Interactive Exploration of Design Trade-Offs

ADRIANA SCHULZ and HARRISON WANG, Massachusetts Institute of Technology, USA
EITAN GRINSPUN, Columbia University, USA
JUSTIN SOLOMON and WOJCIECH MATUSIK, Massachusetts Institute of Technology, USA

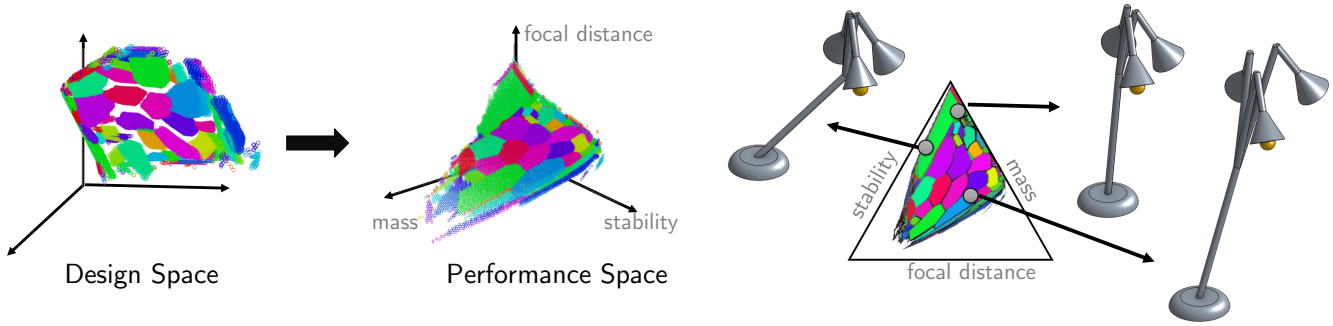

Fig. 1. Our method allows users to optimize designs based on a set of performance metrics. Given a design space and a set of performance evaluation functions, our method automatically extracts the *Pareto set*—those design points with optimal trade-offs. We represent Pareto points in design and performance space with a set of corresponding manifolds (left). The Pareto-optimal solutions are then embedded to allow interactive exploration of performance trade-offs (right). The mapping from manifolds in performance space back to design space allows designers to explore performance trade-offs interactively while visualizing the corresponding geometry and gaining an understanding of a model's underlying properties.

Typical design for manufacturing applications requires simultaneous optimization of conflicting performance objectives: Design variations that improve one performance metric may decrease another performance metric. In these scenarios, there is no unique optimal design but rather a set of designs that are optimal for different trade-offs (called Pareto-optimal). In this work, we propose a novel approach to discover the Pareto front, allowing designers to navigate the landscape of compromises efficiently. Our approach is based on a first-order approximation of the Pareto front, which allows entire neighborhoods rather than individual points on the Pareto front to be captured. In addition to allowing for efficient discovery of the Pareto front and the corresponding mapping to the design space, this approach allows us to represent the entire trade-off manifold as a small collection of patches that comprise a high-quality and piecewise-smooth approximation. We illustrate how this technique can be used for navigating performance trade-offs in computer-aided design (CAD) models.

CCS Concepts: • **Computing methodologies** → **Shape modeling**; *Graphics systems and interfaces*;

Additional Key Words and Phrases: Pareto optimality, design for manufacturing, shape modeling

Authors' addresses: Adriana Schulz, aschulz@csail.mit.edu; Harrison Wang, hwang123@mit.edu, Massachusetts Institute of Technology, Cambridge, MA, 02138, USA; Eitan Grinspun, eitan@cs.columbia.edu, Columbia University, New York, NY, 10027, USA; Justin Solomon, jsolomon@mit.edu; Wojciech Matusik, wojciech@csail.mit.edu, Massachusetts Institute of Technology, Cambridge, MA, 02139, USA.

