathbf{x}'$ is the best, but unless point (1) or (2) is satisfied, it can never be guaranteed this solution is the best globally.

Critically, BBO methods make assumptions about the structure of the objective landscape, despite it being unknown. These assumptions guide the choice of optimization strategy, as different methods exploit different landscape features (e.g., smoothness, locality, or structure). However, the shape of the landscape varies significantly between problems, even within the same class, due to changes in constraints, or parameterizations. This variability means no single strategy performs best across all problems; instead, the effectiveness of an optimizer depends on how well its underlying assumptions align with the landscape induced by the problem.

2.4 Landscape Characteristics and BBOB functions

Over the decades, many conventions have been developed to describe different characteristics of fitness landscapes [37]. Figure 7 shows three examples of selected BBOB functions from different groups that represent different landscape characteristics. BBOB functions are grouped into categories based on their structural properties, such as separability, conditioning, modality, and global structure. These groupings help benchmark optimization algorithms under varying levels of difficulty and landscape features.

1. Separable: Variables can be optimized independently; low interaction.

2. Moderate Conditioning: Mild variable interactions; moderately scaled.

3. High-Conditioned and Unimodal: Strongly stretched landscapes; sensitive to search direction.

4. Multi-modal with Global Structure: Many local optima but with exploitable global trends.

5. Multi-modal with Weak Global Structure: Deceptive landscapes with many traps and little usable global guidance.

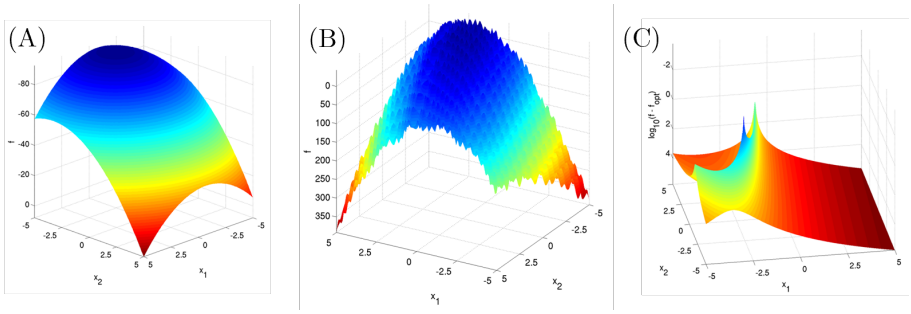


Figure 7: A three panel of example BBOB functions to be maximized: (A) f1 Sphere (group 1); (B) f8 Rosenbrock (group 2); (C) f15 Rastrigin (group 4) [38].

Modality of the fitness landscape refers to the number of local optima. *Multimodal* problems are generally harder to solve as some optimization strategies get trapped in *basins of attraction*. These are local regions in the fitness landscape that form a valley towards the local optimum. *Unimodal* problems contain only a single (global) optimum, but are rare in complex real-world problems. These are often much easier to solve but still might suffer from plateaus for example.

Plateaus are flat regions of the landscape: a change of decision variables does not lead to change in the objective. In the case of TO this could be caused by parameters describing geometry outside of the material domain, changing these parameters does not affect the geometry inside the material domain and hence the objective. Lack of structure is obviously not beneficial to any search procedure, and simulation budget may be easily wasted on exploring regions such as these.

Ruggedness quantifies rapid changes in the objective over small regions. A high ruggedness is caused by many small peaks and valleys, which can again trap local search methods. This often goes hand-in-hand with a high modality since frequent optima (modality) imply frequent directional changes (ruggedness). However, when an objective is noisy but has an unimodal global structure, the landscape may exhibit high ruggedness despite low modality. This mostly affects gradient-based approaches.

Separability refers to the measure of how much an objective can be decomposed into subproblems defined on subsets of the decision variables. Strong interactions among decision variables might complicate the search. For example $f(x_1, x_2) = x_1^2 + x_2^2$ can be decomposed as $f(x_1, x_2) = f_1(x_1) + f_2(x_2)$ where $f_1(x_1) = x_1^2$ and $f_2(x_2) = x_2^2$. This separability means the decomposed objectives can be be optimized individually. The objective $f(x_1, x_2) = (x_1 + x_2)^2$ for example contains the $2xy$ interaction term.

Convex-, such as the parabolic above, and *linear* functions are well-studied and easier to optimize [39]. Real-world objectives are rarely perfectly convex or linear, often exhibiting non-convexities due to noise, discontinuities, or complex interactions.

Ill-conditioning refers to situations where small changes in one direction

leads to a much bigger change in output than others, often due to steep cliff. This typically arises when the function's level-sets are highly elongated, such as in ellipsoidal shapes with large aspect ratios. If not accounted for, algorithms may take many small steps in the flat directions or oscillate in steep ones.

Whenever the dimension d increases, fully covering the search space becomes exponentially more expensive, a phenomenon known as *the curse of dimensionality* [40]. For example dividing each search variable in 10 steps leads to a total of 10^d search points: in 1D a line, yields 10 search points; a 2D plane, 100 search points; and a 3D cube, 1000 search points. For a 10 dimensional search space this means 10 billion search points. Clearly, *exhaustive search* is computationally infeasible, not even to speak about the coarse resolution of 10 points per variable.

2.5 ELA Features

To characterize the previously introduced landscape features, the original ELA paper proposes a several set of expert-designed features [28]. Such a set addresses observables of the landscape, descriptive of a certain trait of the problem. To calculate the features, a dataset is collected consisting of design vectors $\mathbf{x} \in \mathcal{X}$ and corresponding objective responses $f(\mathbf{x})$. An unbiased Latin Hypercube Sampling (LHS) strategy is used for covering the search space, however as noted in the previous section, due to the curse of dimensionality this becomes progressively harder for larger dimensions. Typically the number of samples is linearly scaled with dimension.

Many of the originally proposed features are highly correlated, in a follow-up paper, a selection of the features is proposed to be sufficiently descriptive [41]. These were selected using the BBOB functions. Here the feature set is decomposed into a 2D projection using a Principal Component Analysis (PCA). We list these features with a description in Table 1.

Although for the TO problem the objective is the same: finding the material distribution that minimizes the compliance; the different parameterizations describe this problem in different ways. As discussed previously, this results in different landscape characteristics. If we can capture these different landscape characteristics through this ELA feature set, they could be correlated to BBO performance of different methods. This is shown to be the case for the BBOB functions [42].

Feature	Description
ela.distr.skewness	Skewness statistic of y-distribution [28].
ela.meta.lin_simple.intercept	The intercept of a linear regression model fitted on the landscape [28].
ela.meta.lin_simple.coef.max	The largest slope factor of the linear regression model [28].
ela.meta.lin_simple.adj_r2	The adjusted R^2 of the linear regression model [28].
ela.meta.quad_simple.-coef.min_by_max	Ratio of the smallest to the largest absolute values among the quadratic coefficients (squared terms only) in a regression model excluding interaction terms [28].
ela.meta.quad_simple.adj_r2	The adjusted R^2 of the quadratic regression model, excluding interaction terms [28].
ela.level.mmce_lda_25	For the 25% level-set of Linear Discriminant Analysis (LDA), the Mean cross-validation accuracy [28].
ela.level.mmce_qda_25	For the 25% level-set of Quadratic Discriminant Analysis (QDA), the Mean cross-validation accuracy [28].
ela.level.lda_qda_25	The ratio of the above metrics [28].
ic.eps_s	The information content settling sensitivity [43].
ic.eps_ratio	Half of the partial information sensitivity [43].
disp.ratio_mean_02	Ratio of the average pairwise distances among the top 2% best-performing points to the average pairwise distances among all points in the design space [44].

Table 1: Selection of the ELA features to be sufficiently descriptive of the BBOB functions, as presented in [41]

2.6 Evolutionary-based search

Evolutionary Algorithms (EA) as the name implies is a field that aims to solve problems using evolution-inspired mechanisms such as populations, cross-over among individuals, mutation, and selection [1]. Traditionally, the field is split up into:

- Genetic Programming (GP), finding programs/instruction sets;
- Genetic Algorithms (GA), finding coded representations;
- Evolution Strategies (ES), optimizing continuous design vectors.

We focus on ES, though all three share the same biologically-inspired aspects. The difference between GA and ES is that GA searches typically for coded representations reminiscent of DNA strands, whereas ES is used for continuous search spaces.

The genotype-phenotype mapping converts the encoded design vector (gen) into a observable (pheno). An individual within a population caries its design vector as DNA and is sometimes also attributed local information controlling its mutation procedure. The genotypes of individuals are recombined to form offspring: new individuals. Sometimes this recombination-mutation analogy is somewhat blurred and forms just a more general reproduction concept. Then according to selection, the weak individuals are removed from the population and a new stronger generation emerges. This is survival of the fittest, only the strong survive.

Let $\mu \geq 1$ denote the size of the population, and including generated offspring $\lambda \geq \mu$. In ES typically one of two selection schemes are used: the $(\mu + \lambda)$ or the (μ, λ) selection. In the first the fittest new generation is selected from both the old and the new individuals, the latter replaces the old generation with a complete new one, i.e. only from offspring. Figure 8 shows a flowchart of the standard EA. In here a population $\mathcal{P}^{(g)}$ at generation g consists of μ or λ individuals, with an individual from the search space $\mathcal{X}$.

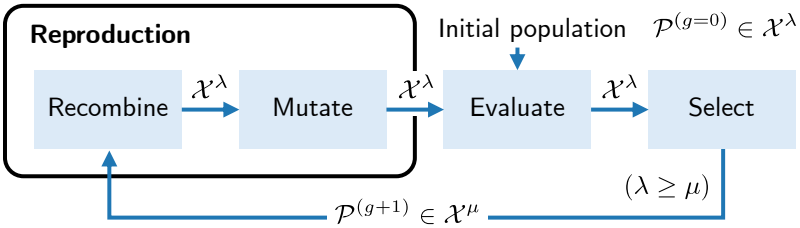


Figure 8: A flowchart for the general working principle of an Evolutionary Algorithm. $\mathcal{X}$ denotes the design space and $\mathcal{P}$ a population of individuals.

2.6.1 Differential Evolution

Differential Evolution (DE), a classic EA designed for continuous domains [45], incorporates a differential mutation update. In the reproduction stage for each individual (referred to as the target vector) $\mathbf{x} \in \mathcal{P}^{(g)}$ in the population, a potential offspring $\mathbf{x}'$ is generated. If this trial $\mathbf{x}'$ outperforms the original $\mathbf{x}$, it replaces it in the population for the next generation $\mathcal{P}^{(g+1)}$.

DE generates its offspring by first performing a mutation operation. For each target vector $\mathbf{x}$, three other unique individuals are randomly sampled from the current population: $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{P}^{(g)}$. This implies the population size is bounded: $\mu \geq 4$. Some subset of the coordinates of $\mathbf{x}'$ is sampled with the crossover rate to be replaced by some combination of $\mathbf{a}, \mathbf{b}, \mathbf{c}$, the leftover simply take on the value of the original coordinate in $\mathbf{x}$. If j is such a selected index for crossover,

then

$$x'_j \leftarrow a_j + F(b_j - c_j). \tag{16}$$

In here F is a differential weight, a variation controlling scaling factor. If j is not selected for crossover then,

$$x'_j \leftarrow x_j. \tag{17}$$

This crossover rate is usually set quite high, often around 90%, meaning on average only for 10% the original coordinates remain. This makes the terminology maybe a bit confusing. To be clear, $\mathbf{a}$ is the base vector which is mutated according to Equation 16. The difference $\mathbf{b} - \mathbf{c}$ gives direction and magnitude to the mutation, hence the name DE. Typically, $F \sim 80\%$. And, some arbitrary information is retained using the crossover mechanism. Depending on your standards, mating usually is performed in pairs; DE uses four individuals to generate offspring. This and the rates are motivated primarily empirically. This makes DE somewhat less grounded.

Then, the new offspring $\mathbf{x}'$ is added to the new generation $\mathcal{P}^{(g+1)}$ if it outperforms the original individual: $f(\mathbf{x}') \leq f(\mathbf{x})$. Otherwise, the original persists and is added to $\mathcal{P}^{(g+1)}$. This means DE falls into the $(\mu + \lambda)$ category, so-called elitist selection, a good individual can survive over multiple generations. In Figure 9 it is shown how this selection procedure starts to occupy the valley of the landscape towards the minimum.

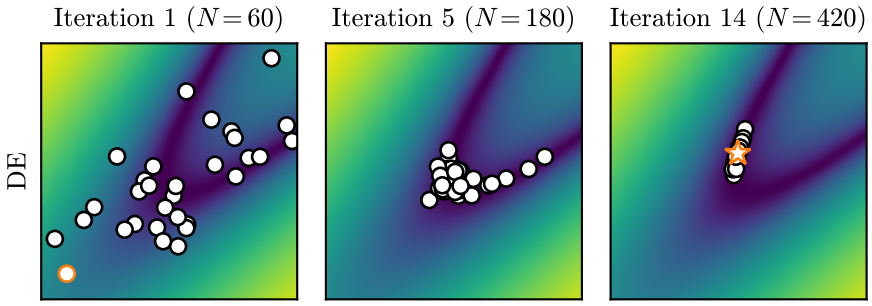


Figure 9: Iterations of the DE method showing convergence to the minimum of a landscape, with N the total number of evaluations.

2.6.2 Covariance Matrix Adaptation Evolution Strategy

Unlike selecting and evolving individuals like in traditional EA, Covariance Matrix Adaptation-ES (CMA-ES) models the population with a multivariate normal distribution [46]. Each generation offspring is sampled which is used to

evolve the parameters of the distribution.

$$\mathbf{x}_j^{(g+1)} \sim \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)}) \quad \text{with} \quad j = 1, \ldots, \lambda \tag{18}$$

In here $\mathbf{m}^{(g)} \in \mathbb{R}^n$ is the mean of the population, $\mathbf{C}^{(g)} \in \mathbb{R}^{n \times n}$ the covariance matrix, and $\sigma^{(g)}$ the step-size, all at generation g. After evaluating the λ offspring, the best μ are used to update the mean and covariance matrix. This means CMA-ES falls into the (μ, λ) category. For a mathematical treatment see [47].

In essence the goal of CMA-ES is to push the mean of the distribution downhill and align the covariance matrix with the contours of the landscape (Figure 10). Modeling the pairwise dependencies makes sampling offspring more effective in sampling towards local minima. Over the decades many update variations have been proposed including rank-one and rank-μ updates, step-size adaptation, and active covariance updates [47].

These refinements aim to improve convergence speed, robustness, and adaptability to non-separable or ill-conditioned problems. Also thanks to the modeling of the pairwise dependencies CMA-ES is invariant to rotation of the landscape which makes it suitable for non-separable objectives. Since it is gradient-free and has ranking-based selection, CMA-ES is robust to noise (ruggedness), scale transformations, and discontinuities in the objective function, making it well-suited for black-box optimization where derivative information is unavailable or unreliable.

Yet, still the stochasticity can result in widely different paths attributed to different basins of attraction. Population-based algorithms consequently are: local search algorithms, once converging to a particular region of the search space, it is impossible to jump to the other side to explore. The goal of the covariance matrix is to model the landscape locally to step into the right direction, but of course lacks model complexity to describe the landscape globally (Figure 10).

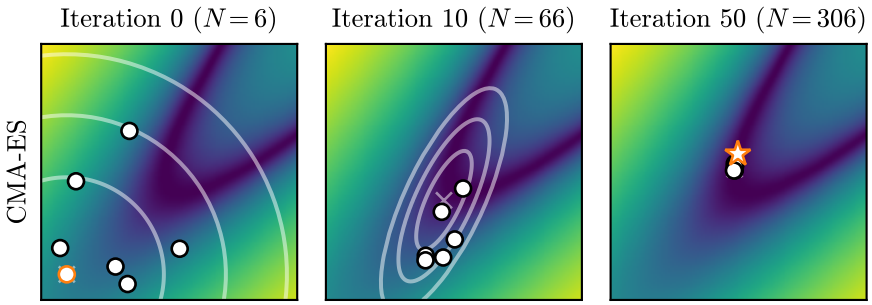


Figure 10: Iterations of the CMA-ES method showing updating of the population distribution converging towards the minimum, with N the total number of evaluations.

2.7 Bayesian Optimization

Bayesian Optimization (BO) is a sequential black-box optimization method that uses a probabilistic framework to model neighborhoods and uncertainty using a surrogate model to exploit and explore promising regions [2]. For continuous search spaces BO uses a Gaussian Process (GP) model to estimate the mean and uncertainty around observations. Moving further away from an observation, the less certain we are about its response (Figure 11). Generally this model-based approach is more suitable for engineering applications with tight budgets [48], such as TO where physics simulations take significant amounts of time.

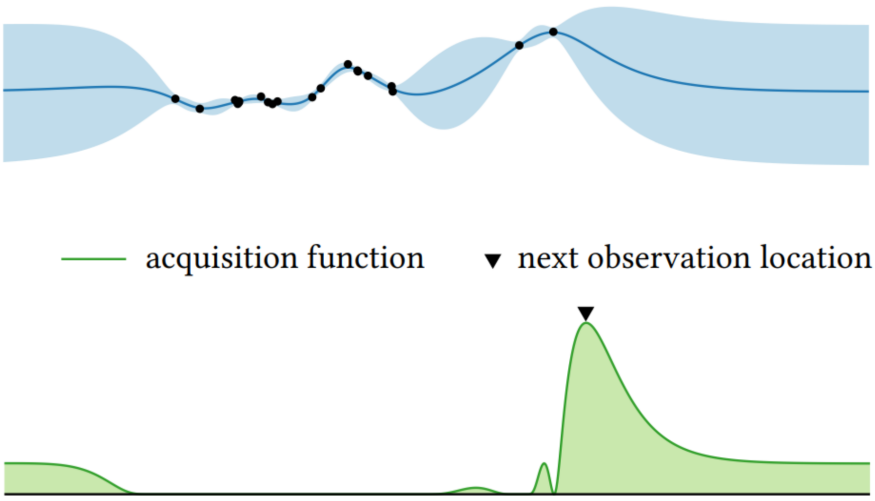


Figure 11: A probabilistic surrogate model of the objective function (blue), and selecting the next evaluation point by maximizing an acquisition function, which balances exploration and exploitation by favoring regions likely to yield better solutions, in this case attempting to maximize the function (adapted from [2])

Special Acquisition Functions (AF)s are devised to sample new designs; exploit the region when the model expects good designs, explore regions the model is uncertain about (Figure 11). Variants of AFs balances exploitation and exploration differently. Some examples include: Probability Improvement (PI), Expected improvement (EI), and Upper Confidence Bound (UCB) [48].

PI exploits more aggressively by suggesting points that have high probability to improve the current best based only on the mean prediction. EI also takes the uncertainty into account to maximize an improvement function; in general EI is more exploratory than PI. UCB defines the acquisition function by adding a tunable multiple of the model's uncertainty to the predicted mean. This means the exploration-exploitation trade-off in UCB is controlled through a hyperparameter.

Unlike the local search in EA, BO aims to model the global landscape. Typically, the model is updated after each observation and is applied on problems $d < 20$. For high dimensions this could lead to significant overhead in updating and sampling the surrogate models: the training complexity of GPs is $\mathcal{O}(n^3)$. Yet, this overhead in high cost simulation problems is assumed to be negligible in comparison to evaluation of the objective. BO is prevalent in hyperparameter optimization problems with tight budgets ~ 100; budgets of this magnitude are deemed small enough to ignore BO-induced overhead.

2.7.1 Heteroscedastic Evolutionary Bayesian Optimization

Heteroscedastic Evolutionary BO (HEBO) addresses the homoscedasticity and stationary assumptions made in Vanilla BO [49]. Real-world problems exhibit varying noise levels (heteroscedasticity) and changing characteristics (non-stationary) across the search space. This means that in HEBO compared to Vanilla BO, the surrogate model's behavior can vary across the design space depending on the region, both in modeling the mean and uncertainty. HEBO facilitates this by means of a parameterized input- and output transformations. Figure 12 shows iterations of the HEBO method applied on a simple 2D landscape. It shows both the modeling of contours of the landscape and its uncertainty of regions that are not explored (blue-shaded).

Besides this more flexible surrogate modeling, a GA is used to find good trial points among an ensemble of AFs, this is the "evolutionary" in HEBO. Since AFs might have somewhat of conflicting goals, in Vanilla BO selecting the AF becomes a hyperparameter on its own. HEBO counters this by evaluating several AFs and finding the non-dominated set (Pareto front); the set of trials where the score of no trial can be improved in one AF without simultaneously worsening at least one other AF. In principle it means, that several AFs are balanced to create a diverse set of trials not favoring one single sampling strategy.

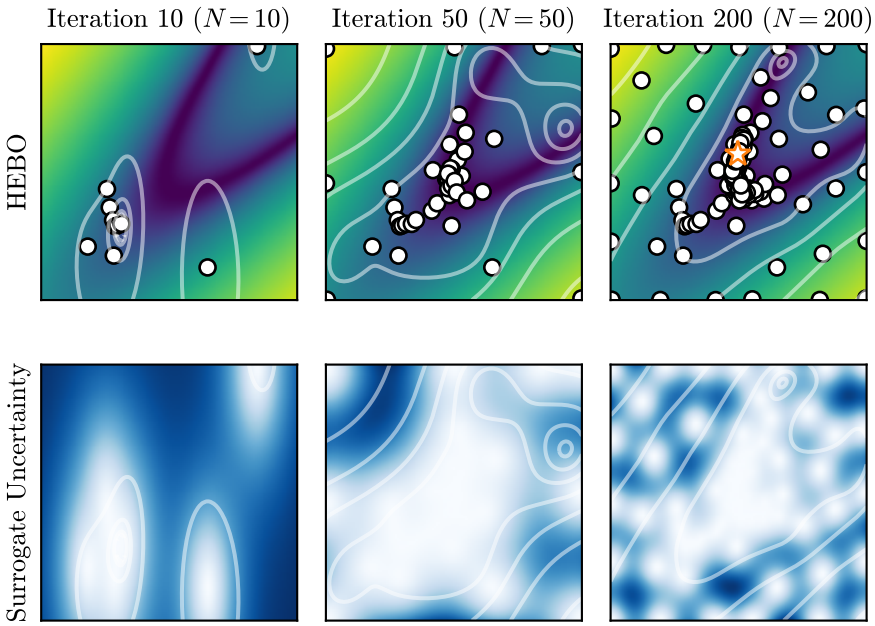


Figure 12: Iterations of the HEBO method showing the surrogate model progressively better modeling the contours of the landscape , with N the total number of evaluations.