0730-0301/2018/8-ART131 $15.00
https://doi.org/10.1145/3197517.3201385

ACM Reference Format:
Adriana Schulz, Harrison Wang, Eitan Grinspun, Justin Solomon, and Wojciech Matusik. 2018. Interactive Exploration of Design Trade-Offs . *ACM Trans. Graph.* 37, 4, Article 131 (August 2018), 14 pages. https://doi.org/10.1145/3197517.3201385

1 INTRODUCTION

In typical manufacturing applications, design goals are measured by how the finished products *perform* in the physical world. Design is driven by objectives such as weight, stability, durability, compliance, and other functionality-related metrics. Directly specifying designs that optimize performance objectives is challenging since it requires predicting physical behavior. For this reason, performance-driven computational methods are critical and have been widely explored in recent research on manufacturing-oriented design. These approaches guide users in efficiently exploring design spaces while optimizing for a given performance objective, which can be measured using physical simulations.

A fundamental limitation of typical design optimization techniques is that they require a single objective function for evaluating performance. In most applications, however, multiple criteria are used to evaluate the quality of a design. Structures must be stable *and* lightweight. Vehicles must be aerodynamic, durable, *and* inexpensive to produce. In most cases, the performance objectives are not only multiple but also *conflicting*: improving a design on one axis often decreases its quality on another axis. In reality, designers and engineers navigate a complex landscape of compromises, generating objects that perhaps do not optimize any single quality measure but rather are considered optimal under a given performance trade-off.

Since it is impossible to optimize more than one criterion at a time, standard optimization approaches require expressing a set of performance criteria in a combined objective function that balances

incompatible features. The typical approach is to use weighted combinations of different performance metrics. For example, in the case where there are two performance metrics, an objective could be expressed as $\alpha f_1(x) + (1-\alpha) f_2(x)$, where the f_i's are performance functions defined on the design space and $0 \leq \alpha \leq 1$ is a proxy for the trade-off between f_1 and f_2.

Unfortunately, such a proxy fails to capture the full space of optimal design trade-offs, called the *Pareto set*. A design point is Pareto-optimal if an adjustment to the design cannot simultaneously improve *all* performance metrics; any improvement to one metric necessarily worsens another. If $F(x) = (f_1(x), f_2(x))$ is a function that maps the *design space* onto the *performance space*, then F maps the Pareto *set* onto the Pareto *front*. While the main task of a designer might be considered navigation of the Pareto front, its shape is typically nonlinear and even disconnected. Linear changes in α do not correspond to linear, or even continuous, changes on the Pareto front (see Figure 4). For this reason, guessing a reasonable α, or sampling over α, can be imprecise, unstable, or even intractable. Furthermore, the introduction of new performance metrics renders any previous choice of α obsolete.

Instead of using proxy objectives, we seek to discover, represent, parameterize, and explore the Pareto front directly. In our approach, the Pareto front discovery is done in a pre-computation step, and the resulting representation is used to define an interactive tool that allows users to navigate the complex trade-offs between multiple performance metrics and instantaneously find corresponding designs. The main challenge in creating this representation is that finding all Pareto points requires on the one hand exploring the *diverse* solutions that correspond to different trade-offs and on the other hand *converging* to solutions in each particular direction to find points that are optimal in the Pareto sense.

From a technical perspective, our algorithm is built upon a first-order approximation of the Pareto front derived from duality theory in multi-objective optimization. This approximation serves two roles. First, it enables exploration of the Pareto front near a single point once it has been discovered. Second, it efficiently captures a *region* of the Pareto front, allowing us to approximate the entire front with just a few continuum pieces instead of a dense set of sample points. Each continuum piece also stores a mapping to the Pareto set in the design space. The end result is a technique that discovers the Pareto front and represents it as a union of relatively few manifold pieces that can be stored and queried efficiently within an interactive design tool that presents the designer with only the *relevant* (Pareto optimal) set of designs. Our representation is well-suited for the nonlinear and possibly disconnected nature of the Pareto front.

We evaluate our algorithm against the well-established ZDT benchmark suite [Zitzler et al. 2000] for multi-objective optimization, which covers a broad range of realistic geometric forms for the Pareto front and Pareto set. We then demonstrate our software tool in the context of interactive design based on real-world CAD models.