3 Methodology

3.1 Problem Definition

In accordance to previous work [24, 35], we study a horizontal cantilever beam problem. The material domain is rectangular with a 2:1 ratio, its left-hand side is fixed, and a load is applied to the center of the right-hand side (Figure 13). This domain is subdivided into 100x50 square elements to form the mesh of the FEM setup. The goal is to minimize the compliance of a binary material distribution with a 50% material budget. To calculate this compliance the mesh is simulated using membrane mechanics, in accordance to [50].

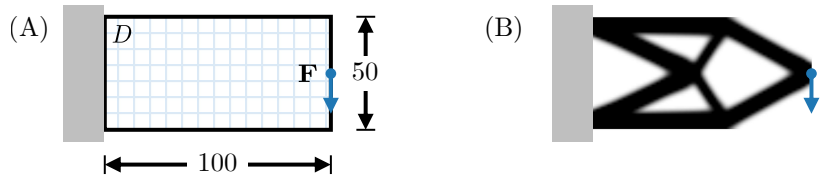


Figure 13: (A) A schematic overview of the studied problem, showing the design domain D and application of load $\mathbf{F}$. (B) The optimal topology obtained by SIMP for the same mesh using [51] (adapted from [52]).

Unlike most literature, no initial design is considered. Frequently, optimizers are warm-started using an initial design that both satisfies the volume constraint and is connected; this clearly biases search procedure. Previously, perturbations of diagonal designs were used as initial designs [24, 35], and choosing this initial design was found to be critical for finding good topologies.

Instead, our approach begins with a uniformly sampled design vector. This enforces a more challenging and unbiased optimization process, requiring the optimizer to construct viable structures from scratch. While this may result in slower early convergence, it provides a clearer measure of the optimizer's ability to discover high-performing designs without prior assumptions. Additionally, this setup avoids encoding any implicit topological preferences into the initial design, such as cross shapes in structural TO, to ensure a fair representative case for other problems where solutions are not intuitive.

Additional to the volume constraint, to limit unnecessary calls to the FEM simulation, a connectivity constraint is introduced which is violated if the design is disconnected. This leads to the general problem definition:

$$\begin{cases} \min_{\mathbf{x}\in[0,1]^d} & f(\mathbf{x}) \\ \text{s.t.} & g_1(\mathbf{x}) = \mathcal{V}(\mathbf{x}) - V_{\max} \leq 0 \\ & g_2(\mathbf{x}) = \mathcal{C}(\mathbf{x}) \leq 0 \end{cases} \tag{19}$$

where $\mathcal{C}(\mathbf{x})$ is connectivity constraint. Again, f represents the compliance objective, $\mathcal{V}$ the volume of the design, and $V_{\max}$ the volume budget. The volume

budget is set at 50% of the domain area.

Previously, Raponi et al. used an underlying graph representation of MMCs to constraint their connectivity [35]. The endpoints of the beam are connected with an edge, in a sense this forms the figurative bone of a MMC beam. Hence a skeleton of the structure can be devised as the collective of these bones, which is said to be connected whenever the underlying edges intersect and are connected to the edge- and load points. An objection to this method is that the thickness of the beams is not considered. That is, the rectangular beam is parameterized with two shape dimensions (t_i, l_i) from which thickness t_i is completely ignored even though t_i might be larger than l_i.

Besides neglecting the thickness of the beam, this method is exclusive to regular MMC beams. However, since we investigate the effect of parameterization on optimizer performance, we seek a more general connectivity constraint inspired by this graph representation, and works for any geometry generating parameterization.

3.1.1 Connectivity Constraint Proposal

To counter these problems, we propose a more general connectivity constraint which is independent from the parameterization by considering geometry it produces, opposed to an underlying skeletal structure. To do this, the shortest distance between the outline of connected components is used.

Several generated geometries might be overlapping and compose a single connected component. The j-th and k-th geomtries are connected if the intersection is non-empty: $\Omega_j \cap \Omega_k \neq \emptyset$. By enumerating the geometries in a graph and drawing edges between intersecting pairs j and k, connected components C_i are found (shaded in Figure 14).

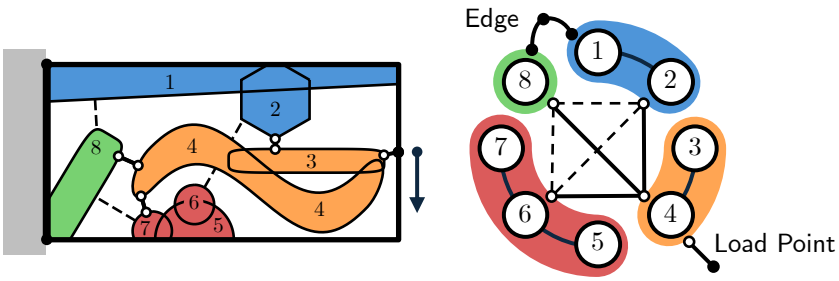


Figure 14: An example for calculating the least distance to connect four components of geometry to each other and edge- and load point boundary geometry.

Note even though if two components are not connected directly, they still can be part of the same connected component by means of an intermediate

$$\Omega_j \cap \Omega_k \neq \emptyset \text{ and } \Omega_k \cap \Omega_l \neq \emptyset \implies \{\Omega_j, \Omega_k, \Omega_l\} \subseteq C_i. \tag{20}$$

That is only if Γ_j never produces disconnected structures, otherwise it needs to be decomposed in separate connected structures. Hence we make the assumption the parameterization only produces geometries that are fully-connected. Besides internal connection of the structure, it also is required to be connected to boundary conditions such as an edge or point. We clearly make the distinction between these from $\mathbf{\Omega}_{\text{geo}} = \{\Omega_j\}$ as:

$$\mathbf{\Omega}_{\text{bounds}} = \{\Omega_{\text{pt},j}\} \cup \{\Omega_{\text{line},j}\} \tag{21}$$

$$= \{\mathbf{p}_j\} \cup \{(1-t)\mathbf{q}_j^{(1)} + t\mathbf{q}_j^{(2)} : t \in [0,1]\} \tag{22}$$

In here $\mathbf{p}_j$ represents individual boundary points, and $\mathbf{q}_j^{(1)}$ and $\mathbf{q}_j^{(2)}$ line control points to defining the geometry of line boundaries.

Now we attribute all distinct connected components a node in a graph $C_i \in V_C$ (Figure 14). We let the weight of edges in this connected components graph denote the shortest distance between the pair of connected components:

$$\mathbf{W}_C = [w_{ij}] = [w_{ji}] = [d_{\min}(C_i, C_j)]. \tag{23}$$

This means the graph is undirected and fully connected. We define the shortest distance function between two components as follows:

$$d_{\min}(C_i, C_j) = \min\{\|\mathbf{x}_i - \mathbf{x}_j\| : \mathbf{x}_i \in C_i \text{ and } \mathbf{x}_j \in C_j\} \tag{24}$$

This directly implies $d_{\min}(C_i, C_i) = 0$; the edge-weight matrix $\mathbf{W}_C$ is 0 on the diagonal. Now by assuming not all pairwise distances are identical, the minimum spanning tree (MST) of the graph can be found. The MST is the connected graph with the least amount of total edge weight [53]. Intuitively this is the shortest distance to connect the structure.

The weights matrix $\mathbf{W}_C$ is transformed into $\mathbf{W}_{C,\text{MST}}$ representing the MST graph by discarding the worst connections, i.e. setting them to 0.

We deliberately left out the boundary geometry out of this procedure since they might form a bridge to connect the geometry. For example in Figure 14, the lower left component and the top one are connected both to the line boundary, however they are hardly connected as both have no influence on each other and form not a connected structure. Therefore besides the distance of the MST, we add the least distance from geometry to boundary conditions explicitly, defining the connectivity constraint as:

$$\mathcal{C}(\mathbf{x}) = \underbrace{\sum_{(i,j)} (\mathbf{W}_{C,\text{MST}}(\mathbf{x}))_{ij}}_{\text{inter components}} + \underbrace{\sum_{\Omega_b \in \mathbf{\Omega}_{\text{bounds}}} \min_j d_{\min}(\Omega_b, C_j)}_{\text{components to boundaries}}. \tag{25}$$

For some fully connected design $\mathbf{x}^\circ$, there is only one connected component C_1. This means $\mathbf{W}_C = [d_{\min}(C_1, C_1)] = [0]$. Also when fully connected, it is connected to the boundaries so $d_{\min}(\Omega_k, C_1) = 0$ for all boundary geometries Ω_k. Hence $\mathcal{C}(\mathbf{x}^\circ) = 0$.

For disconnected designs $\mathbf{x}^{\times}$ there are either multiple connected components with such that $d_{\min} > 0$, or a singular component is not connected to the boundaries, or both. All three these scenarios imply $\mathcal{C}(\mathbf{x}^{\times}) > 0$.

3.1.2 Constrained Objective

To solve the optimization problem Equation 19 using black-box optimizers, the compliance objective for feasible designs is combined with the penalties from the constraints into a single objective: f_{obj} which is defined both for feasible- and infeasible regions. We first define the combined constraint as:

$$g(\mathbf{x}) = c(\tilde{g}_1(\mathbf{x}) + \tilde{g}_2(\mathbf{x})). \tag{26}$$

In here $c \geq 0$ is a scale-factor, and $\tilde{g}_{\square}(\mathbf{x})$ represents the normalized constraint such that $\mathbf{x} \mapsto [0, 1]$. That is: 0 represents no violation (feasible), and 1 represents the worst violation for the problem domain. In the case of the volume constraint this is just the exceeded volume fraction:

$$\tilde{g}_1(\mathbf{x}) = \frac{g_1(\mathbf{x})}{V_{\text{domain}}} = \frac{\max\{(\mathcal{V}(\mathbf{x}) - V_{\max}), 0\}}{V_{\text{domain}}}. \tag{27}$$

To prevent constraints balancing each other in $\tilde{g}_1(\mathbf{x}) + \tilde{g}_2(\mathbf{x})$, for example trading volume to violate the connectivity constraint, the max operator is used to restrict $\tilde{g}_1(\mathbf{x}) \geq 0$.

The normalized connectivity constraint can be obtained by normalizing the connected components C_j linearly relative to the design domain D:

$$D = [0, d_x] \times [0, d_y] \implies \tilde{C}_j = \{(x/d_x, y/d_y) : (x, y) \in C_j\} \subseteq [0, 1]^2. \tag{28}$$

From the normalized geometry a normalized version of the connectivity $\tilde{C}(\mathbf{x})$ can be calculated as:

$$\tilde{g}_2(\mathbf{x}) = \frac{\tilde{\mathcal{C}}(\mathbf{x})}{\sqrt{2}}. \tag{29}$$

The normalized connectivity is divided by $\sqrt{2}$ since this is the largest line distance in $[0, 1]^2$. Since $\mathbf{W}_{\tilde{C},\text{MST}}(\mathbf{x})$ by definition only has positive elements, like the volume constraint, $\tilde{g}_2(\mathbf{x}) \geq 0$. From this we can guarantee $g(\mathbf{x}) \geq 0$ in general, and $g(\mathbf{x}^{\circ}) = 0$ for a feasible design $\mathbf{x}^{\circ}$.

Note $\tilde{g}_2$ does not strictly to $[0, 1]$, it can exceed 1 for extreme scenarios. For example if the total geometry consists of the boundary conditions and a single point in the corner of the design domain:

$$\tilde{\mathbf{\Omega}}(\mathbf{x}) = \{\tilde{\Omega}_{\text{left-edge}}, \tilde{\Omega}_{\text{load-point}}\} \cup \tilde{\Omega}_{\text{geo}} \quad \text{where} \quad \tilde{\Omega}_{\text{geo}} = \{(1, 1)\}. \tag{30}$$

Then

$$\frac{\tilde{\mathcal{C}}(\mathbf{x})}{\sqrt{2}} = \frac{d(\tilde{C}_{\text{left-edge}}, \tilde{C}_{\text{geo}}) + d(\tilde{C}_{\text{geo}}, \tilde{C}_{\text{load-point}})}{\sqrt{2}} = \frac{1 + 0.5}{\sqrt{2}} \approx 1.06 \tag{31}$$

An objection to this method is that the aspect ratio is not preserved in the normalization. Although this effect might be negligible due to the relatively small aspect ratio 2:1, for more extreme domain this effect should be noted. An aspect-ratio preserving normalization of the geometry for domain $D = [0, d_x] \times [0, d_y]$ can be calculated as:

$$\tilde{C}_j = \{(x/\max\{d_x, d_y\}, y/\max\{d_x, d_y\}) : (x, y) \in C_j\} \subseteq [0, 1]^2. \tag{32}$$

Now we define the constrained objective on both feasible- and infeasible regions as follows:

$$f_{\text{obj}}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } g(\mathbf{x}) = 0, \\ \max_{\mathbf{x}' \in [0,1]^d} f(\mathbf{x}') + g(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{33}$$

The compliance $f(\mathbf{x})$ is computed only feasible designs, i.e. if $g(\mathbf{x}) = 0$, otherwise the design deemed infeasible gets its constraint score added to the worst feasible design compliance. In a sense this creates an artificial valley towards the feasible region and a smooth transition between infeasible and feasible.

This differs from the previous work in which the simulation was also ran using infeasible designs [24, 35], a penalty was only added to this objective response whenever the design was infeasible. However, we have to note this can depletes the simulation budget unnecessarily. Especially for unconnected designs, the compliance result for such a design in our case can be a factor 10^9 as big as an arbitrary connected design. Yet, in the case of a connected design that exceeds the volume budget by a small amount such a penalty strategy could be useful, as the resulting objective values remain informative and within a reasonable scale, guiding the model toward feasible regions.

Our definition (Equation 33) is somewhat paradoxical; the objective requires knowledge of the worst possible feasible design which is basically the inverse objective. To a certain extent finding $\max f(\mathbf{x})$ might be of same complexity as finding $\min f(\mathbf{x})$. It could lead to several tactics: (1) waiting until sufficient amount of feasible designs are evaluated to make a reliable estimate; or (2) make an educated guess based on domain-knowledge. This educated guess might be one based on an intuitive bad design, or one based on a simplified mathematical model for example.

In our case, we consider two candidates for bad feasible designs: a regular thin horizontal bar, and a thin outline tracing the border of the domain (Figure 15). Both yield compliance values around but below 500, hence we approximate $\max f(\mathbf{x}) \sim 500$. When increasing the thickness of the designs, compliance drops quickly and is found below 1 for a material budget of 50%. Since we therefore estimate the range of f spanning three orders of magnitude, we set the scale of the constraints to the same range $c = 500/0.5 = 10^3$. This approach is deliberately chosen to not tune c for one of the optimizers directly to make a fair comparison, and mimic a minimal simulation budget approach.

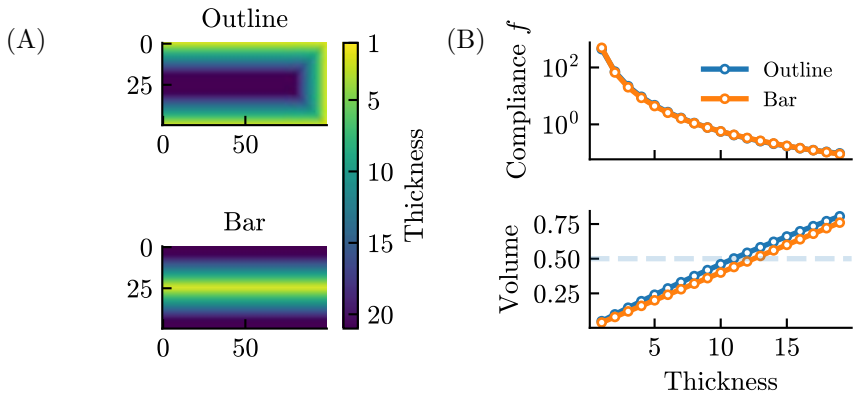


Figure 15: (A) The two designs used for estimating the worst compliance with increasing thickness: bar and outline. (B) Corresponding compliance and volume fractions corresponding to the designs presented in (A).

3.2 Program Structure

The program is build-up as modular as possible to facilitate different optimizer-parameterization combinations. Since the optimizers are "out-of-the-box" black-box optimizers, they essentially only require just a function f that responds the system's response for a trial design vector $\mathbf{x}$. This means when a problem is queried, first the parameterization realizes the design. Then the connectivity constraint and volume constraint are checked. When they are met, the design goes through the simulation and the systems response is returned. If not, a penalized response is returned and the simulation budget N is not decremented (Figure 16).

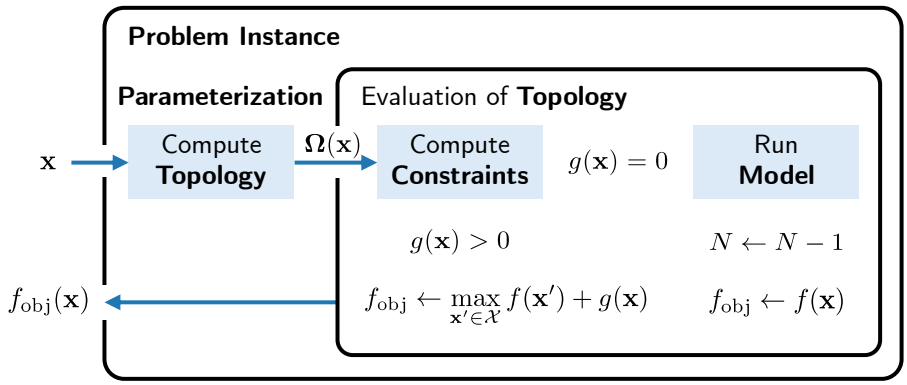


Figure 16: A flowchart showing the basic principles of the program.

A ProblemInstance contains a Topology data class, Parameterization, Constraints, Model, and a Objective. Once queried the Parameterzation alters the geometry (polygons) and the rasterized geometry (mask) in the Topology. The Topology is passed into the Constraints to calculate the calculate the connectivity and volume using the polygon representation. If met, the rasterized mask representation is passed into the Model. The state of the Model is updated using this new Topology. The Objective is a mapping from Model state to score. If the Objective is called, the simulation budget N is decremented by one, if only the constraints are used not. Essentially we see the constraints as "free" while the Model update involves a high-cost calculation. The core modular Python setup, an implementation for membrane physics model, and a problem builder for the horizontal cantilever problem can be found on: github.com/jelle-westra/TO.

3.3 Practical Implications

As stated above, the geometry is calculated in the parameterization in the form of polygons. This is done by first calculating the outline of the parameterized shape, and placing it inside the design domain. This polygon is generated using the Python Shapely package, a package that facilitates polygon operations [54]. From here the connected components are found as proposed in subsubsection 3.1.1 by using union operations, and after obtaining the connected components, the distance between polygons (of the connected components) is calculated. After this the MST can be calculated with Scipy [55], and the volume constraint can be directly calculated from the polygon. We discuss the rest of the used packages, and their use cases later in subsection 4.1.

Since the geometry must be rasterized onto the FEM mesh, it is possible that a connected continuous representation becomes disconnected after rasterization, especially in the case of thin structures. This is possible if the outline of the polygon partially intersects a pixel of the mesh (Figure 17A). This could result in a simulation call of a disconnected mesh.

We choose therefore before calculation of the constraints to rasterize the

geometry to mesh. From this rasterized version we make again a polygon from which the constraints can be easily calculated. This also ensures the calculated volume corresponds to the same volume as used in the FEM mesh.

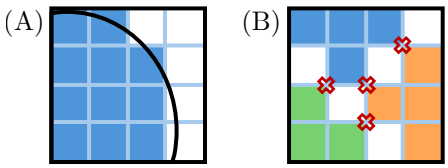


Figure 17: (A) Rasterization of a continuous geometry onto a discrete mesh. (B) Illegal checkerboard connections found in grid-based parameterizations and after rasterization of continuous geometry. The different colors represent different components and illegal connections are depicted with red crosses.

A problem that comes with prior rasterization before checking the connectivity constraint is that the checkerboard problem can occur. When pixels are solely connected diagonally, with one point, they are considered to be not connected. To facilitate this, when calculating the connectivity after rasterization, we shrink the components by a tiny fraction to separate the components. We calculate weight in the component graph (Figure 14) between the components now as follows:

$$w_{ij} = \max\{d_{\min}(C_i, C_j), \mathrm{px}_w/2\} \tag{34}$$

Recall, C_i and C_j denote the i-th and j-th connected components respectively, and $d_{\min}$ finds the minimal distance between the two, and we define px_w as the pixel width, assuming the pixels are square. This ensures full connections as depicted with the shaded regions in Figure 17B; pixels may only connect to a diagonal by means of an intermediate side connection.

3.4 Parameterizations

We briefly discuss the implementation details of the different parameterizations introduced in subsection 2.2. We define a d-dimensional parameterization always to operate on the design space $[0, 1]^d$.