2 RELATED WORK

Performance-driven design. In many manufacturing applications, design specifications are determined by how they affect the performance of the resulting model. Design for functionality means finding the optimal configuration for a set of performance objectives. This requires solving inverse problems that are challenging in large design spaces. Previous works address these challenges with two main approaches: exploration with iterative feedback and direct optimization. From furniture design [Umetani et al. 2012], to model airplanes [Umetani et al. 2014] and robots [Megaro et al. 2015], exploration tools allow users to navigate a design space with real-time feedback on performance. On the other hand, optimization approaches directly determine a final configuration that achieves desirable properties, such as structural stability [Prévost et al. 2013; Whiting et al. 2012], frequency spectra [Bharaj et al. 2015], motion [Bächer et al. 2015; Du et al. 2016], or buoyancy [Musialski et al. 2015].

Optimization has the advantage of being automatic, while user exploration can be time-consuming. Exploration tools, however, are advantageous when multiple considerations are taken into account. In such scenarios, it is challenging, if not impossible, to determine a single objective function for direct optimization; therefore, an exploration interface can more effectively guide users to solutions with desired performance trade-offs. Our approach combines the advantages of both techniques. We use offline optimization to reduce the search space to solutions that achieve optimal performance trade-offs. We then define an exploration interface that allows users to search in this subset. This approach lets us handle problems with multiple performance objectives while reducing the amount of user effort during exploration, since the search spaces are smaller and all discovered solutions are Pareto-optimal. While multi-objective optimization approaches that discover a set of Pareto-optimal points have become increasingly common in engineering practice [Agrawal and van de Panne 2013; Bandaru and Deb 2015; Deb and Srinivasan 2006], in this work we propose a novel technique that finds a sparse representation of the full front, allowing for interactive exploration of design trade-offs.

Performance Space Exploration. Recent works on design exploration propose pre-computing the performance metrics on a constrained design space to provide real-time performance feedback as users explore variations in the *design* space [Schulz et al. 2017; Shugrina et al. 2015; Yau et al. 2006]. While these works enable efficient exploration of models that can be parametrized with a small set of parameters, their main limitation is that by sampling in the design space they suffer the curse of dimensionality and cannot be used in applications where design spaces are high-dimensional. In this work, we argue that in designing for functionality, it is not necessary to represent the full design space since only a subset of solutions will correspond to optimal design trade-offs. By representing only this subset, which lies in a much lower dimensional *performance* space, our approach can not only scale to large design spaces but also allows for a more meaningful exploration based on performance trade-offs.

Our approach is also related to works on material design that use pre-computation to find the gamut of achievable material properties [Bickel et al. 2010; Dong et al. 2010; Zhu et al. 2017]. In these works, the design space is the set of materials that can be output by a given device, and the performance metrics are the material properties that are evaluated. Our approach is the most similar to [Zhu et al. 2017], which combines a probabilistic search and a continuous optimization when pre-computing the material gamut. While our method draws ideas from these approaches, the fundamental difference is that instead of a gamut that defines all the possible material combinations in a d-dimensional material property space, our application requires defining a combination of $(d-1)$-dimensional manifolds that represents the optimal performance trade-offs.

Multi-Objective Optimization. The task of finding the set of optimal design trade-offs amounts to solving a multi-objective optimization problem, where the objectives are the performance metrics. Numerous methods have been proposed for solving multi-objective optimization problems. The Normal Boundary Intersection [Das and Dennis 1998] and Normalized Normal Constraint [Messac et al. 2003] methods aim to produce a well-distributed set of solutions, which can accurately approximate the shape of the Pareto front but can produce false positive solutions when the problem is non-smooth and/or contains local minima. Evolutionary algorithms often are applied to address this issue by iteratively modifying a population of candidate solutions that undergoes reproduction and removal similar to natural evolution; see [Zhang and Xing 2017] for a survey.