For all parameterizations y-symmetry is imposed, meaning the design domain D is split in half by a horizontal symmetry line. Any geometry drawn using the parameterization will be mirrored among this line Figure 18. Since, by definition we can only define geometry within the domain such that $\Omega_j \subseteq D$, we clip-off geometry produced outside the design domain (Figure 18).

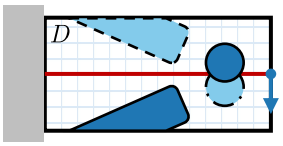


Figure 18: Mirroring of geometry among the (red) horizontal symmetry line, dark shaded geometry is generated by the parameterization, whereas the light shaded is imposed by the symmetry condition.

3.4.1 Honeycomb Tiling

The Honeycomb Tiling achieved by instancing a grid of points representing the centers of the tiles. All the tiles are pre-computed and activated when its corresponding design variable exceeds half of its search domain: $x_j > 1/2$.

Since we operate on a rectangular design domain, there does not exist a perfect hexagonal grid fitting exactly inside, and completely filling the entire domain. Hence the tiling will be projected on top of the domain. Also to comply with a certain amount of tiles, it is sometimes necessary that tiling can be stretched. Figure 19 shows two orientations the tiling can have. Given the symmetry line we continue with horizontal orientation shown in Figure 19B and Figure 19C.

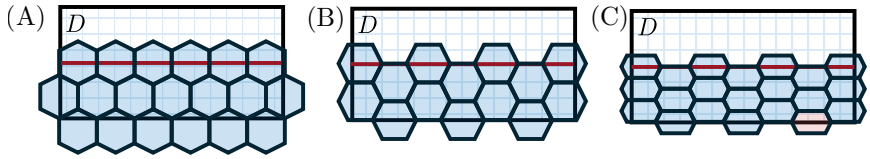


Figure 19: Examples of Honeycomb tiling: (A) in a vertical orientation; (B) a horizontal orientation; (C) in a horizontal orientation with stretched tiles.

For fair comparisons with MMCs and Curved MMCs, we will only compare parameterizations of the same dimension. However, this implies we should create honeycomb grids of a particular dimension to be comparable, obviously this is not always possible. For example Figure 19C contains 21 cells. For instance comparing 20D parameterizations, requires the removal of one cell. For our experiments, we remove the cells that are the least contributing to the design domain: the bottom row. Given that the structure needs to be connected to the left wall, we start from the right hand side removing cells (shaded in red Figure 19C).

3.4.2 MMCs

In previous work, authors rasterize their beams directly to mesh using the implicit level-set function ($\phi_j : D \to \mathbb{R}$ (Equation 11). However, since we compute

the geometry in form of polygons explicitly, suiting the connectivity constraint, instead the boundary is calculated directly:

$$\tilde{\Gamma}(s) = \begin{pmatrix} \operatorname{sgn}(\cos 2\pi s)|\cos 2\pi s|^{2/m} \\ \operatorname{sgn}(\sin 2\pi s)|\sin 2\pi s|^{2/m} \end{pmatrix}. \tag{35}$$

In here $\tilde{\Gamma}$ represents the points on the boundary of the normalized hyperellipse, defined parametrically on $s \in (0, 1)$, and m again is the shape controlling parameter set at $m = 6$. This normalized shape corresponds to the MMC where length and thickness both are $l = t = 2$, the radius of the normalized hyperellipse is 1. Also this normalized shape has no orientation $\theta = 0$, and is centered at the world origin $(x_0, y_0) = (0, 0)$.

Using Shapely's Polygon operations the shape is scaled to correspond to its dimensions: length l, and thickness t. After which it is rotated by θ and translated to position (x_0, y_0) (Figure 20). This corresponds to the same geometry as obtained by using the level-set definition (Equation 11). The domain $s \in [0, 1]$ is sampled using 1000 linearly spaced points to ensure a sufficiently smooth shape before rasterization.

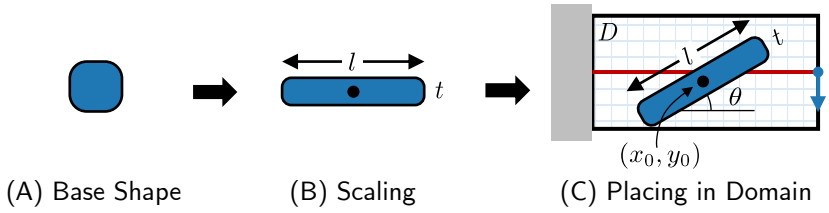


Figure 20: Overview of generation of geometry in the MMC parameterization: (A) first generation of the normalized base shape; (B) then scaling to correspond to the right dimension; (C) and lastly placing it and orienting the geometry in the design domain.

Capsule shape proposal Previous works use this hyperellipses as approximation of rectangular beams, this creates smooth continuous boundaries suitable for gradient-based optimization [21]. However, since gradients are not used in BBO, various other beams can be devised like with sharp discontinuous corners like a rectangle. Since we use generic transformations to generate the geometry, we only have to substitute the generation of the normalized base shape (Equation 35).

After rasterization, a possible drawback of the hyperellipse beam representation is that the thickness of the beam slowly decreases approaching the corners (Figure 21). In the case of a 1 pixel thick beam, this even could result in disappearing geometry towards the endpoints (disconnected geometry). Of course the optimizer should be able to deal with this problem given the response of the objective. However, the we attempt to avoid this entirely by proposing a more cohesive capsule shape.

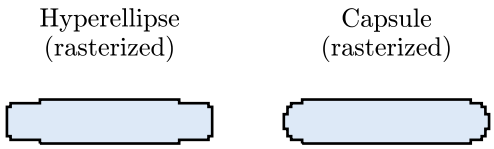


Figure 21: Examples of the hyperellipse- and capsule shapes using the same dimensions after rasterization.

Before rotating and translating the shape into the design domain, the capsule can be build up by connecting two semi-circles, with radius $t/2$ and placed $l/2$ apart. This still gives the advantage of the smooth endpoints but keeps the beam thickness constant. This also avoids having to define a global shape defining parameter such as $m = 6$ in the hyperellipse case. One could also argue to use a rectangle with rounded corners but this also results in the former problem: you either have to define a global hyperparameter that controls how much of the shape is rounded, or an additional parameter to the beam formulation is needed.

Endpoints representation proposal The angular representation of beams may suffer from describing geometry outside the domain. Consider the case where a beam's origin is placed at the corner of the domain at an angle (Figure 22). Since the beam is clipped by the domain boundaries, increasing its length l has no effect to the objective; the geometry within the domain stays simply the same.

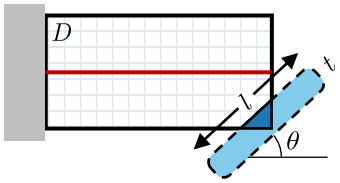


Figure 22: An example of geometry being described outside the design domain, when placing the beam in the corner of the domain using the angular MMC parameterization.

Also not suiting a local search procedure for a cyclic parameter like θ might get the procedure stuck. Assuming a ranking based procedure like EAs, once θ reaches steps towards a local minimum around the boundary, it is unable to jump the other side of the search space, even though these geometries are similar. For example the optimizer approaches $\theta \approx 0$ (representing 0°), since $\mathcal{X} \in [0, 1]^d$, it is impossible to turn beam any further to negative angles, even though the objective might be better. This means without explicit adaptation

of this cyclic behavior, the optimizer has to jump to the other side of the design space to represent this geometry.

We therefore propose another representation of the beam using two endpoints (x_1, y_1) and (x_2, y_2), and a thickness t (Figure 23A). This means we prevent defining geometry outside the domain and having a cyclic parameter. If we let the endpoints describe the caps of the geometry and allow them any position in D, only the tip of the beam can be connected to the wall. To prevent this the geometry is extended by $t/2$ in both directions to allow the beam to be connected to the domain bounds with full thickness.

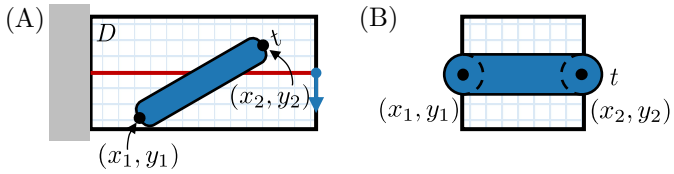


Figure 23: Proposal of the endpoints parameterization of MMCs: (A) a beam placed inside the design domain; and (B) slightly extending the geometry by half the thickness to ensure the beam can be connected to the domain boundaries with full width.

A representation like this can easily be transferred into the original. Let,

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \end{pmatrix}. \tag{36}$$

Using these we can transform the endpoints representation into the angular and use the same method for calculating the geometry. The endpoints representation is converted into angular as:

$$\begin{pmatrix} x_0 \\ y_0 \\ \theta \\ l \\ t \end{pmatrix} = \begin{pmatrix} x_1/2 + x_2/2 \\ y_1/2 + y_2/2 \\ \arctan(\Delta y)/(\Delta x) \\ t + \sqrt{(\Delta x)^2 + (\Delta y)^2} \\ t \end{pmatrix}. \tag{37}$$

However, this representation comes at the cost of adding a symmetry: the endpoints can be swapped to generate the same geometry.

This means we can either parameterize MMCs using the quintuple: (x_0, y_0, θ, l, t) for angular representation; or (x_1, y_1, x_2, y_2, t) for the endpoints representation. I both cases 5 parameters are necessary to parameterize a MMC beam meaning a parameterization of n beams results in a search space dimension of $d = 5n$.

Permutations of Beams Problem We have to note the following issue when it comes to the MMC parameterization, namely symmetries. Consider a 15D parameterization of three controllable MMC beams, which become six after the symmetry condition (Figure 24A). Let the three 5D decision vectors

$\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(3)}$ represent the first, second, and third beam of this 15D parameterization.

These 5D blocks are independent from each other; any permutation of these blocks result in the same geometry. To put it simply it, it does not matter in what order we draw the beams. For example, $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)})$ describes the same geometry as $(\mathbf{x}^{(3)}, \mathbf{x}^{(2)}, \mathbf{x}^{(1)})$.

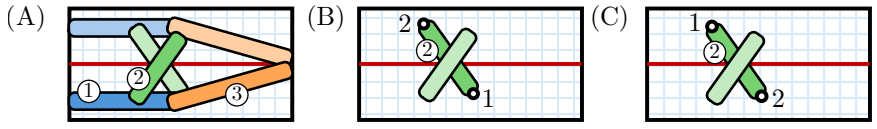


Figure 24: Schematic overview of symmetry implications: (A) order of drawing does not affect geometry; (B) The cross-shape in (A) can also be drawn by using the mirrored version of beam 2; and (C) swapping the ends points in comparison to (B) also produces the same geometry.

In fact, all permutations of these blocks will result in the same geometry. Let us assume we control n beams. Then for each configuration there are $n!$ configurations that produce exactly the same geometry simply by changing the beam order.

Assuming we use the endpoints parameterization, the symmetry condition itself also produces another symmetry in the design space. As shown inFigure 24B, the same cross as in Figure 24A can be drawn by making use of the symmetry condition. Since this can be applied for each beam individually, it introduces 2^n symmetries. Additionally, the endpoints for each beam can be swapped to produce the same geometry. This again, produces another 2^n symmetries. Leading to a total of

$$\text{TotalSymmetries}(n) = n! \cdot 2^{2n} \tag{38}$$

symmetries for a parameterization controlling n MMC beams represented by endpoints. Obviously, this scales poorly with an increasing number of beams (Table 2).

No. Controllable Beams n	No. Beams After Symmetry	No. Permutations $n!$	2^{2n}	Total Symmetries
1	2	1	4	4
2	4	2	16	32
3	6	6	64	384
4	8	24	256	6,144
5	10	120	1,024	122,880
10	20	3,628,800	1,048,576	3,805,072,588,800

Table 2: Calculation of the number of symmetries in the design space caused by defining MMC beams as endpoints.

3.4.3 Curved MMCs

To keep modularity of the code, a deformation of beams is defined as additional parameters to the original MMC parameterization. For the thickness we add the left- and right thickness t_L and t_R respectively, we let the original thickness t of the MMC represent the center thickness t_M. The deformation for curved beams is only performed in the local coordinate frame of the beam [36]. This means we can apply the deformation before transforming it in the design space.

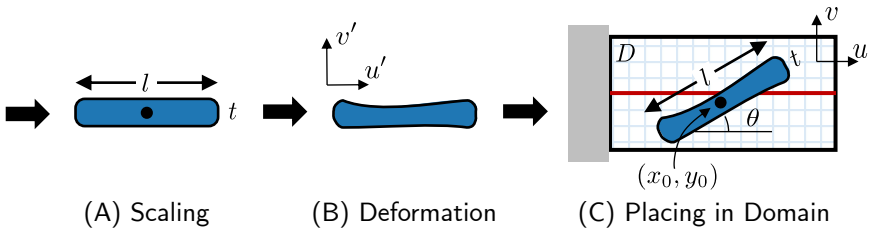


Figure 25: Overview of generation of deformed geometry in the curved MMC parameterization: (A) then scaling the normalized base shape as presented earlier; (B) performing the deformation in the local coordinate frame (u', v'); (C) and lastly placing it and orienting the geometry in the design domain.

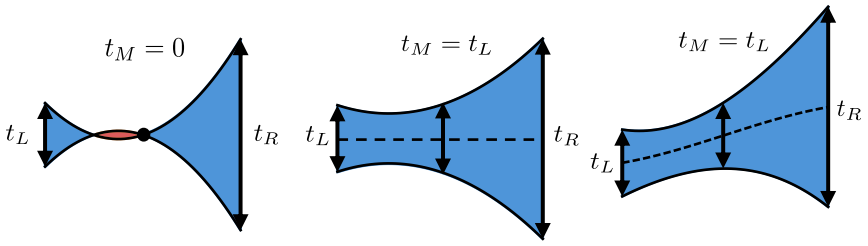
In general, after computing the base shape and applying scaling, the points defining the polygon boundary are not equally spaced. For example, in a capsule shape, a long straight edge connects the two semi-circular ends. This edge can only deform smoothly if it is divided into smaller segments; otherwise, only the endpoints can move, resulting in a straight edge again after deformation.

To enable smooth deformation across the entire boundary, including straight sections, the polygon is re-sampled with 1000 equidistant points before deformation.

Recall the deformation of Curved MMCs consists of two components: parabolically interpolated thickness, and a sinusoidal offset. If we let (u', v') denote a point on the boundary of the beam in the local coordinate frame, then this deformation is given by:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} \mapsto \begin{pmatrix} u' \\ f_t(u')v' + f_{\mathrm{CL}}(u') \end{pmatrix} \tag{39}$$

Here $f_t(u')$ denotes a scaling factor for thickness, and $f_{\mathrm{CL}}(u')$ is the sinusoidal offset of the centerline. Figure 26A,B shows what it looks like to first apply the parabolic interpolation of thicknesses, and in Figure 26C, we also include the centerline offset.



(A) Self-Intersection (B) Valid Variable Thickness (C) Full Deformation

Figure 26: Examples of deformations on the beam where: (A) the problem of self-intersecting geometry is possible for some thickness combinations; (B) showing a valid interpolation of thicknesses; and (C) showing the full deformation including the sinusoidal offset.

About the sinusoidal deformation we note the following: decomposing of the length l into left- and right components is de facto the same as adding a phase; both effectively shift the center point (x_0, y_0) of the deformation along the horizontal centerline (see original presented image Figure 6). This means the regular MMC parameters can be kept as they are, and the deformation can be defined on top of the original parameterization. Also, the center point of the beam stays at origin of the local coordinate frame, which benefits transformation to the design domain.

To realize this, we swap the decomposition of l into l_L and l_R from the original proposal [36], for a phase variable in the sine deformation:

$$f_{\mathrm{CL}}(u') = a \sin(b(u' + c)). \tag{40}$$

Again, a and b control the deformation, and u' is the u coordinate in the θ-rotated coordinate frame, and c is the our proposed phase-shift variable.

The parabolic scaling factor can be calculated as follows:

$$f_t(u') = (t_L + t_R - 2t_M)/4 \cdot (2u'/l)^2 + (t_R - t_L)/4 \cdot (2u'/L) + t_M. \tag{41}$$

Since the endpoints of the beam stretch from $-l/2$ to $l/2$, u' is normalized to [-1,1] by dividing it by $l/2$. For some combinations of t_L, t_M, and t_R, this interpolation scheme can result in negative thickness for some parts of the beam (Figure 25A), especially when t_M gets close to 0. This results in self-intersecting geometry. Guo et al. did not have this issue since gradient-based optimization of good initial designs was considered [36].

To counter this, we calculate t_L and t_R as portion of t_M, where they are at least mapped to $0.25t_M$, and maximally to $4t_M$. If we let $x_L \in [0, 1]$ denote the

decision variable controlling t_L, we calculate this thickness as follows:

$$t_L = [(4 - 0.25)x_L + 0.25]t_M \tag{42}$$

The same holds for the right thickness t_R.

Since we couple the variables like this, it is questionable whether this is the best solution. The solution is empirical, and the range somewhat arbitrary. However, it does ensure we do not generate invalid geometry. And we deem the range is sufficient to enough freedom to the optimizer, while not coming up with too extreme geometries. Another option is to check for this invalid geometry and create some sort of constraint or repair procedure for it.

To close off, the deformation can be defined on top of either the angular MMC representation $(x_0, y_0, \theta, l, t) + (t_L, t_R, a, b, c)$, or the endpoints representation $(x_1, y_1, x_2, y_2, t) + (t_L, t_R, a, b, c)$ in a modular fashion. Both gain five more shape-deformation parameters, resulting in a parameterization of n curved MMC beams having a search space dimension of $d = 10n$.

4 Experimental Setup

To limit the scope of the experiments we focus on three different dimensions: 10D, 20D, and 50D.

We consider the effect of three different parameterizations on the optimizer choice. As discussed in subsection 3.4 these are the: MMC, curved MMC, and Honeycomb Tiling parameterizations. We deem these to be diverse in enough in parameterization. One that strictly has straight beams (MMC, 5D per beam); one where we give beams more freedom (Curved MMC, 10D per beam); and a static tiling (Honeycomb, 1D per tile).

The three optimizers under consideration (as discussed in subsection 2.6 and subsection 2.7) are DE, CMA-ES, and HEBO. Although only three are selected, they represent a diverse range of optimization strategies: DE is a local search method driven by variation and selection within a population; CMA-ES adapts a probabilistic model of the population distribution; and HEBO builds a global surrogate model of the fitness landscape.

This should give enough variation to study the interplay between parameterization and optimizer for our TO case study.

4.1 Packages

For the calculation of the polygons the Shapely package is used, a package that facilitates many polygon operations [54]. These operations are used for transforming the geometry to the design domain, and for calculating inter-component distances used in the connectivity constraint calculation. For rasterization, the Rasterio package [56]. And, for calculating the MST, the SciPy package is used [55].

For all three methods, we use publicly available implementations: for DE, the SciPy package is used [55]; for CMA-ES, the pycma package maintained by the original author [57]; and for HEBO, the package accompanying the original paper [49].

The ELA features are calculated using a Python version of the original flacco R-package: named conveniently pflacco [58]. The proposed feature set that intend to use, contains one feature that is not available in this Python implementation, the authors of the original paper have used the original R-package [41]. This is the ela.meta.quad simple.coef.min by max feature. We therefore continue the ELA campaign without this feature. To find the PCA components of the ELA feature sets, the scikit-learn package is used [59].

The FEM simulation code is adapted from: github.com/BayesOptApp/Topology_Optimization, written by Elena Raponi and Iván Olarte Rodríguez.

4.2 Parameterization Settings

4.2.1 Honeycomb Tiling

For the Honeycomb Tiling the number of tiles equals the dimension. We generate the tilings as follows: for 10D we take two rows of tiles; for 20D three rows;

and for 50D four rows of tiles (Figure 27). In the 20D case this results in the removal of one tile, and in the 50D case the removal of two tiles. In all cases the outer columns of the tiling are at higher offset than the even columns. This is done to keep the tilings as consistent as possible. Each tile is attributed its own independent decision variable $x_j \in [0, 1]$, which is used to activate the tile if $x_j > 1/2$.

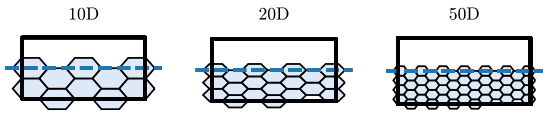


Figure 27: The proposed Honeycomb Tiling grids for the three different dimensions: 10D, 20D, and 50D.

4.2.2 MMC and Curved MMC

Regarding the beam dimensions, note that for comparison of parameterizations of the same dimension, Curved MMC designs contain half as many beams as regular MMC. Due to the imposed y-symmetry condition, the number of beams is effectively doubled, but only half are directly controlled by the optimizer. In effect, each control parameter influences a pair of mirrored beams simultaneously (see Figure 18). Table 3 lists the number of total beams for MMC and Curved MMC per dimension.

Dimension d	No. beams MMC with symmetry	No. beams Curved MMC with symmetry
10	4	2
20	8	4
50	20	10

Table 3: The total number of beams per dimension after mirroring of the geometry using the horizontal symmetry line.

We attribute different denormalization scales for the decision variables $\mathcal{X} \in [0, 1]^d$, for the angular parameterization (x_0, y_0, θ, l, t) and the endpoints parameterization (x_1, y_1, x_2, y_2, t).

In accordance to previous works, because the symmetry condition,the center of the beam (x_0, y_0) is confined to the lower half of the design domain. Still the beam bridge the full domain by placing its center on the symmetry boundary and extending its length (see Figure 20). Accordingly:

- x_0 is mapped from $[0, 1] \mapsto [0, d_x]$, where d_x is the domain width;
- y_0 from $[0, 1] \mapsto [0, d_y/2]$, where d_y is the domain height;

- θ from $[0,1] \mapsto [0,\pi]$; and
- l from $[0,1] \mapsto [0, \sqrt{d_x^2 + d_y^2}]$, which is the maximum possible beam length.

For the endpoints parameterization to represent the same range of geometries, the beam endpoints must be allowed in both halves of the domain. Therefore, for both endpoints:

- the x coordinates are mapped from $[0,1] \mapsto [0, d_x]$; and
- the y-coordinates from $[0,1] \mapsto [0, d_y]$.

For the thickness t, we use two different denormalization ranges. For initial exploratory experiments on the proposed endpoints representation and capsule shape, the following mapping is used:

$$[0,1] \mapsto \left[0, \frac{\sqrt{d_x^2 + d_y^2}}{n}\right],$$

where n denotes the number of beams.

In later experiments, this range is reduced to make it harder to fill the domain too easily. The adjusted mapping is:

$$[0,1] \mapsto \left[0, \min\left\{\frac{d_x}{n}, \frac{(d_y/2)}{n}\right\}\right]$$

This choice is based on the maximum thickness that would just fill the design domain if n beams were stacked vertically (d_x/n) or horizontally $((d_y/2)/n)$. We use $d_y/2$ to account for the symmetry conditions: filling the lower half automatically fills up the upper as well.

This denormalization ensures it is possible to fill-up the full design domain, while keeping the thickness range as small as possible. We make clear during the experiments which range is used.

For denormalization of the deformation parameters in the Curved MMC we stick to the proposed relative denormalization of t_L and t_R in terms of t (Equation 42). For the sine deformation parameters a, b, and c we set:

- a is mapped to the same range as t;
- b is mapped from $[0,1] \mapsto [-2\sqrt{d_x^2 + d_y^4}, 2\sqrt{d_x^2 + d_y^4}]$; and
- c is mapped from $[-\pi, \pi]$.

The mapping of b implies that the deformation can be more than one full wavecycle in both phase directions. We assume this range is reasonable to give the optimizer enough freedom while not resulting in too extreme beams with high frequency deformation.

Dimension d	Simulation budget N	DE population size	CMA-ES population size
10	200	150	10
20	400	300	12
50	1000	750	15

Table 4: Simulation budget and population sizes of DE and CMA-ES for corresponding parameterization dimensions.

4.3 Optimizer Settings

For all three optimizers, always the default settings are used, with exception to max iteration parameters. For DE and CMA-ES, the default population size is dependent on the dimension of the problem (Table 4). The surrogate model of HEBO on the other hand is updated after each iteration in sequential fashion. For DE the default mutation parameter is set at (0.5,1), meaning the differential weight F is sampled uniformly in this range for each mutation step (Equation 16). The initial population is sampled using LHS.

In CMA-ES the initial step-size is set at $\sigma^{(0)} = 0.25$, and the initial population mean is sampled uniformly on $[0, 1]^d$ (Equation 18). Although we activated the implemented restart procedure, restarting is never triggered in our experiments.

In default setup the HEBO uses a single AF: MACE, a variant of EI which separately weights the predicted mean and uncertainty to better balance exploration and exploitation. The Matérn 5/2 kernel is used in the Gaussian Process model that forms the surrogate.

For all optimizers we limit the budget of simulation calls to be: $N = 20 \cdot d$. Given that our used optimization methods are stochastic, all experiments are repeated 15 times using random states for reproducibility.

5 Experiments

5.1 MMC Representation

We use the 10D MMC problem, 4 beams, we experiment with the two different representations: the original angular- and the proposed endpoints representation of beams (subsubsection 3.4.2). Recall the rationale for the endpoints representation was to avoid the cyclic angle parameter θ, and to avoid the parameterization generating geometry outside of the design domain. Since CMA-ES is known to be a robust and easy to use out-of-the-box algorithm, only CMA-ES is considered in these experiments.

To investigate the angle-locking problem we run the problem twice with the angular representation:

- once where θ is mapped to $[0, \pi]$;
- and once by phase shifting this range with 90°: i.e., $[\pi/2, 3\pi/2]$.

This is done to inspect how of the choice denormalization range affects the procedure.

Both angular representations for the 10D case perform worse than the endpoints representation (Figure 28). However, it is clear that the phase-shifted representation performs much better than the regular range.

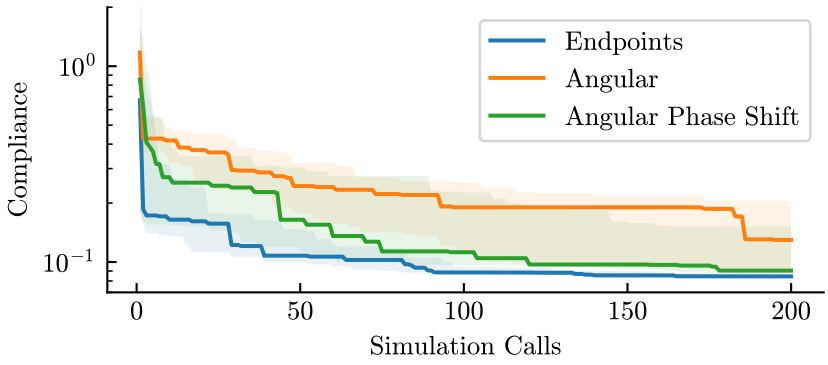


Figure 28: Median convergence curves of CMA-ES applied on the 10D MMC problem for different underlying parameterizations of MMCs. The shaded region represent the Interquartile Range.

Observing individual designs shows how as expected significant amounts of geometry are placed outside of the design domain in angular representations (Figure 29). This confirms the endpoints representation is better to describe geometry within the design domain.

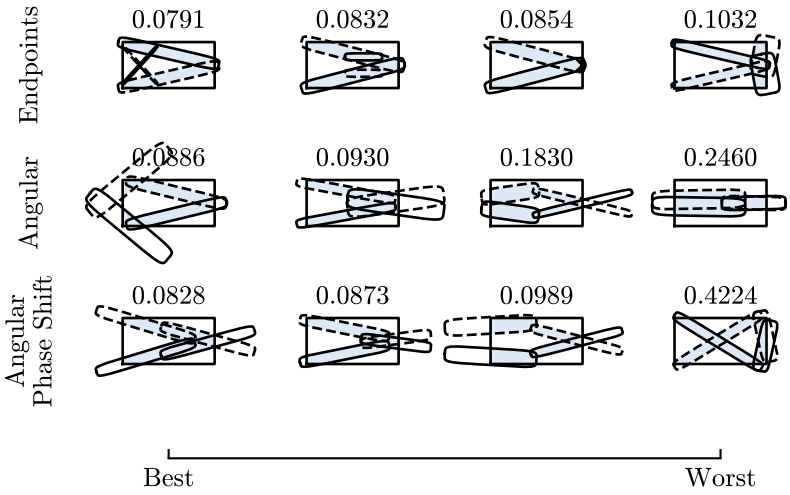


Figure 29: A selection of designs obtained on the 10D MMC problem for different underlying parameterizations of MMCs obtained with CMA-ES.

In the 10D problem, about 1/3 of designs using the angular representation include near-horizontal beams, which leads to worse compliance. However, this isn't necessarily true for higher dimensions: in the 15D case, the known optimum includes horizontal beams, and similar trends may hold for 20D or 50D problems, potentially aiding optimization.

The effect of the denormalization range has the following implication: the performance of a representation is biased to the problem. In the case where the problem is rotated 90°, such that the edge is on top and the load is applied on the bottom, the results of the regular angular- and phase-shifted representations would be swapped. This is all even though optimal topology stays the same; the problem is the same but rotated in world coordinates. The endpoints representation however is invariant to this rotation of the problem and does not rely on an arbitrary denormalization range.

For these reasons we continue for now using our proposed endpoints representation, nevertheless the angular representations could still be an interesting research topic for future research, especially in the context of other optimizers.

5.2 MMC Shape

We now consider the proposed Capsule shape (subsubsection 3.4.2), and compare it against the original Hyperellipse shape and a regular Rectangle shape. Again the CMA-ES method is used, now with endpoints parameterization. In

the 10D problem there was not found a difference in behavior, for thicker beams the above problem is less prevalent. Hence we move towards the 20D problem; the number of beams is doubled to eight.

From Figure 30 we note that there still seems to be not much difference among the curves and selecting a shape based on this would be more tuning the CMA-ES behavior on 20D MMC than making a objective decision. Although not presented here, the designs are all quite similar. However we observe slightly better results with the capsule shape (Figure 31).

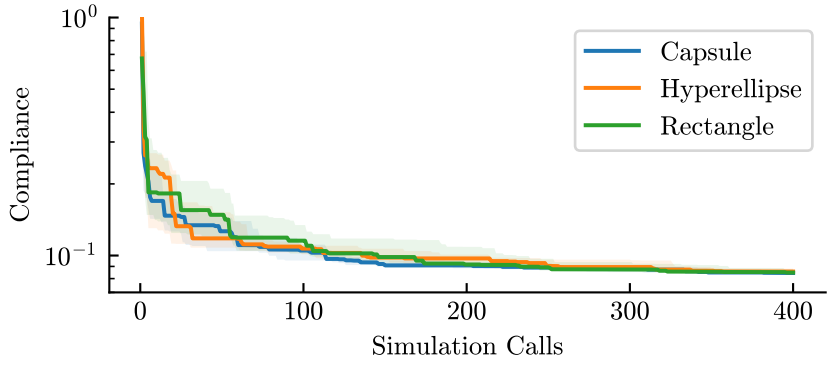


Figure 30: Median convergence curves of CMA-ES applied on the 20D MMC problem for different underlying shapes of the MMC beams. The shaded region represent the Interquartile Range.

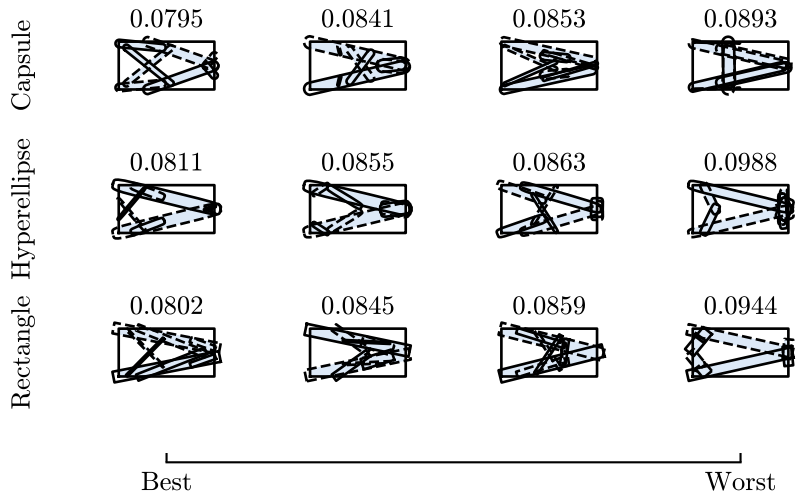


Figure 31: A selection of designs obtained on the 20D MMC problem for different underlying shapes of MMC beams obtained with CMA-ES.

Even though the slight difference, we do continue using the capsule shape for the simple fact that it produces beams that preserve their thickness. In the 10D/20D problems beams are not thin which mitigates this effect. Also, when adding deformation of the beams results in awkward corners. Again, it does not have to be a problem: it is after all the job of the optimizer to deal with these peculiarities. Nevertheless, capsules simply produces smoother shapes when deformed which is also a reason for continuing with the capsules.

5.3 Parameterization Optimizer Combinations

Now we consider all the proposed optimizers: DE, CMA-ES, and HEBO, in combination with all parameterizations: MMC, Curved MMC, and Honeycomb Tilings. In this experiment we choose the second denormalization factor for MMCs as described in subsubsection 4.2.2. For interested readers, all final designs are presented in the Appendix A.

5.3.1 10D Parameterizations

For 10D we observe how regular MMCs are the most competitive (Figure 32 and Figure 33). The Honeycomb Tiling is so simplistic that all optimizers find exactly the same configuration (Figure 34). In Figure 32 all are on the horizontal line below the legend, and all optimizers find this same design.

Analyzing the designs we find the optimizers seem to struggle slightly with a single curved beam (Appendix A). A straight V-shape is more optimal but often the optimizers find final designs of curved beams as shown in Figure 34. This is especially the case for the HEBO method.

Interestingly, MMC remains competitive across all optimizers in the 10-dimensional case (Figure 33), despite the fact that these optimizers are traditionally considered better suited to different types of landscapes. We observe in this 10D case study that devising a good parameterization gives us competitive results among all the optimizers.

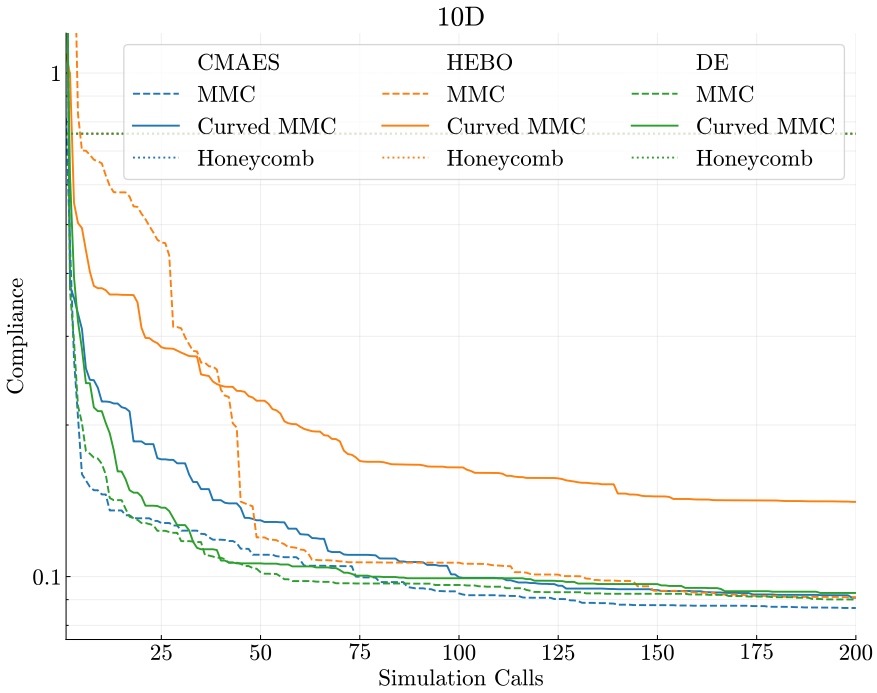


Figure 32: The average convergence curves over 15 runs for the 10D parameterizations.

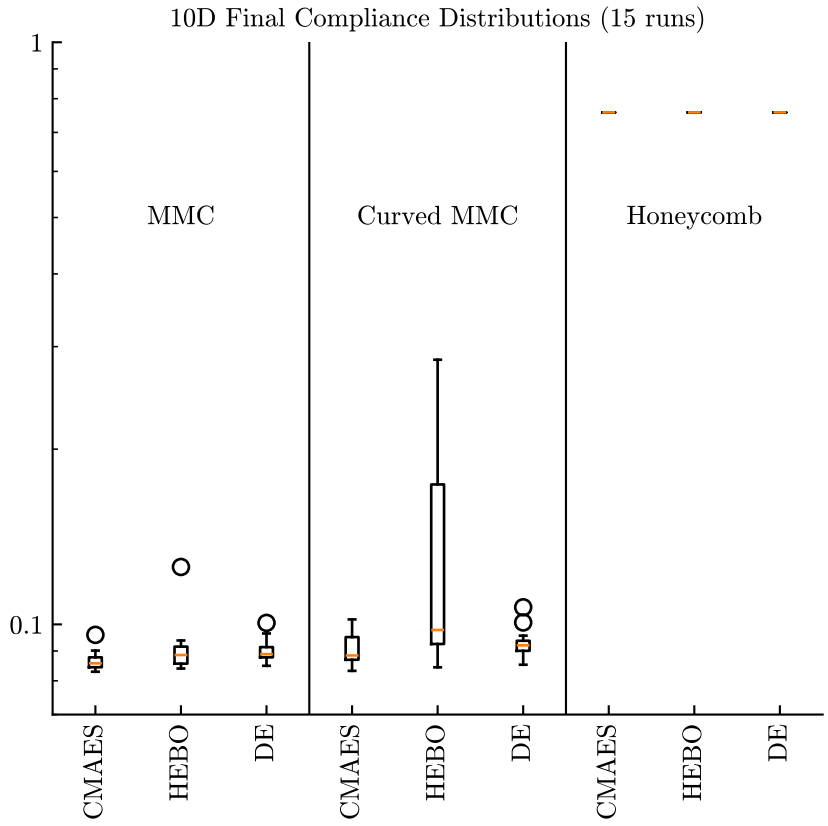


Figure 33: Final distributions of the 15 runs for the 10D parameterizations.

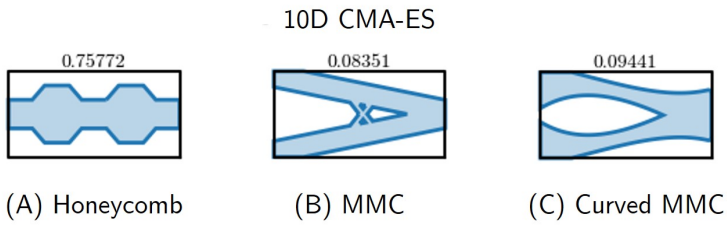


Figure 34: Some selected example designs obtained using the CMA-ES optimizer to show differences among the 10D parameterizations.

5.3.2 20D Parameterizations

For 20D (Figure 35 and Figure 36) we observe how dominance of the MMC parameterization starts to shift to the Curved MMC. Now also for Honeycomb parameterization we observe difference among the optimizers. Again to show difference among the parameterizations, Figure 37 shows the best obtained designs from CMA-ES.

Interestingly we now see difference in optimizer performance for the MMC parameterization while the Curved MMCs are much more robust across different optimizers, despite some outliers for HEBO (Figure 36). Given that regular MMCs have double the amounts of beams, it could mean the complexity of the design space starts to play a role here, as discussed with the number of symmetries (Equation 38). Again we note the following, the best parameterization gives us competitive performance across all our selected optimizers.

To add to symmetry problem, we also add that the effect of deformation parameters is much less than the regular MMC parameters. If changing the endpoints of the beam, it will be either be disconnected or alter connections significantly. The deformation parameters mostly alter the shape in between the endpoints of the beam. We deem that the number of beams is descriptive for the difficulty of the problem additional to the total dimension.

Also the Honeycomb parameterization is deliberately chosen as poor parameterization; given the static cells, it can represent much less diverse and complex designs compared to the MMC parameterizations. In such a poor parameterization we observe that the choice of optimizer affects the final performance the most (Figure 36). This leads to the observation that: optimizer performance is affected by quality of parameterization.

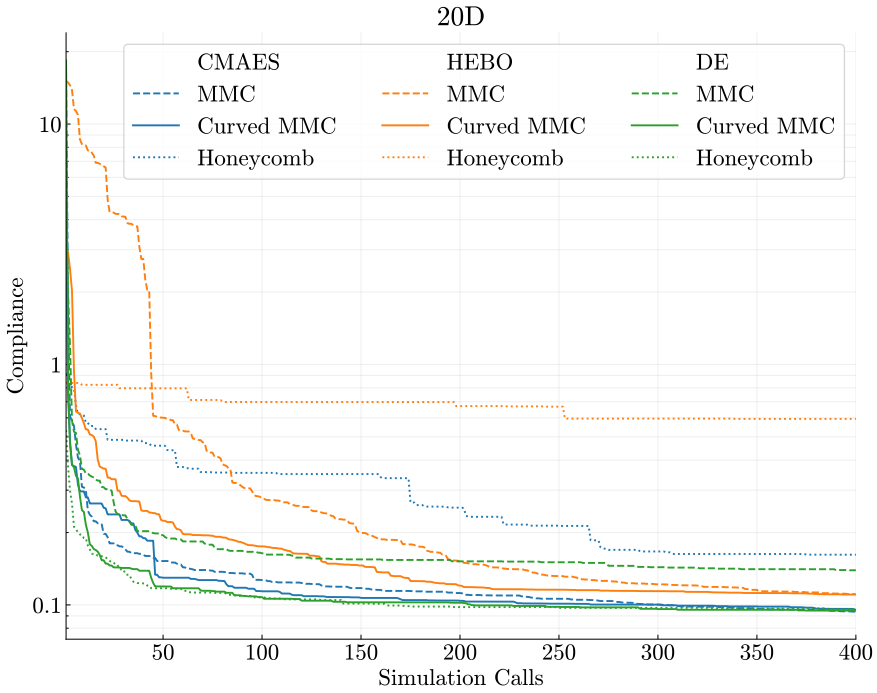


Figure 35: The average convergence curves over 15 runs for the 20D parameterizations

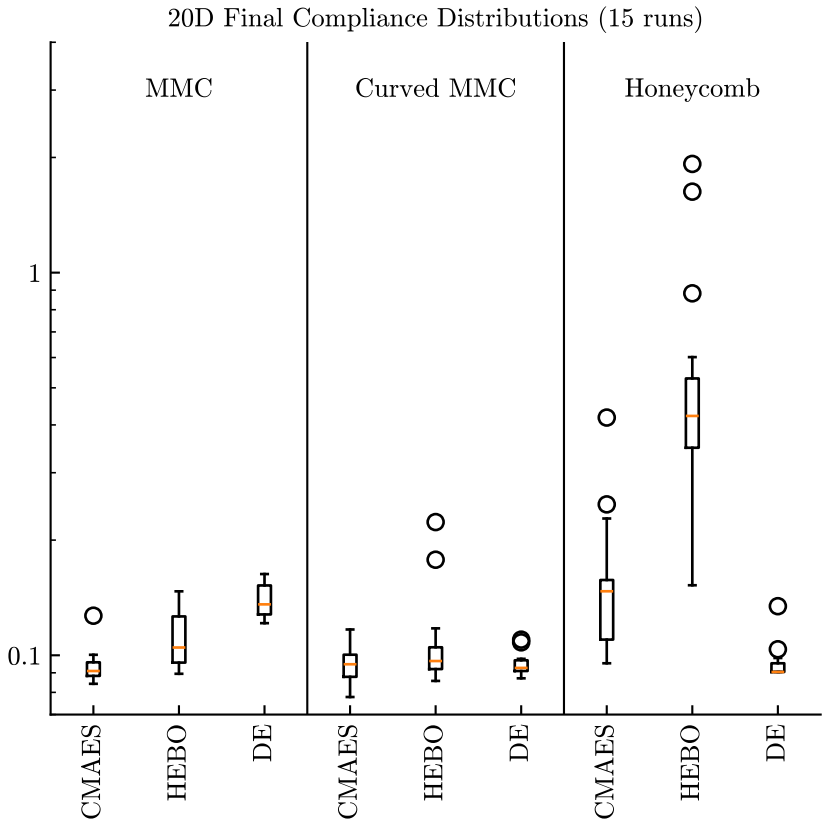


Figure 36: Final distributions of the 15 runs for the 20D parameterizations.