The main difference between these methods and our approach is that instead of searching for a diverse set of discrete points, our proposed method provides a compact representation covering contiguous regions within the space of solutions. Our approach leverages the fact that, while discovering individual points on the Pareto front can be difficult, locally searching around known solutions is easier. Using our method we obtain a relatively small set of manifolds whose union approximates the Pareto front. This representation provides an easily-navigable set of solutions in both design space and objective space, which can be applied to visualization and analysis.

3 OVERVIEW

After walking through a concrete example, we lay out the algorithm in detail.

3.1 Motivating Example

To motivate our choice of mathematical structures and computational approach, consider designing a wrench. As evidenced by the size of a standard toolbox, many styles and shapes of wrenches are potentially useful: Small, lightweight wrenches may be portable and needed for tweaking small mechanical parts, while larger, heavy-duty wrenches exert sufficient torque to turn structurally-critical bolts.

We can measure the quality of a wrench on multiple performance axes. We may prefer lighter wrenches for portability or powerful wrenches for applying more torque. These performance objectives are not harmonious, and hence the engineer designing the wrench must navigate a space of *trade-offs*.

A wrench design can be described by certain *design parameters*, e.g. thickness, material density, and handle width; as a rule of thumb, these are the parameters that are exposed in a parametric CAD model. We restrict ourselves to manufacturable designs, whose design parameters are compatible with our fabrication process. The design parameters serve as coordinates for the *design space* of potential wrenches. After choosing a specific design, we measure its mechanical and physical properties and plot them on a set of performance axes: torque, weight, and so on. This procedure yields a map from design space to *performance space*.

Some wrench designs are not worth considering. Consider a heavy, weak wrench A. Suppose that B is lighter *and* more powerful A. We say that B *dominates* A. There is never a reason to manufacture wrench A: the dominating wrench is preferable in terms of all metrics. On the other hand, if wrench C is more powerful than B *but* also heavier, we have a trade-off to navigate.

After eliminating dominated wrenches, what remains is the subset of interesting wrenches known as the *Pareto set*; the image of the Pareto set in performance space is a lower-dimensional set known as the *Pareto front*. Designing a wrench is therefore a navigation of the Pareto front, seeking the optimal trade-off for a particular application.

Our goal is to develop a computational tool for this navigation process. We provide a technique for efficiently approximating and visualizing the Pareto front, represented as a collection of smooth manifolds embedded in performance space. We then expose the lower-dimensional space of interesting Pareto alternatives in an efficiently-navigable fashion.

3.2 Outline

The input to our method is a parameterized object (e.g., a wrench) and a map from the design parameters (e.g., thickness, density) to the relevant performance metrics (e.g., weight, torque).

To represent the Pareto front succinctly, we derive a first-order approximation in a local neighborhood of the front (§5). Inspired by statistical techniques like principal component analysis (PCA) and canonical component analysis (CCA), we observe that *relatively few of these linear subspaces are needed to capture local variability*. Informed by this observation, we employ the following steps:

- We propose a randomized algorithm for uncovering points on the Pareto set (§6). Our algorithm is designed to encourage diversity in the sampling procedure, helping capture the breadth of possible designs.
- Once a point on the Pareto set is computed, we use our first-order approximation derived in §5 to capture local variability in the relationship between design and performance as a small manifold.
- Pareto-relevant points and their associated manifold approximations are stored in a *performance buffer* that efficiently tracks and updates its estimate of the front (§6.1).
- Once sampling has converged, we are left with a performance buffer full of manifold approximations of the Pareto front. We use a graph cut technique to sparsify our approximation of the front. The end result is a small collection of manifolds that comprises a high-quality and piecewise-smooth approximation of the trade-off manifold (§6.4).

- The dimensionality of the Pareto front generally is lower than the dimensionality of performance space. Hence, we propose a technique for embedding and navigating just the Pareto front, quickly revealing a diverse set of interesting designs (§6.5).

Beyond testing on engineering problems, we also test our algorithm on a standard benchmark of optimization problems from the multi-objective optimization literature (§7.1), verifying that we have not only an effective engineering tool but also an efficient technique for challenging multi-objective sampling problems.