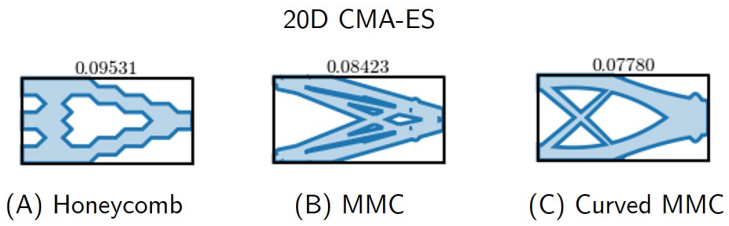


Figure 37: The best obtained designs using the CMA-ES optimizer to show differences among the 20D parameterizations.

5.3.3 50D Parameterizations

For 50D (Figure 38 and Figure 39) the trend continues, the added complexity from having a total of 20 beams in regular MMC in comparison to only 10 beams in Curved MMCs does not benefit the optimizers. For the Honeycomb Tiling we see the same behavior as in 20D, the ranking of the optimizers stays the same. Also for regular MMC the optimizer ranking stays the same but the performance gap starts to grow.

From Figure 40 we can see how complex this MMC parameterization becomes with this many beams.

Again from the final performance we observe the following: for the best parameterization, optimizer performance is competitive among all three. That is, even comparing a state-of-the-art optimizer like HEBO to DE.

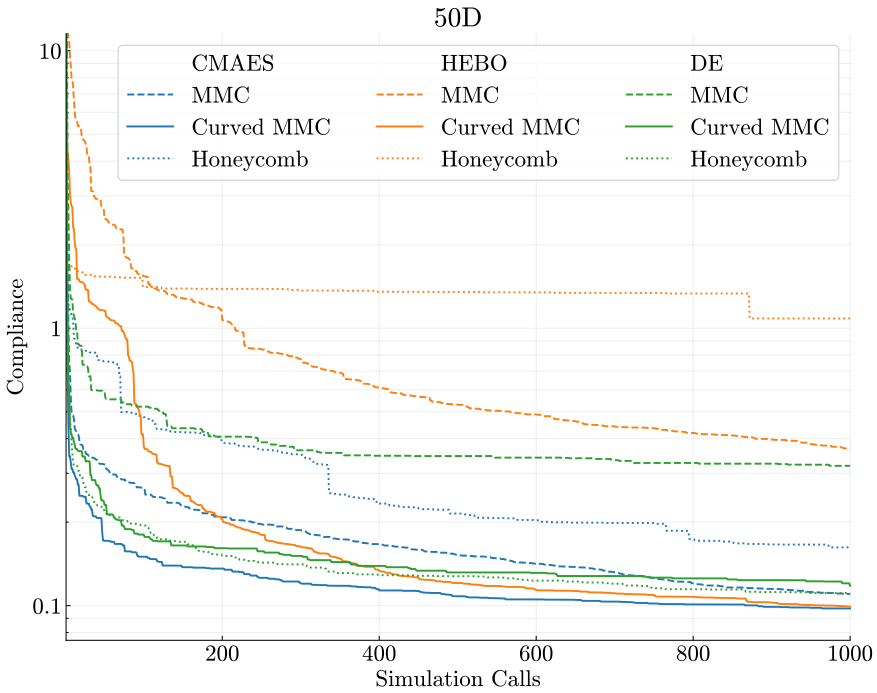


Figure 38: The average convergence curves over 15 runs for the 50D parameterizations

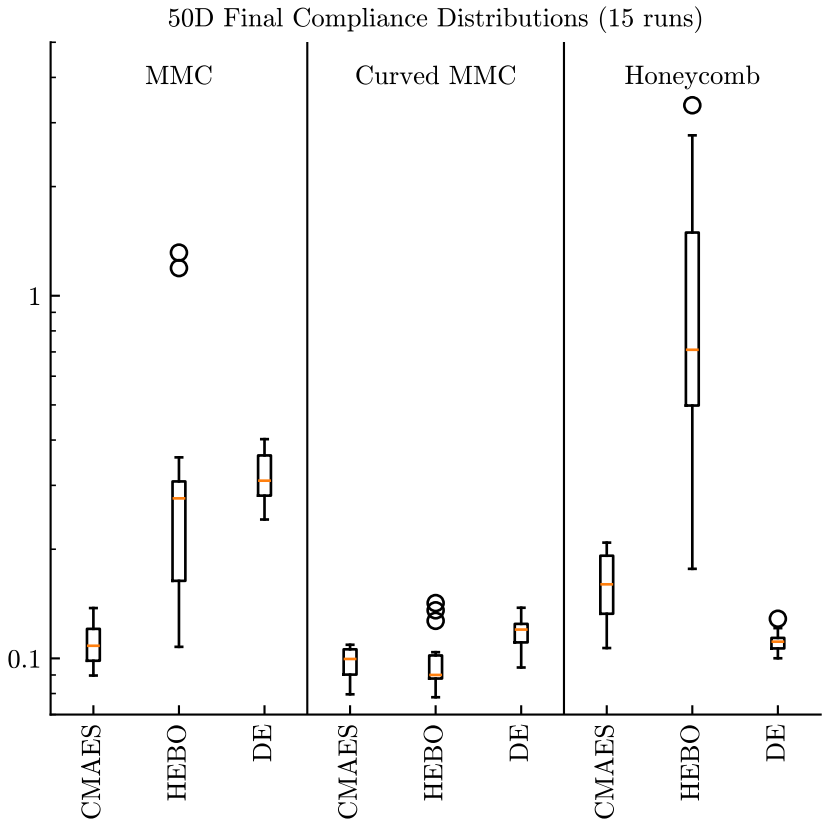


Figure 39: Final distributions of the 15 runs for the 50D parameterizations.

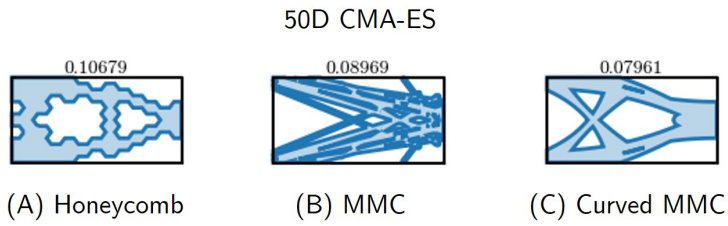


Figure 40: The best obtained designs using the CMA-ES optimizer to show differences among the 50D parameterizations.

5.3.4 Final Remarks

In only one run out of 405 no feasible design was found, this was for the 10D problem with Honeycomb parameterization.

Across all three problem dimensions, we find that designing a good parameterization is significantly more important than selecting a state-of-the-art optimizer. For poor parameterizations, such as the Honeycomb Tiling or MMCs with an increasing number of beams, even advanced optimizers are unable to compensate for the inherent limitations. In such cases, the choice of optimizer becomes critical to remain competitive. However, when a strong parameterization is used, all three tested algorithms perform competitively, regardless of the optimizer-parameterization combination.

In other words, the choice of parameterization has a larger impact on the outcome than the choice of optimizer in our case study. For practitioners, this implies that investing effort into designing or selecting an effective parameterization may yield greater returns than tuning or replacing the optimizer.

5.4 ELA

The philosophy of the BBOB functions is that they represent groups and characteristics commonly found in real-world BBO problems. This means that if we find correspondance in ELA feature space, we expect would similar optimizer performance. To investigate this we visualize the selected ELA features using two-component PCA, for all BBOB functions and our different TO parameterizations. For the ELA campaign a budget of $30 \cdot d$ is reserved, this is generally accepted as bare minimum. However, this already means $30 \cdot 50 = 1500$ samples are required calculating these features in the 50D case. For each BBOB function, the first function instances are repeated each five times, for a total of 25 feature sets per BBOB function. For the TO parameterizations 15 random seeds are considered.

We present all distributions for all BBOB and TO problems found in the Appendix B. From these distributions, we observe that none of the features exhibit values for TO that differ significantly from those observed on BBOB functions. Also, for the linear meta-model features, we observe that both the intercept and the maximum coefficient are exceptionally large only for BBOB function 12, compared to all other functions. As a result, these values dominate the PCA projection, causing all other functions to be compressed into a narrow region. Therefore, we exclude these features from the PCA analysis to avoid distortion.

Figure 41 shows the PCA projections for the 10D, 20D, and 50D problems. With increasing dimension, a increase in structure is observed. In all three our problem definitions are most similar to groups 4 and 5. According to the ELA hypothesis, this means these particular overlapping functions ideally should have the same optimizer behavior.

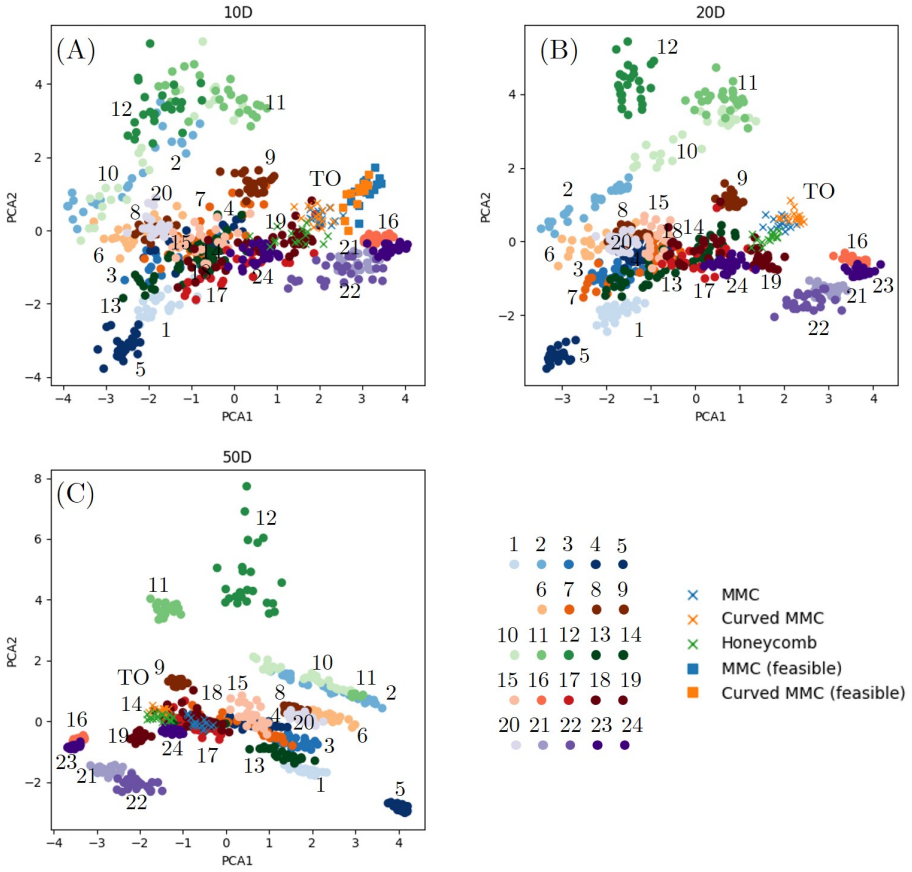


Figure 41: PCA projections of ELA features for BBOB functions and our TO parameterizations for: (A) 10D; (B) 20D; and (C) 50D.

Surprisingly, upon running the DE, CMA-ES, and HEBO optimizers on the BBOB functions we note the following: HEBO is always the best. Clearly, this is not the case in our results. Yet, our problems are projected in between BBOB functions. For this reason, we must immediately question representativeness of BBOB functions of real-world BBO problems.

Yet we note the following for 10D (Figure 41A), MMC and Curved MMC overlap quite well compared to Honeycomb Tiling. In our results for 10D we also find similar performance among MMC and Curved MMC, except for HEBO. For the Honeycomb Tiling the performance is far worse which could make sense given the separation in ELA space. Despite the difference all three map still to the same region.

For 20D (Figure 41B), the TO problems are now outside of the projection of the BBOB functions, although they are still close to some BBOB functions. We note the following: MMC is in between Honeycomb and Curved MMC. We

see this also in the quality of the parameterization from the 20D final performance box-plots (Figure 36). The curved MMC is least affected by optimizer selection, MMC slightly more, and in Honeycomb optimizer selection is critical. In correspondence to 10D this placement in ELA space could indicate quality of parameterization.

However, in the 50D case this does not hold anymore (Figure 41C). Now Curved MMCs and Honeycomb Tiling overlap while we saw complete different optimizer behavior of the two. Curved MMCs we found to be a good robust parameterization for which the optimizer selection is insignificant, whereas for Honeycomb Tilings is was found to be the complete opposite. Also surprisingly we find that f11 of BBOB gets mapped to two separate regions.

One might argue that the overlap is caused by the PCA projection being dominated by the BBOB functions. However, upon applying PCA only to the TO parameterizations, the same overlap is found.

We have to make the following observation: the hypervolume of the feasible space can be quite small compared to the infeasible. This means when sampling, likely all configration-objective pairs sampled to calculate the ELA features are from the infeasible space. For example in the 10D case, taking uniform samples from $[0, 1]^10$ results in an order of $\sim 0.1\%$ feasible designs for all three parameterizations. This decreases for the larger dimensions. This results in the ELA features being calculated only on the infeasible space: the constraint function. The question then becomes: what does this imply. Still this does not explain the difference among overlap in ELA space between the different dimensions.

Also when trying to connect the different overlaps to the evaluations it takes to find feasible space no connection can be made. For Curved MMCs the ranking is: HEBO, DE, CMA-ES, and for DE the ranking is CMA-ES, HEBO, DE. Appendix E shows the number of simulation calls in comparison to the total number of evaluations made to the problem.

For 10D we considered to calculate the ELA features solely on the feasible space, by discarding infeasible points and only keeping the feasible design vectors and corresponding objective responses. This results in around 300k total samples for 300 feasible. Due to the distribution of feasible sampled design vectors of the Honeycomb parameterization, the level-set and distribution features can not be calculated.

For both MMC and Curved MMC, we observe differences in most ELA features between the feasible and infeasible regions (see Appendix C for the box-plots of all features). However, these differences may reflect either genuine difference in the underlying landscapes, or artifacts introduced by differences in the sampling of design vectors across these regions. Since ELA is known to be particularly sensitive to such sampling variations [60], we consider the comparison between feasible and infeasible regions to be unreliable.

6 Discussion

6.1 On the Use of ELA for TO problems

Our experiments have highlighted several limitations when using ELA in the context of real-world constrained TO problems. While ELA features are originally selected to distinguish among well-studied BBOB benchmark functions, their descriptive power for problems like ours, particularly those with complex, constrained, and often discrete-like feasible regions, is questionable.

Possibly a dedicated feature set or a more rigorous analysis could benefit the descriptive task of ELA. For now, we found it difficult to say some thing useful about our problems based the selected feature set.

The most striking issue is the double landscape structure: the feasible and infeasible regions of the design space may exhibit fundamentally different landscapes, both in structure and in difficulty. In our 10D analysis, we attempted to isolate the feasible landscape by computing ELA features solely from feasible samples. However, with feasible regions comprising less than 0.1% of the design space, this approach produced highly biased and sparse estimates, especially for the Honeycomb parameterization, where even basic ELA features such as level-set and distribution measures became incalculable.

It remains a question on how to handle problems such as these in the ELA framework where only feasible calls count towards the budget.

Further, since ELA is known to be highly sensitive to the distribution of the sampled points [60], any comparison between feasible and infeasible landscapes, especially when based on unequally sampled regions, must be interpreted with extreme caution. As such, we consider our current ELA approach insufficient for making robust conclusions about constrained TO parameterizations.

From reasoning about sampling the infeasible space we can argue that little useful can be said about the full optimization performance. Consider the following thought experiment about a 2D landscape. There is a thin ring containing one of the BBOB functions. Outside the ring, the landscape funnels towards the ring, such that the optimizer is guided into this "feasible" region. ELA features from samples outside the ring can never say what happens inside the ring. Since any function can be inside the ring, the optimizer performance can be never known from only sampling the infeasible.

6.2 On the Importance of Parameterization

In subsection 5.3 we have shown the importance of the parameterization in comparison to the optimizer for our horizontal cantilever case-study. A question that remains is how well this transfers to other problem domains. That is, different design domains with different boundary conditions. But also, problems with different objectives, other than the minimization of compliance. An interesting direction could be to look again into the vehicle chrashworthiness problem discussed in previous work [24, 35].

Moreover, as our parameterizations are tightly coupled to the denormalization ranges, future work could explore how sensitive performance is to these ranges. We found this while experimenting with some ranges in the Curved MMC parameterizations. It could be this denormalization range is as critical to the parameterization as the geometric description.

Also, it would be nice to extend the simulation budget. We had trouble getting HEBO towards 5000 simulation calls given the overhead of the surrogate model. However, we found designs representing the optimal design from SIMP (Figure 13B) can be obtained reliably with CMA-ES by extending the optimization run (Appendix F).

7 Conclusion

This thesis investigated the interplay between parameterization and optimizer performance in constrained TO problems. The case-study in question is a horizontal cantilever problem for which compliance is minimized. Through a systematic case study involving three parameterizations: MMC, Curved MMC, and Honeycomb Tiling; across three problem dimensions (10D, 20D, and 50D); and using three optimizers (CMA-ES, DE, and HEBO), we examined how selection of parameterization and optimizer influence solution quality.

First we proposed a method for constraining infeasible designs using a connectivity constraint. This constraint relies on finding the minimum distance required to connect the design. Although we did not benchmark our proposed constraint, it was found it does guide all three different optimizers to feasible designs. In one single case: the 10D Honeycomb grid, in one run, HEBO converged to an infeasible design. For all other experiments, feasible designs were always found using all three different optimizers. Therefore we conclude the proposed connectivity constraint is effective for finding connected designs, across our case-study. Additionally, the proposed endpoint-based parameterization of MMC was found to consistently keep the geometry within the design domain, in contrast to the original angular representation.

Our results show that parameterization quality has a more significant impact on optimization outcomes than the choice of optimizer. Poorly constructed parameterizations, such as the Honeycomb Tiling or MMC with many beams, were found to lead to large difference among optimizer performance. While for a good parameterization all three optimizers were found to be competitive. We found this to be the case in all three selected parameterization dimensions.

For our case study this indicates that the parameterization itself seems to enable robust performance across different optimization strategies. While poor parameterization leads to a reliance of optimizer selection or tuning for competitive performance.

Additionally, we explored the application of ELA in this context. While ELA has shown value in benchmarking standard BBO problems, like BBOB, we found it to be of limited use in constrained TO problems. The sensitivity of ELA features to sampling distributions, combined with the sparsity of the feasible regions, undermines the reliability of many feature calculations. Solely sampling the infeasible region can never infer performance in the feasible, because the performance of the feasible depends on the objective and not only on the constraints. This suggests a need for constraint-aware or domain-specific landscape analysis methods to properly characterize problems like ours. We have not found way to get a better understanding of our parameterization through the use of ELA features.

In this thesis, we observed that for our specific case study involving a constrained, real-world topology optimization (TO) problem, the choice of parameterization had a more significant impact on performance than the choice of optimizer. The optimizer's effectiveness appeared to be limited by the structure and expressiveness of the design space defined by the parameterization.

While this finding is based on a single case study, it suggests that thoughtful, task-specific parameterization design may be a worthwhile focus before investing heavily in optimizer selection or tuning.

7.1 Future Work

First, for future work we would suggest to extend the study by more optimizers and parameterizations. Although we have found strong evidence with three different optimizers, broader selection would help confirm the generality of our conclusions and potentially reveal interactions between specific optimizers and parameterization strategies that were not captured in this study. Also, as denormalization range is a significant part of the parameterization, we would suggest to assess its sensitivity in comparison to our presented results.

It would further interesting to see how our observations hold up in other problem domains. That is, different structural problems besides the static horizontal cantilever, to completely other problem domains such as the vehicle chrashwortiness problem (which initiated this study) [24, 35].

Lastly, we recommend future work on developing constraint-aware ELA methods tailored to TO. Existing ELA feature sets, largely based on BBOB, may not capture the dual landscape structure of TO problems, where only feasible solutions count toward the budget. We suggest exploring new features or sampling strategies that account for feasibility, or applying representation learning to model the feasible region directly, enabling ELA to operate more meaningfully in constrained, real-world design spaces.

This work highlights the importance of thoughtful parameterization design in real-world TO problems and underscores the need for constraint-aware landscape analysis methods.

References

[1] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[3] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, Cambridge, UK, January 2022.

[4] Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. *Finite element method*. John Wiley & Sons, 2012.

[5] Philippe R Spalart and V Venkatakrishnan. On the role and challenges of cfd in the aerospace industry. *The Aeronautical Journal*, 120(1223):209–232, 2016.

[6] Craig Buchanan and Leroy Gardner. Metal 3d printing in construction: A review of methods, research, applications, opportunities and challenges. *Engineering Structures*, 180:332–348, 2019.

[7] Barry Berman. 3-d printing: The new industrial revolution. *Business horizons*, 55(2):155–162, 2012.

[8] Dan William Martinez, Michaela T Espino, Honelly Mae Cascolan, Jan Lloyd Crisostomo, and John Ryan C Dizon. A comprehensive review on the application of 3d printing in the aerospace industry. *Key engineering materials*, 913:27–34, 2022.

[9] Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

[10] Stephane Grihon, Lars Krog, and Klaus Hertel. A380 weight savings using numerical structural optimization. In *20th AAAF colloquium on material for aerospace applications, Paris, France*, pages 763–766, 2004.

[11] Lars Krog, Alastair Tucker, Martin Kemp, and Richard Boyd. Topology optimisation of aircraft wing box ribs. In *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 4481, 2004.

[12] Martin Philip Bendsøe and Noboru Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer methods in applied mechanics and engineering*, 71(2):197–224, 1988.

[13] Behrooz Hassani and Ernest Hinton. A review of homogenization and topology optimization i—homogenization theory for media with periodic structure. *Computers & Structures*, 69(6):707–717, 1998.

[14] Tristan Briard, Frédéric Segonds, and Nicolo Zamariola. G-dfam: a methodological proposal of generative design for additive manufacturing in the automotive industry. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 14(3):875–886, 2020.

[15] Martin P Bendsøe. Optimal shape design as a material distribution problem. *Structural optimization*, 1:193–202, 1989.

[16] Krister Svanberg. The method of moving asymptotes—a new method for structural optimization. *International journal for numerical methods in engineering*, 24(2):359–373, 1987.

[17] Y Mike Xie and Grant P Steven. Basic evolutionary structural optimization. In *Evolutionary structural optimization*, pages 12–29. Springer, 1997.

[18] Osvaldo M Querin, Grant P Steven, and Yi Min Xie. Evolutionary structural optimisation (eso) using a bidirectional algorithm. *Engineering computations*, 15(8):1031–1048, 1998.

[19] Michael Yu Wang, Xiaoming Wang, and Dongming Guo. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1-2):227–246, 2003.

[20] Grégoire Allaire, François Jouve, and Anca-Maria Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1):363–393, 2004.

[21] Xu Guo, Weisheng Zhang, and Wenliang Zhong. Doing topology optimization explicitly and geometrically—a new moving morphable components based framework. *Journal of Applied Mechanics*, 81(8):081009, 2014.

[22] Ole Sigmund and Kurt Maute. Topology optimization approaches: A comparative review. *Structural and multidisciplinary optimization*, 48(6):1031–1055, 2013.

[23] Jonathan Mirpourian and Niels Aage. Arbitrary boundary conditions in topology optimization: Applications in strongly coupled viscothermal multiphysics. 2025.

[24] Mariusz Bujny, Nikola Aulig, Markus Olhofer, and Fabian Duddeck. Identification of optimal topologies for crashworthiness with the evolutionary level set method. *International Journal of Crashworthiness*, 23(4):395–416, 2018.

[25] Eric Sandgren, E Jensen, et al. Topological design of structural components using genetic optimization method. 1990.

[26] Colin D Chapman, Kazuhiro Saitou, and Mark John Jakiela. Genetic algorithms as an approach to configuration and topology design. *Journal of mechanical design*, 116(4):1005–1012, 1994.

[27] Ole Sigmund. On the usefulness of non-gradient approaches in topology optimization. *Structural and Multidisciplinary Optimization*, 43(5):589–596, 2011.

[28] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 829–836, 2011.

[29] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[30] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 2002.

[31] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*. PhD thesis, INRIA, 2009.

[32] Nico P Van Dijk, Kurt Maute, Matthijs Langelaar, and Fred Van Keulen. Level-set methods for structural topology optimization: a review. *Structural and Multidisciplinary Optimization*, 48(3):437–472, 2013.

[33] Rajat Saxena and Anupam Saxena. On honeycomb parameterization for topology optimization of compliant mechanisms. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 37009, pages 975–985, 2003.

[34] Mariusz Bujny, Nikola Aulig, Markus Olhofer, and Fabian Duddeck. Evolutionary level set method for crashworthiness topology optimization. In *ECCOMAS Congress*, 2016.

[35] Elena Raponi, Mariusz Bujny, Markus Olhofer, Nikola Aulig, Simonetta Boria, and Fabian Duddeck. Kriging-assisted topology optimization of crash structures. *Computer Methods in Applied Mechanics and Engineering*, 348:730–752, 2019.

[36] Xu Guo, Weisheng Zhang, Jian Zhang, and Jie Yuan. Explicit structural topology optimization based on moving morphable components (mmc) with curved skeletons. *Computer methods in applied mechanics and engineering*, 310:711–748, 2016.

[37] Mike Preuss. *Multimodal optimization by means of evolutionary algorithms*. Springer, 2015.

[38] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical report, Citeseer, 2010.

[39] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[40] RICHARD BELLMAN. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[41] Konstantin Dietrich and Olaf Mersmann. Increasing the diversity of benchmark function sets through affine recombination. In *International Conference on Parallel Problem Solving from Nature*, pages 590–602. Springer, 2022.

[42] Quentin Renau, Johann Dréo, Carola Doerr, and Benjamin Doerr. Towards explainable exploratory landscape analysis: extreme feature selection for classifying bbob functions. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 17–33. Springer, 2021.

[43] Mario A Muñoz, Michael Kirley, and Saman K Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE transactions on evolutionary computation*, 19(1):74–87, 2014.

[44] Monte Lunacek and Darrell Whitley. The dispersion metric and the cma evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 477–484, 2006.

[45] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[46] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[47] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[48] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[49] Alexander I Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, et al. Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349, 2022.

[50] Gokhan Serhat and Ipek Basdogan. Lamination parameter interpolation method for design of manufacturable variable-stiffness composite panels. *Aiaa Journal*, 57(7):3052–3065, 2019.

[51] Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S Lazarov, and Ole Sigmund. Efficient topology optimization in matlab using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16, 2011.

[52] Mariusz Bujny. *Level set topology optimization for crashworthiness using evolutionary algorithms and machine learning*. PhD thesis, Technische Universität München, 2020.

[53] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[54] Sean Gillies, Casper van der Wel, Joris Van den Bossche, Mike W. Taves, Joshua Arnott, Brendan C. Ward, et al. Shapely, August 2024.

[55] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[56] Sean Gillies et al. Rasterio: geospatial raster i/o for Python programmers, 2013–.

[57] Nikolaus Hansen, yoshihikoueno, ARF1, Sait Cakmak, Gabi Kadlecová, Guillermo Abad López, Kento Nozawa, Luca Rolshoven, Youhei Akimoto, brieglhostis, and Dimo Brockhoff. Cma-es/pycma: v4.3.0, July 2025.

[58] Raphael Patrick Prager and Heike Trautmann. Pflacco: Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems in Python. *Evolutionary Computation*, pages 1–25, 07 2023.

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[60] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In *International Conference on Parallel Problem Solving from Nature*, pages 139–153. Springer, 2020.

A Final Designs of Parameterization-Optimizer Combinations

Figures below show all the final designs obtained in subsection 5.3. They are sorted based on objective score.

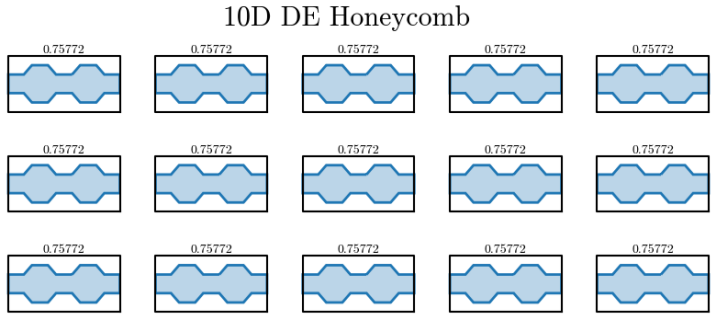


Figure 42: Final designs obtained with DE on the 10D Honeycomb parameterization (budget 200).

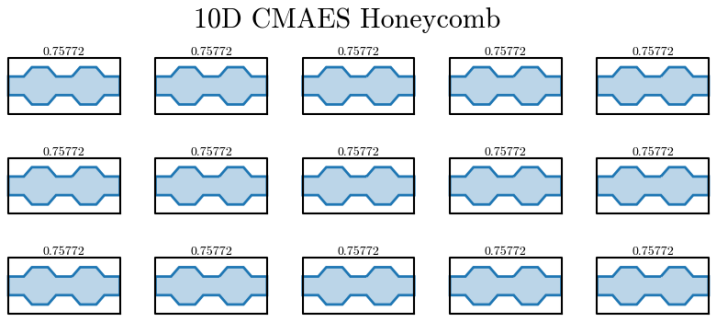


Figure 43: Final designs obtained with CMAES on the 10D Honeycomb parameterization (budget 200).

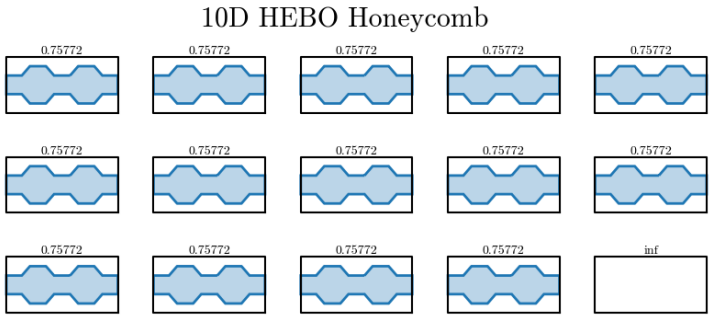


Figure 44: Final designs obtained with HEBO on the 10D Honeycomb parameterization (budget 200).

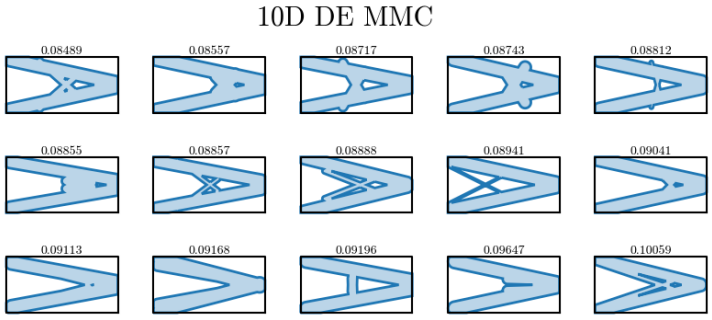


Figure 45: Final designs obtained with DE on the 10D MMC parameterization (budget 200).

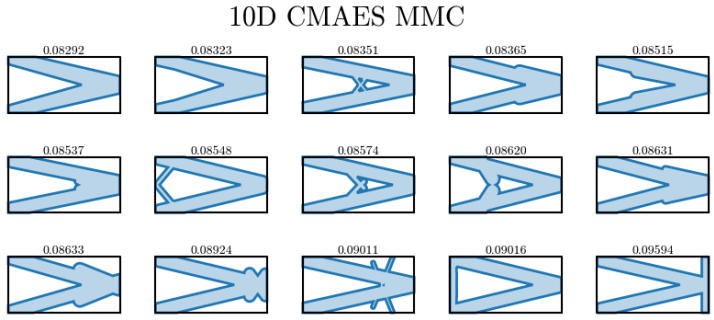


Figure 46: Final designs obtained with CMAES on the 10D MMC parameterization (budget 200).

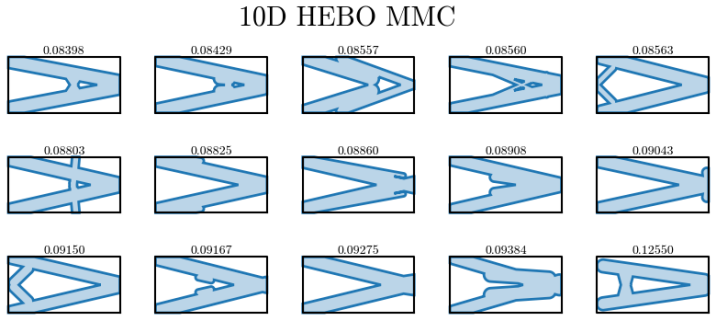


Figure 47: Final designs obtained with HEBO on the 10D MMC parameterization (budget 200).

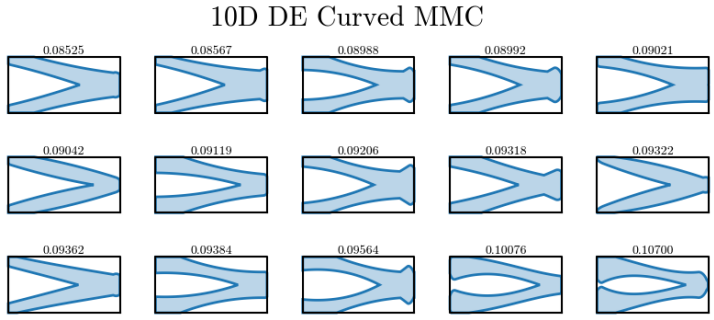


Figure 48: Final designs obtained with DE on the 10D Curved MMC parameterization (budget 200).

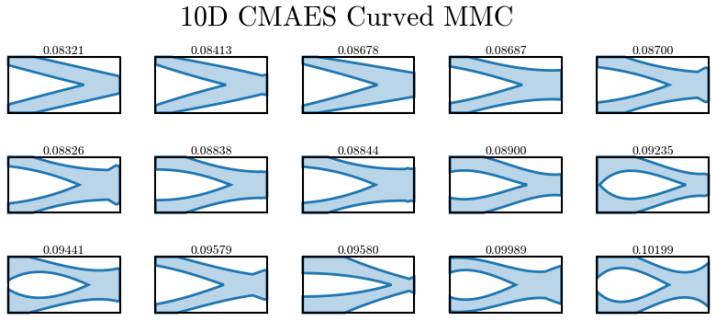


Figure 49: Final designs obtained with CMAES on the 10D Curved MMC parameterization (budget 200).

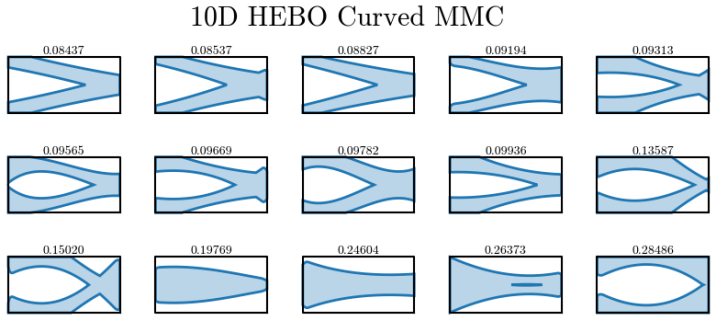


Figure 50: Final designs obtained with HEBO on the 10D Curved MMC parameterization (budget 200).

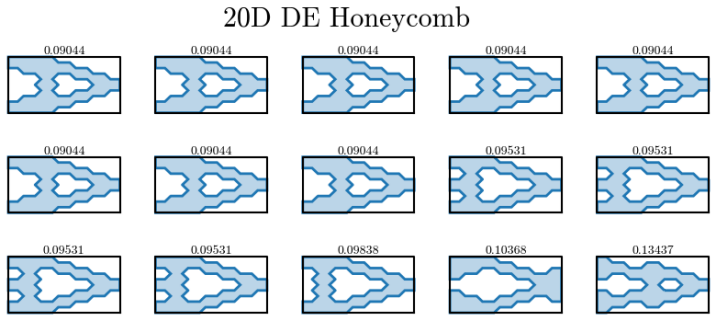


Figure 51: Final designs obtained with DE on the 20D Honeycomb parameterization (budget 400).

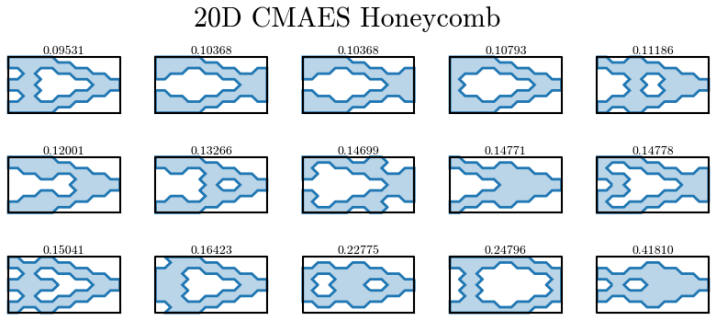


Figure 52: Final designs obtained with CMAES on the 20D Honeycomb parameterization (budget 400).

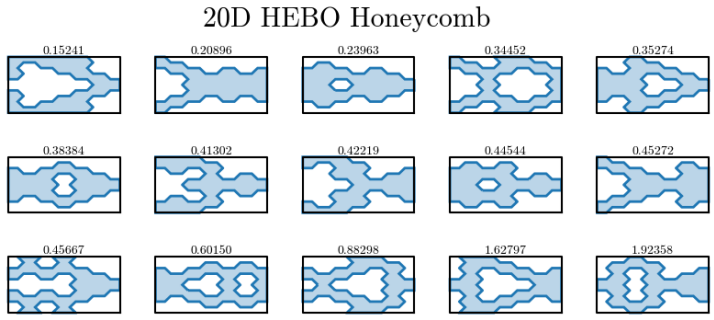


Figure 53: Final designs obtained with HEBO on the 20D Honeycomb parameterization (budget 400).

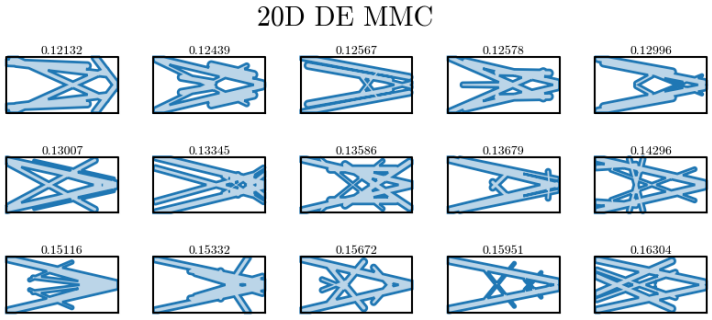


Figure 54: Final designs obtained with DE on the 20D MMC parameterization (budget 400).

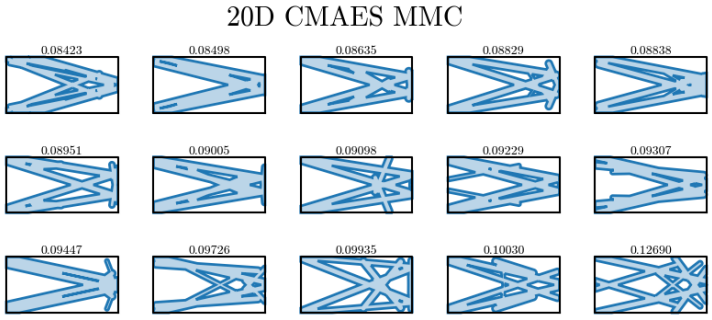


Figure 55: Final designs obtained with CMAES on the 20D MMC parameterization (budget 400).

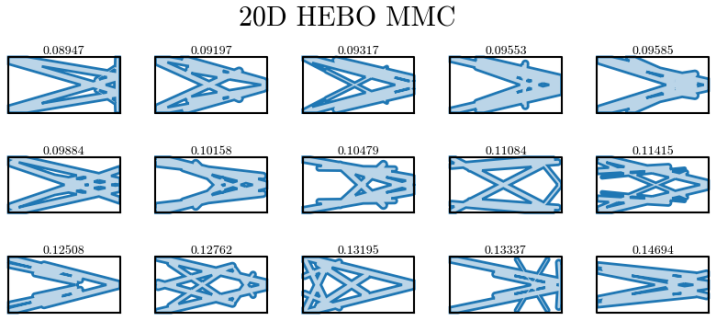


Figure 56: Final designs obtained with HEBO on the 20D MMC parameterization (budget 400).

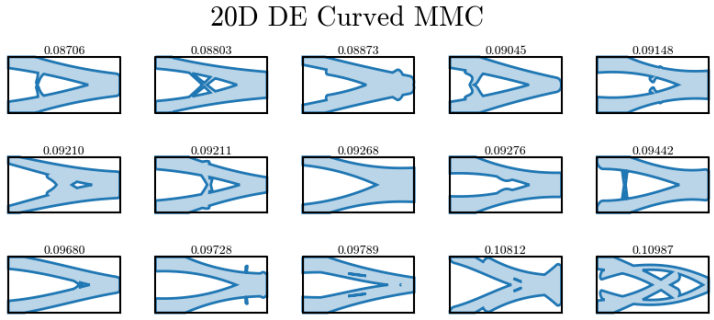


Figure 57: Final designs obtained with DE on the 20D Curved MMC parameterization (budget 400).

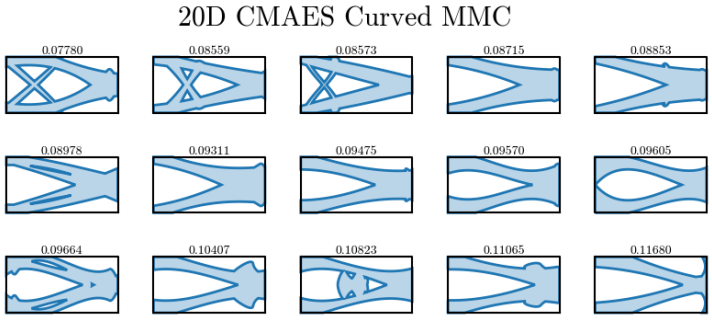


Figure 58: Final designs obtained with CMAES on the 20D Curved MMC parameterization (budget 400).