4 MATHEMATICAL PRELIMINARIES

In this section, we introduce the mathematical toolbox that formalizes the idea of trade-offs in an engineering context. This quick summary establishes the notation we need in our paper; for a broader discussion we refer the reader to [Deb and Deb 2014].

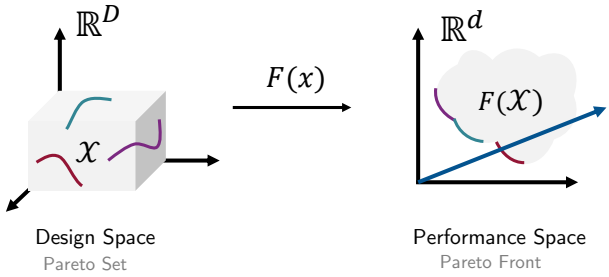

Fig. 2. The Pareto set represents the points in design space with optimal performance trade-offs that get mapped to the Pareto front in performance space. Different colors indicate different manifolds in design and performance space with a one-to-one mapping. Any ray from the origin (blue line) can only intersect the Pareto front once.

4.1 Definitions

The set of exposed parameters for an engineering model is known as the *design space*:

Definition 4.1 (Design space and constraint). The *design space* $\mathcal{X}$ for a multi-objective problem is defined as a subset of $\mathbb{R}^D$ of feasible points:

$$\mathcal{X} := \{\mathbf{x} = (x^1, \ldots, x^D) \in \mathbb{R}^D : g_j(\mathbf{x}) \leq 0 \ \forall j \in \{1, \ldots, K\}\}.$$

Here, each function g_j represents a single constraint on $\mathbf{x}$. We use $G(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^K$ to denote the concatenation $(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$.

Intuitively, $\mathcal{X}$ is the set of all manufacturable objects. Constraints g_j might capture hard constraints identified by the engineer, such as a limit on the total material available to manufacture an object.

Next, we need a notion of an objective function for optimization:

Definition 4.2 (Performance metric and space). A set of *performance metric* functions $f_i : \mathbb{R}^D \to \mathbb{R}$ assign real-values to each design vector $\mathbf{x}$; we use $F(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^d$ to denote the concatenation $(f_1(\mathbf{x}), \ldots, f_d(\mathbf{x}))$. We choose the convention that *small* values of $f_i(\mathbf{x})$ are desirable for metric f_i. The *performance space* $\mathcal{S}$ is the image of the design space $\mathcal{X}$ under the performance metrics:

$$\mathcal{S} := F(\mathcal{X}) \subseteq \mathbb{R}^d.$$

The performance metric functions are often complex and computationally intensive. Multi-objective problems typically involve multiple performance metrics ($d \geq 2$), e.g. weight, torque, and other measures; typically $d \ll D$.

Our algorithm reveals only those points in design and performance space that are not out-performed on every axis by some other design:

Definition 4.3 (Pareto optimality). A point $\mathbf{x} \in \mathcal{X}$ is *Pareto optimal* if there does not exist any $\mathbf{x}' \in \mathcal{X}$ so that $f_i(\mathbf{x}) \geq f_i(\mathbf{x}')$ for all i and $f_i(\mathbf{x}) > f_i(\mathbf{x}')$ for at least one i. The set of all Pareto-optimal points is the *Pareto set* $\mathcal{P} \subseteq \mathbb{R}^D$; the image $F(\mathcal{P}) \subseteq \mathbb{R}^d$ is the *Pareto front.*

4.2 KKT Conditions

Our algorithm pre-computes $F(\mathcal{P})$ and represents it compactly to allow for fast exploration and optimization. We represent $F(\mathcal{P})$ as a union of $(d-1)$-dimensional manifolds. As we will show, for (almost) any point $\mathbf{x} \in \mathcal{P}$, there is a local neighborhood $\mathcal{B}(\mathbf{x})$ such that $F(\mathcal{P}) \cap \mathcal{B}(\mathbf{x})$ is a $(d-1)$-dimensional manifold in $\mathbb{R}^d$; we store $F(\mathcal{P})$ as the union of a sparse set of simple manifold approximations.