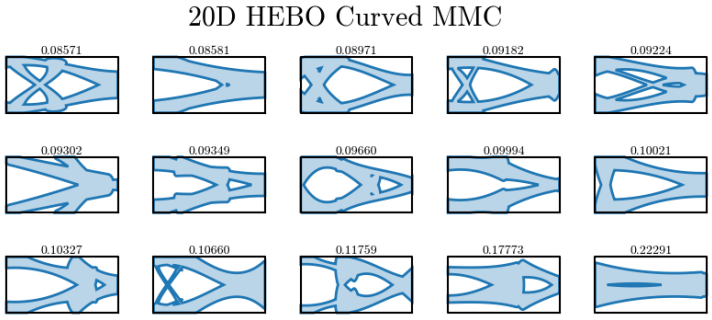


Figure 59: Final designs obtained with HEBO on the 20D Curved MMC parameterization (budget 400).

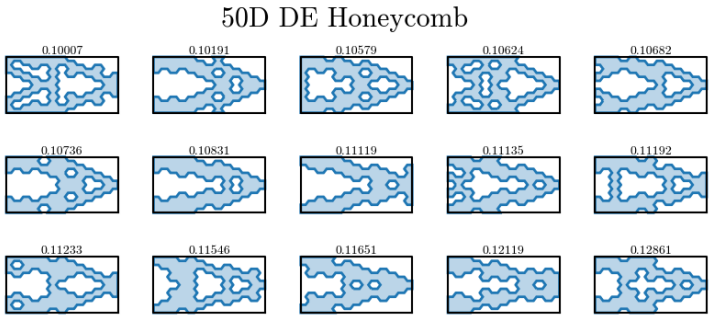


Figure 60: Final designs obtained with DE on the 50D Honeycomb parameterization (budget 1000).

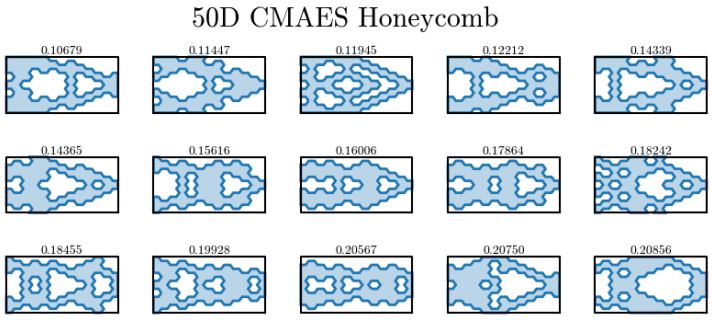


Figure 61: Final designs obtained with CMAES on the 50D Honeycomb parameterization (budget 1000).

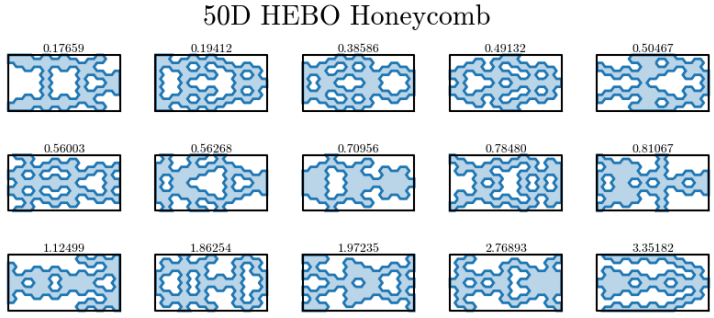


Figure 62: Final designs obtained with HEBO on the 50D Honeycomb parameterization (budget 1000).

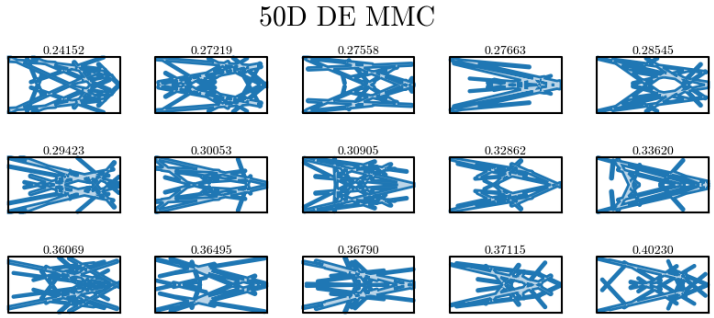


Figure 63: Final designs obtained with DE on the 50D MMC parameterization (budget 1000).

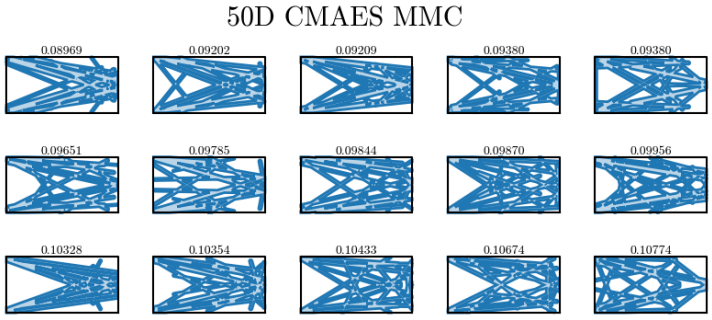


Figure 64: Final designs obtained with CMAES on the 50D MMC parameterization (budget 1000).

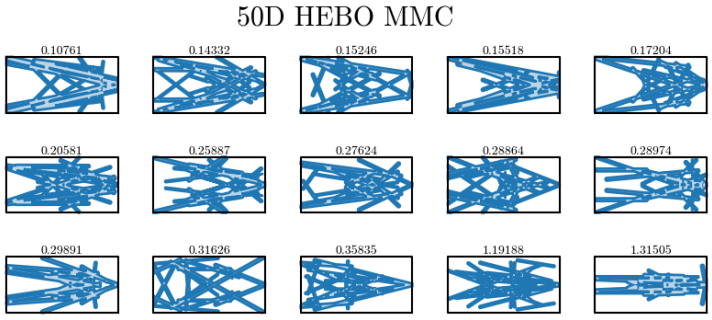


Figure 65: Final designs obtained with HEBO on the 50D MMC parameterization (budget 1000).

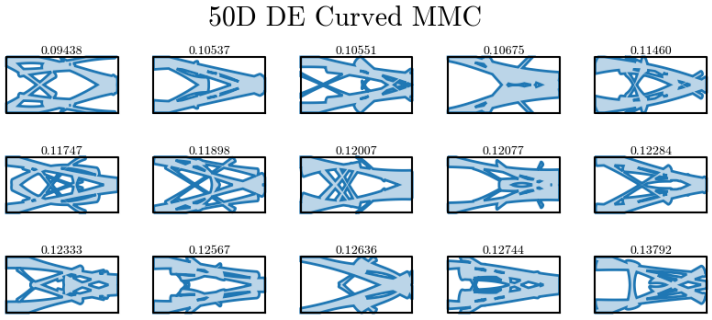


Figure 66: Final designs obtained with DE on the 50D Curved MMC parameterization (budget 1000).

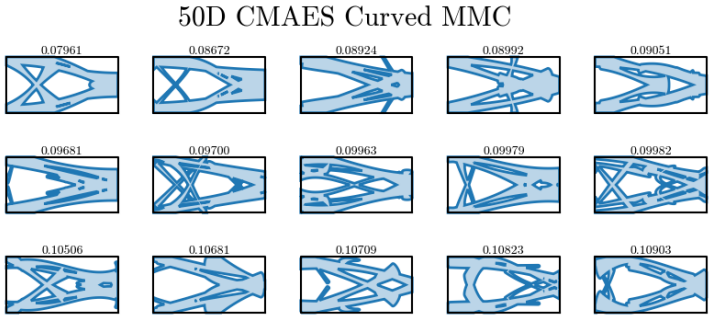


Figure 67: Final designs obtained with CMAES on the 50D Curved MMC parameterization (budget 1000).

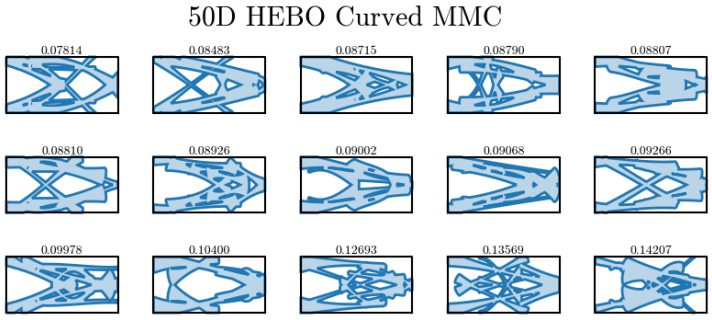


Figure 68: Final designs obtained with HEBO on the 50D Curved MMC parameterization (budget 1000).

B ELA Features BBOB and TO

Figure 69, Figure 70, and Figure 71 show the distributions of all the calculated ELA features are displayed. The numbers on the x-axis represent the BBOB functions. Numbers 25, 26, and 27 are our TO parameterizations. With 25: MMC, 26: Curved MMC, and 27: Honeycomb. These are highlighted in the shaded region. Per problem we show three distributions, these correspond to the three dimensions: 10D, 20D, and 50D. Besides these we also show the distributions for the feasible samples beyond the shaded region which were only obtained for 10D: MMC and Curved MMC. A dedicated comparison between the infeasible and feasible features is given in the next section.

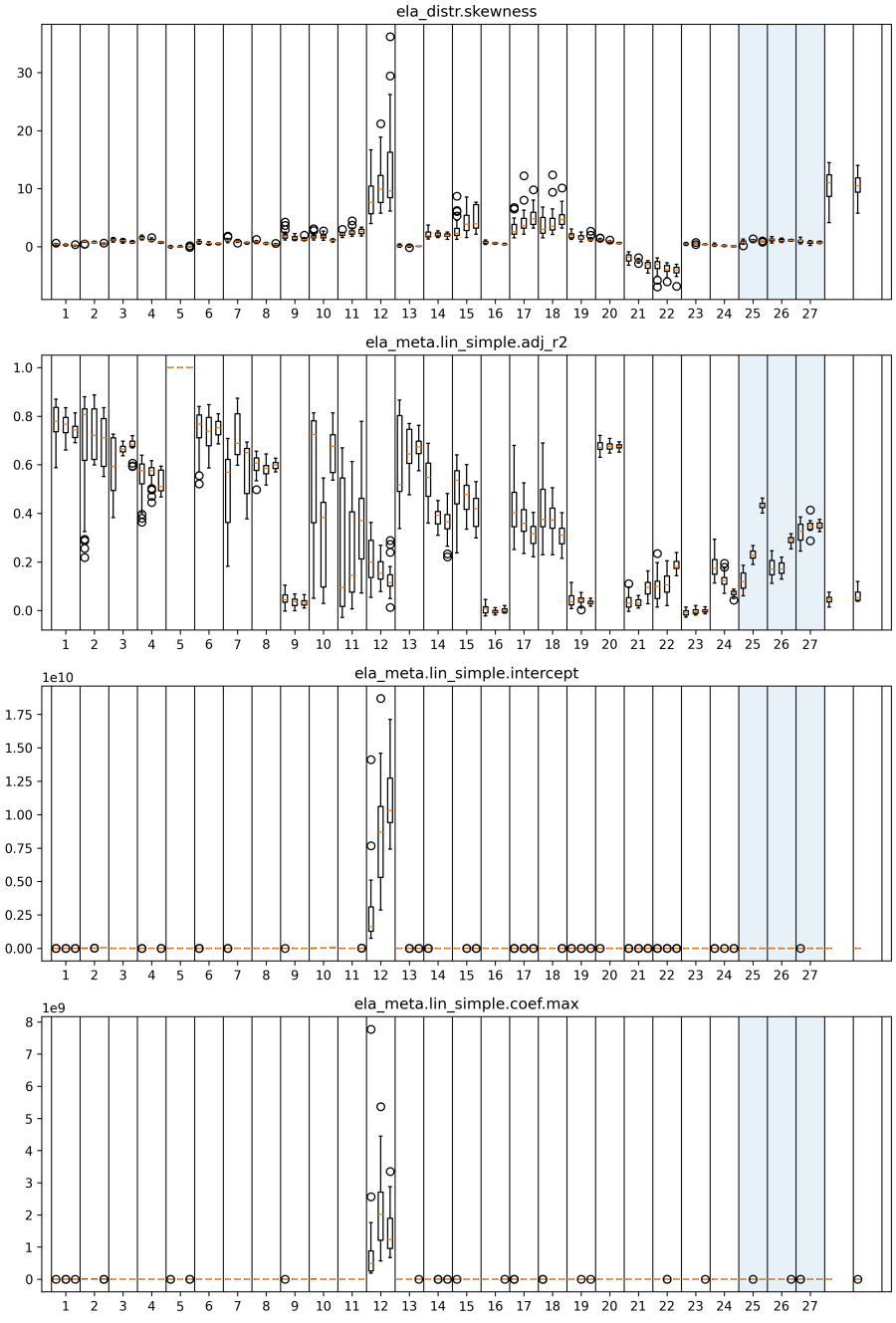


Figure 69: A selection of the ELA feature comparison between the BBOB functions and TO parameterizations (shaded) for dimensions: 10D, 20D, and 50D (1/3).

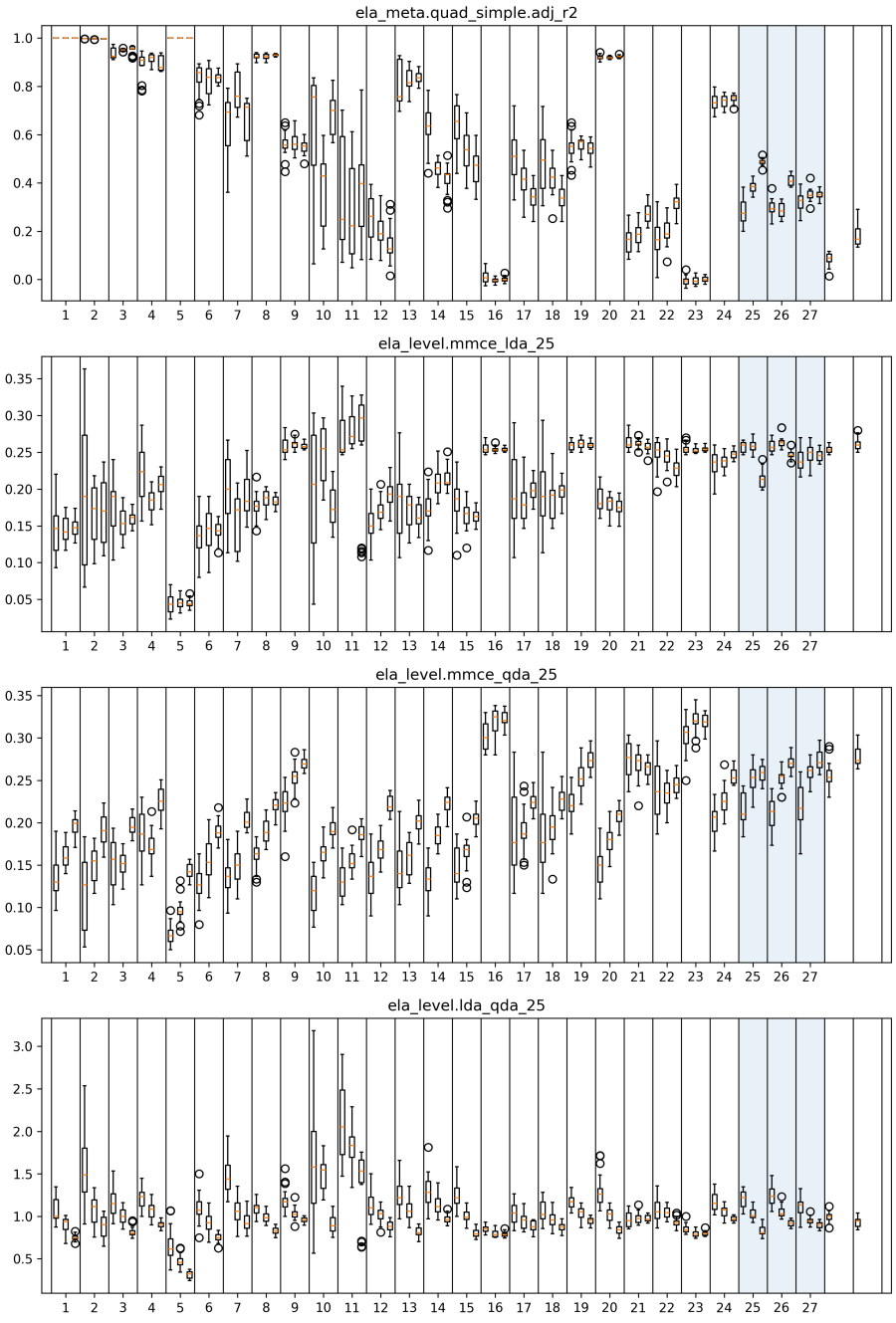


Figure 70: A selection of the ELA feature comparison between the BBOB functions and TO parameterizations (shaded) for dimensions: 10D, 20D, and 50D (2/3).

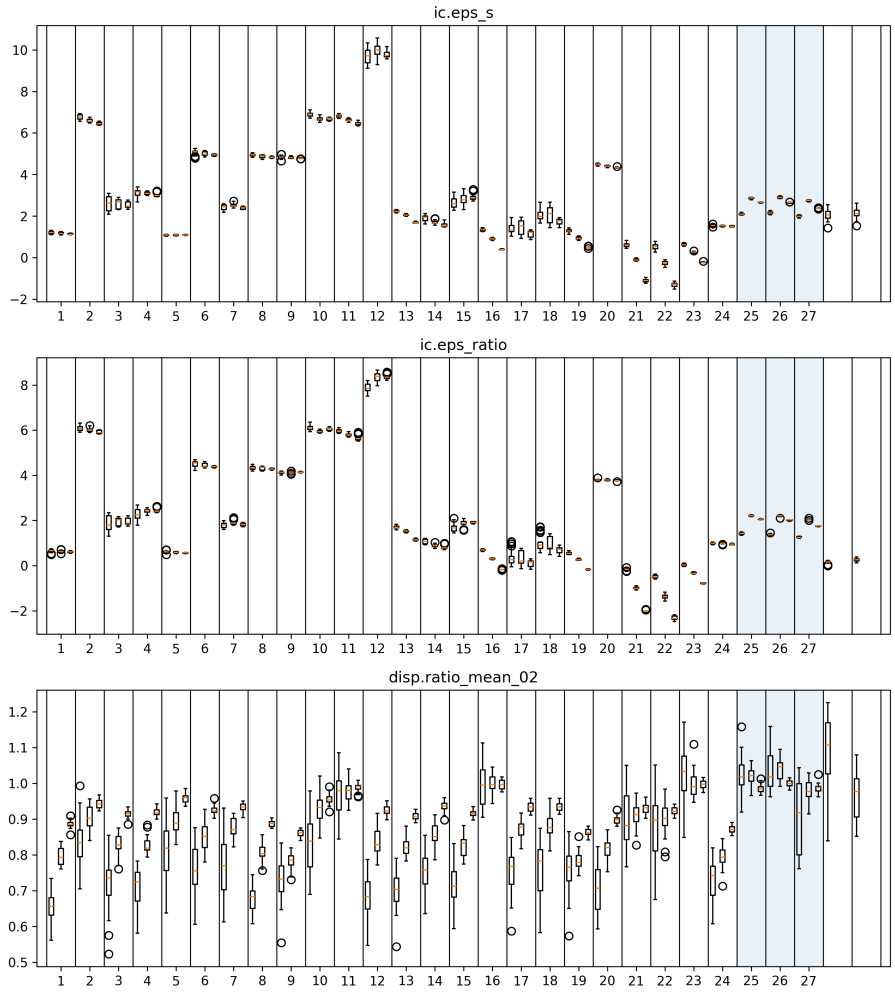


Figure 71: A selection of the ELA feature comparison between the BBOB functions and TO parameterizations (shaded) for dimensions: 10D, 20D, and 50D (3/3).

C ELA Features TO for Infeasible and Feasible

Figure 72, Figure 73, and Figure 74, show the comparisons of the feasible sample and infeasible sample of ELA features for our TO problems.

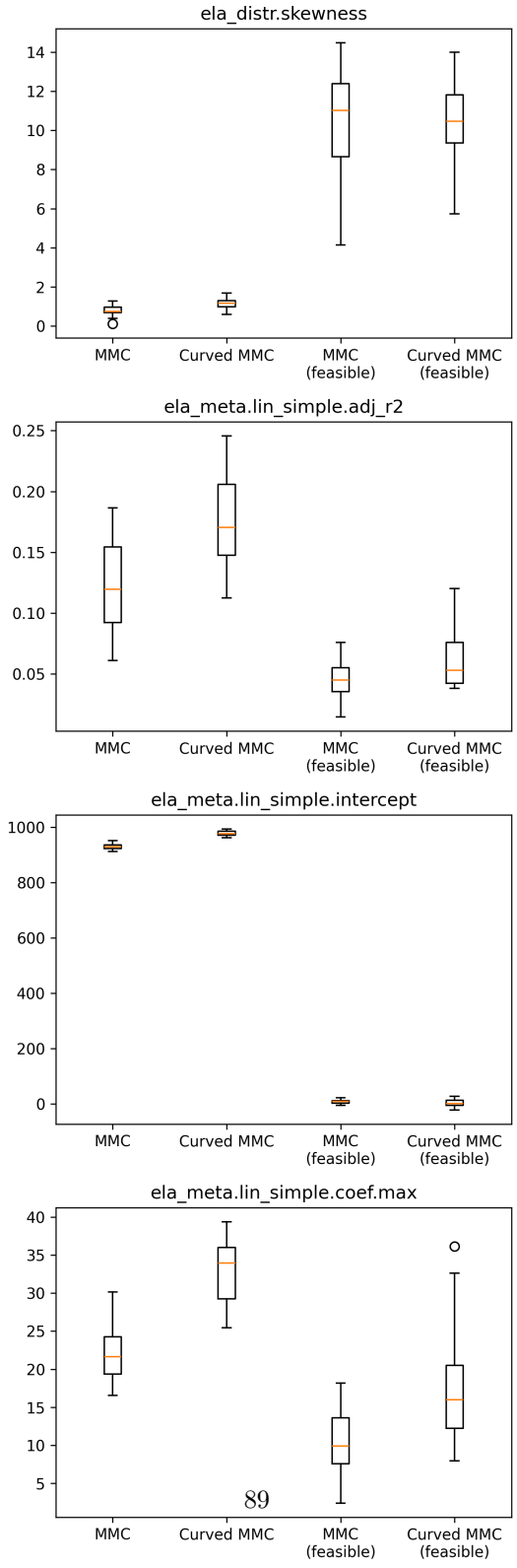


Figure 72: A selection of the ELA feature comparison between the feasible- and infeasible TO samples on 10D (1/3).

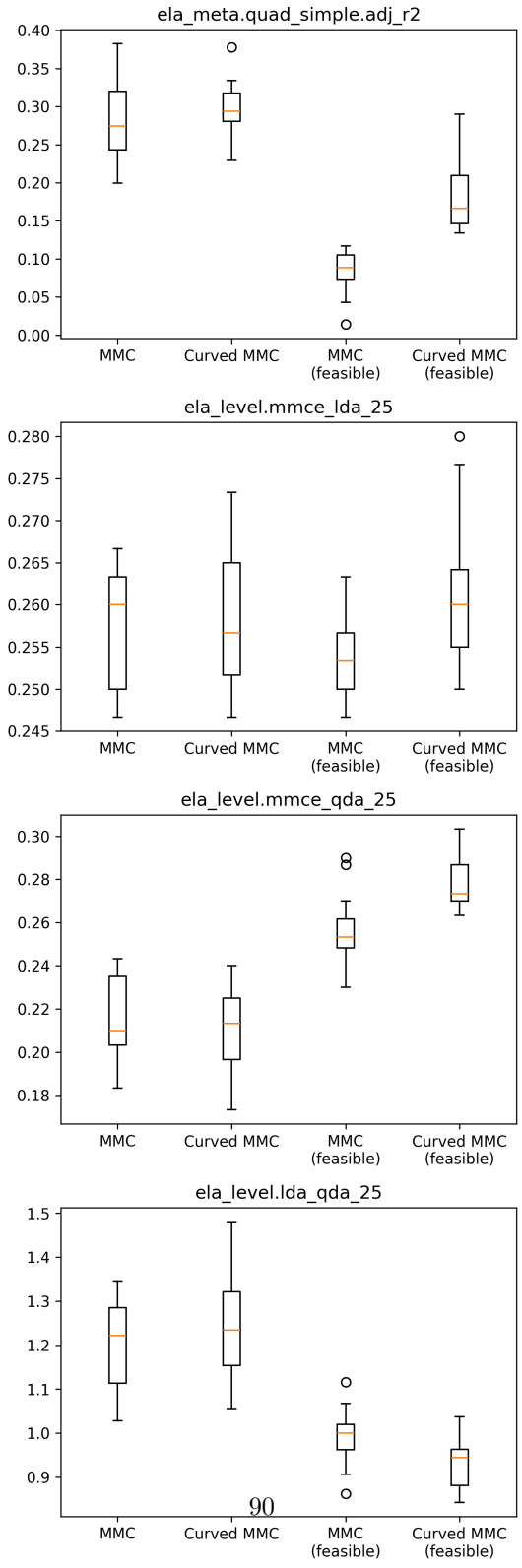


Figure 73: A selection of the ELA feature comparison between the feasible- and infeasible TO samples on 10D (2/3).

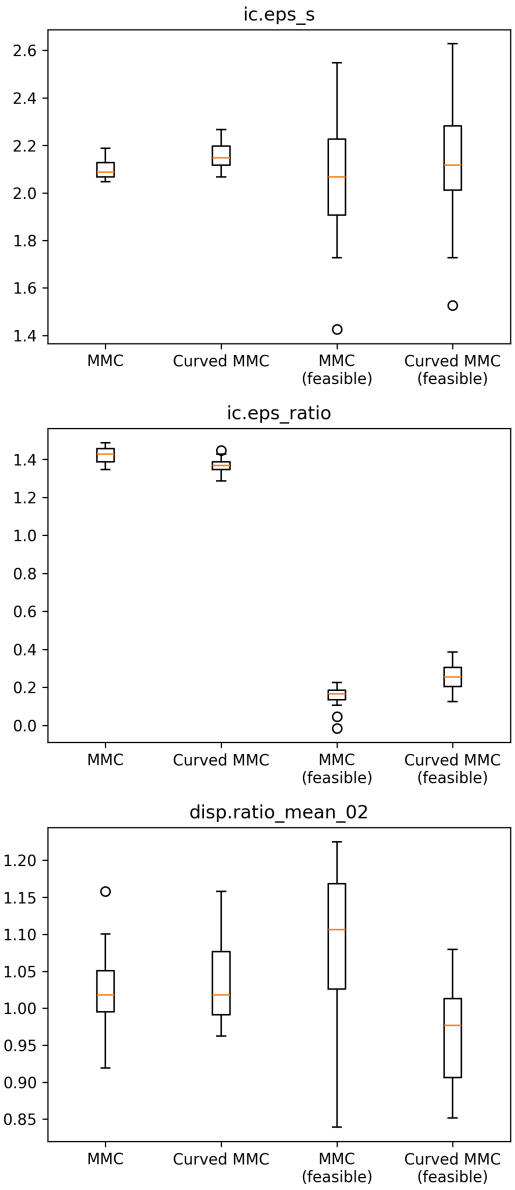


Figure 74: A selection of the ELA feature comparison between the feasible- and infeasible TO samples on 10D (3/3).

D BBOB Convergence Curves

We show the convergence curves for the 10D BBOB functions (Figure 75), 20D BBOB functions (Figure 76), and the 50D BBOB functions (Figure 77), for the optimizers: CMA-ES, HEBO, and DE.

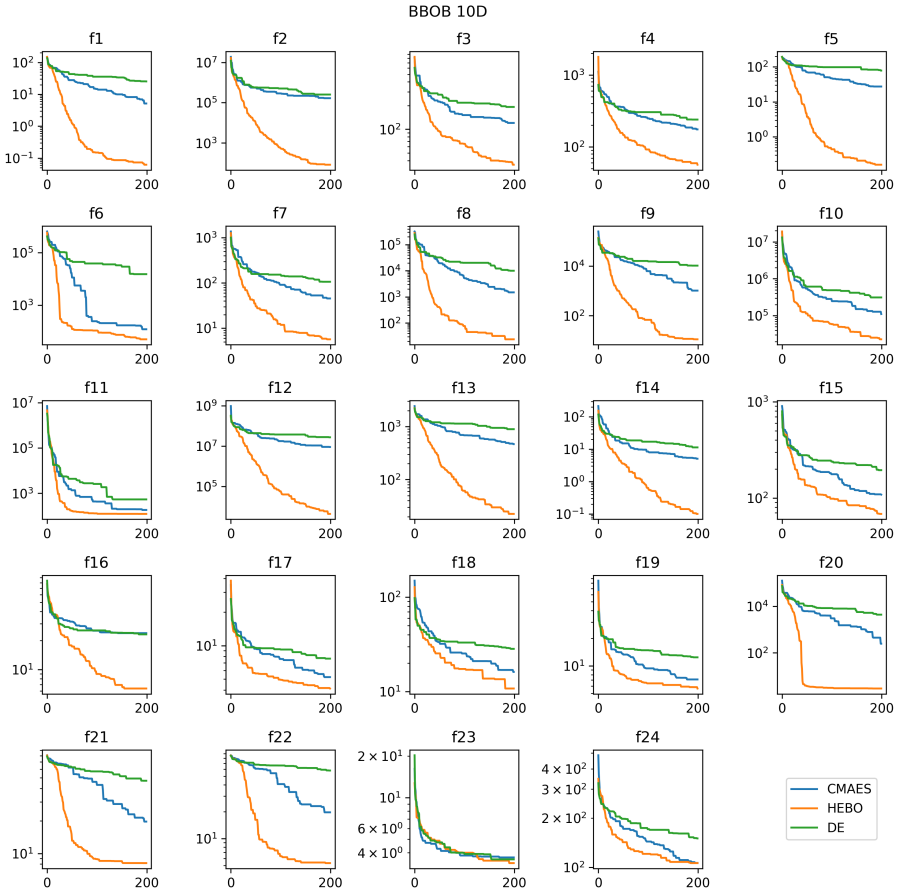


Figure 75: BBOB convergence curves for the 10D problems, for optimizers: CMA-ES, HEBO, and DE.

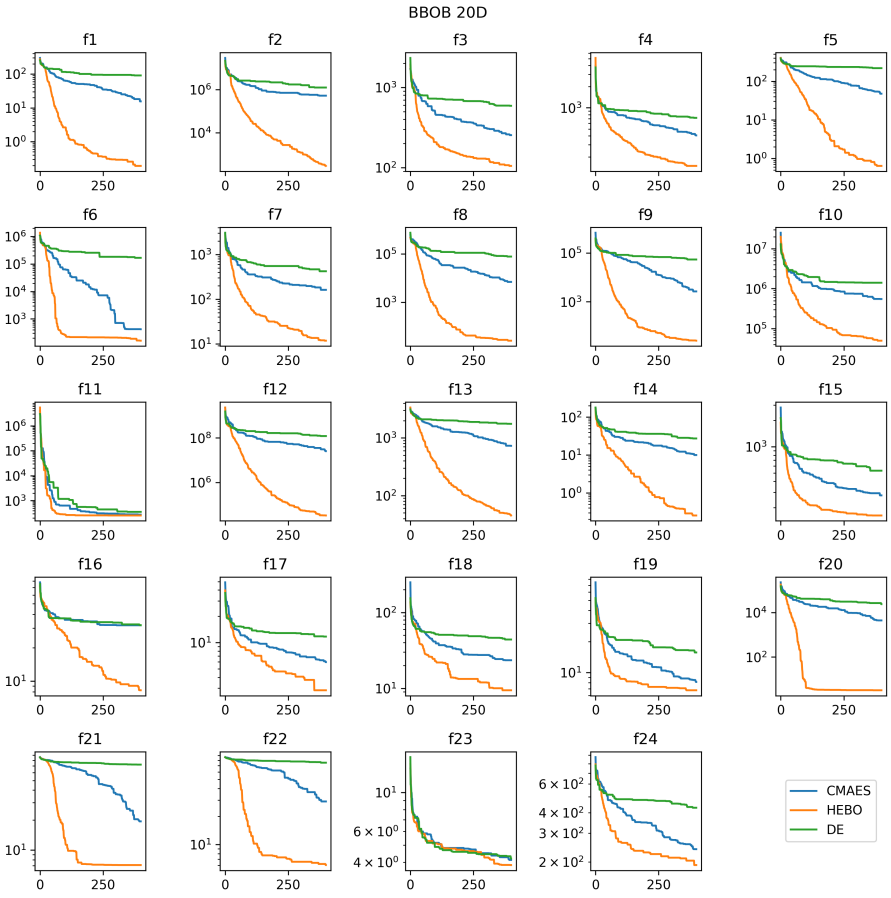


Figure 76: BBOB convergence curves for the 20D problems, for optimizers: CMA-ES, HEBO, and DE.

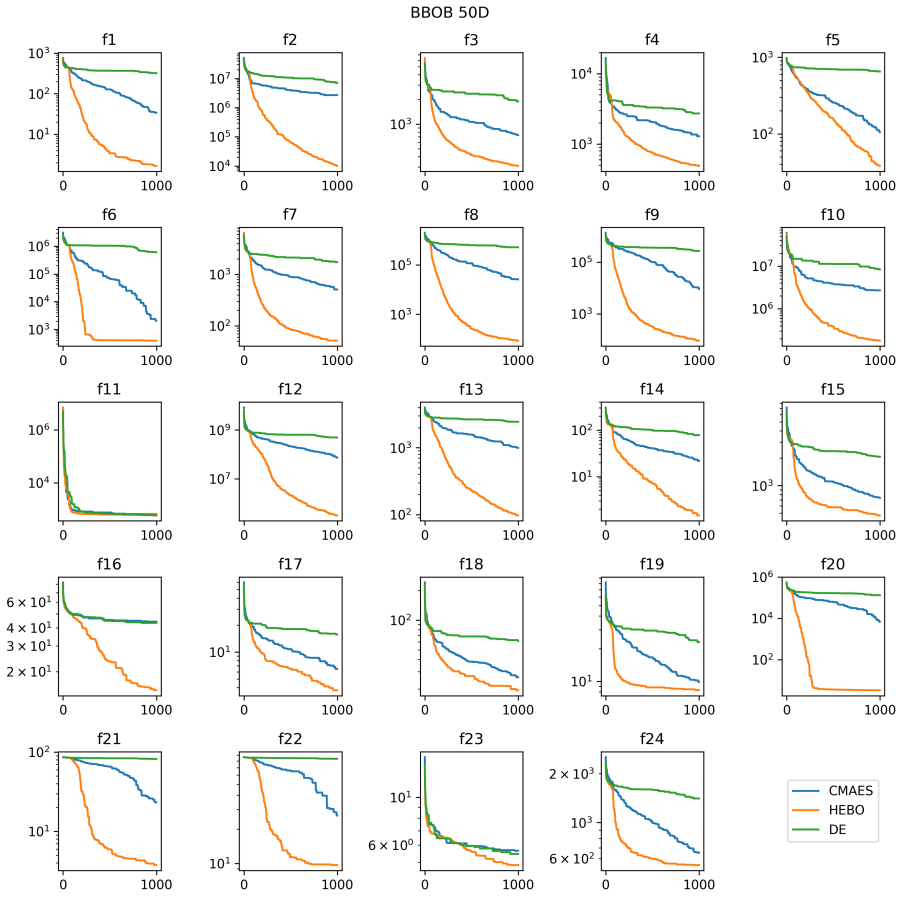


Figure 77: BBOB convergence curves for the 50D problems, for optimizers: CMA-ES, HEBO, and DE.

E Simulation Budget over Optimization Procedures

In general we find that DE makes much more calls to the infeasible region of the problem in comparison to CMA-ES and HEBO: Figure 78, Figure 79, and Figure 80. These graphs show the total evaluations we make to the problem, that is, feasible and feasible, in comparison to the simulation budget. The dashed line represents the limit: assuming all evaluations of the problem are feasible.

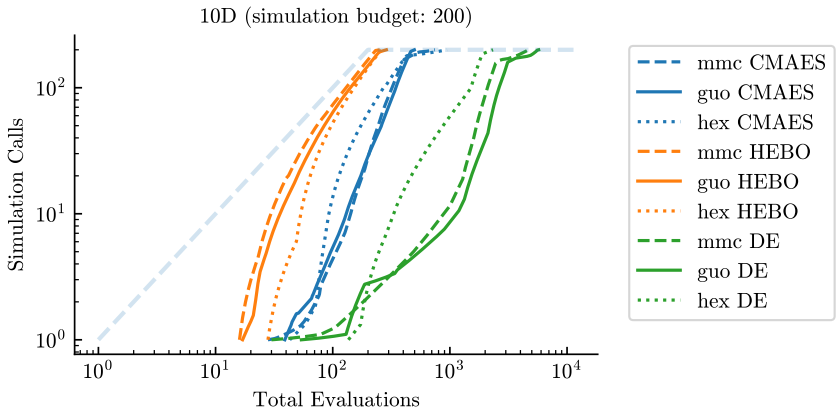


Figure 78: The total evaluations made to the problem against the number of simulation calls of these total evaluations, for the 10D parameterizations. And, the dashed line representing the limit assuming all evaluations were feasible.

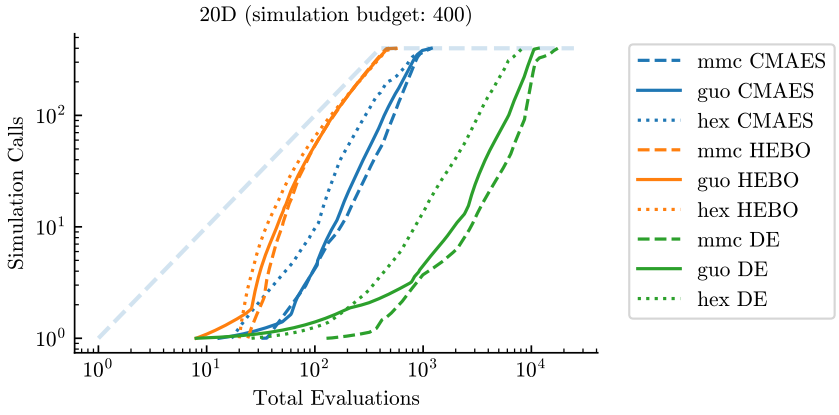


Figure 79: The total evaluations made to the problem against the number of simulation calls of these total evaluations, for the 20D parameterizations. And, the dashed line representing the limit assuming all evaluations were feasible.

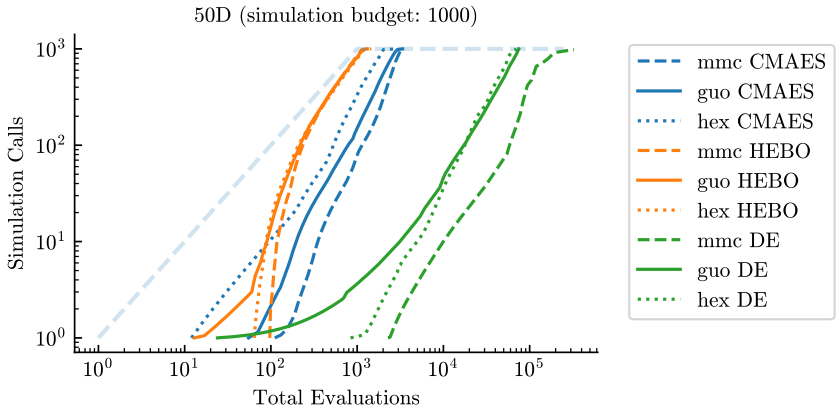


Figure 80: The total evaluations made to the problem against the number of simulation calls of these total evaluations, for the 50D parameterizations. And, the dashed line representing the limit assuming all evaluations were feasible.

F CMA-ES Extended Runs

Initially, a larger simulation budget was planned. However, it turned out to be infeasible to reach 5000 simulation calls with HEBO due to the computational overhead of its surrogate model. Despite this limitation, we still achieved results close to the SIMP baseline by using the CMA-ES method with a budget of $N = 100 \cdot d$ (Figure 81).

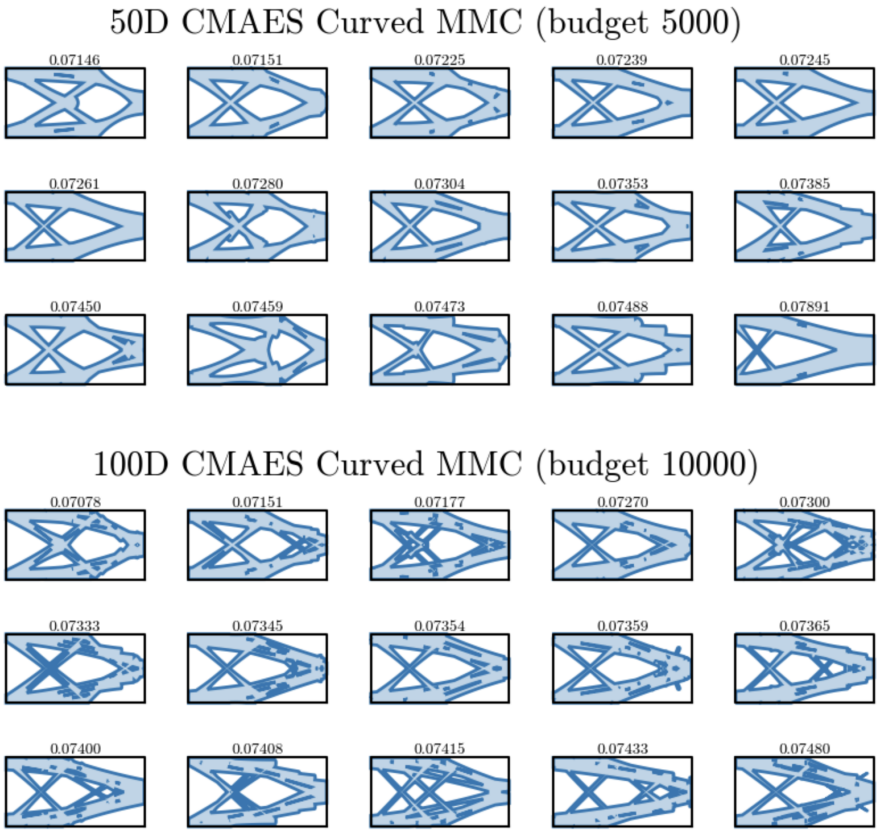


Figure 81: Final designs found with the CMA-ES optimizer on the Curved MMC parameterization on an extended budget of $100 \cdot d$ for: (top) 50D; and (bottom) 100D.