Intuitively, this approximation is justified as follows. From the definition of Pareto optimality, if a point $x \in \mathcal{X}$ is Pareto optimal, then there is no other point whose performance is not worse than x in every metric and better than x in at least one metric. Hence, we can draw a ray in performance space along which there can be at most one Pareto optimal point, illustrated in Figure 2. This effectively extracts just points on the boundary of the Pareto front, a lower-dimensional set and justifies the following proposition:

Proposition 4.4. *For every nonnegative $\alpha \in \mathbb{R}^d$, there exists at most one $t > 0$ such that $t\alpha \in \mathcal{P}$.*

We parameterize this set using a "performance buffer," defined in §6.1.

This lower-dimensional observation is justified by the theory of *KKT conditions* from multi-objective optimization. Following [Deb and Deb 2014], we denote Pareto-optimal points as solution of the *primal* problem notated

$$\begin{aligned} \min_x \quad & \{f_i(x)\} \\ \text{s.t.} \quad & x \in \mathcal{X}. \end{aligned} \tag{1}$$

Standard approaches to multi-objective optimization aim at finding solutions to this primal problem. The main challenge is that the space of solutions that are Pareto-optimal is large, disconnected, and prone to local minima. Typical approaches use genetic algorithms to find solutions that are diverse and optimal [Zhou et al. 2011]. On a high level, these methods try to reach the Pareto front by searching in different directions and using randomization to avoid local minima.

Inspired by *primal-dual algorithms* in optimization, the key insight in our approach is that while discovering a single point on the Pareto set is challenging, once a point has been found it can be used to uncover a large Pareto region on its neighborhood. Our approach is to consider a *dual* problem, defined by the so-called KKT conditions:

Proposition 4.5 (KKT conditions [Hillermeier 2001]). *Assuming that f_i and g_k are continuously differentiable and that the*

vectors $\{\nabla g_{k'}(\mathbf{x}^*) \mid k' \text{is an index of an active constraint}\}$ *are linearly independent, then for any solution* $\mathbf{x}^*$ *to Equation 1 there exist dual variables* $\alpha \in \mathbb{R}^d$ *and* $\beta \in \mathbb{R}^K$ *such that*

$$\left\{ \begin{array}{l} \mathbf{x}^* \in \mathcal{X} \\ \alpha_i \geq 0 \;\; \forall i \in \{1, \ldots, d\} \\ \beta_k \geq 0 \;\; \forall k \in \{1, \ldots, K\} \\ \beta_k g_k(\mathbf{x}^*) = 0 \;\; \forall k \in \{1, \ldots, K\} \\ \sum_{i=1}^{d} \alpha_i = 1 \\ \sum_{i=1}^{d} \alpha_i \nabla f_i(\mathbf{x}^*) + \sum_{k=1}^{K} \beta_k \nabla g_k(\mathbf{x}^*) = 0 \end{array} \right\} \tag{2}$$

It is worth noting that KKT conditions are necessary but not sufficient, so we must *a posteriori* check that candidate points satisfying these conditions are indeed Pareto-optimal. This check is extremely efficient thanks to our performance buffer. The KKT conditions verify that at least locally the Pareto front is $(d-1)$-dimensional, thanks to the constraint on the sum of α.

5 FIRST-ORDER APPROXIMATION

We begin our technical discussion by motivating a first-order approximation of the Pareto front. This formula is a straightforward corollary of the KKT conditions in Proposition 4.5; conceptually, it characterizes the proper directions to walk in design and performance space to maintain Pareto optimality after a single Pareto point is found. This formula can be understood as a source of efficiency for our algorithm relative to other sampling algorithms, since entire neighborhoods rather than individual points on the Pareto front are captured.

We state our condition as follows:

Proposition 5.1 (KKT Perturbation). *Suppose* $\mathbf{x}(t) : (-\varepsilon, \varepsilon) \rightarrow \mathbb{R}^D$ *is in the Pareto set in a neighborhood of* $t = 0$*, that is,* $\mathbf{x}(t) \in \mathcal{P}$ *for all* $t \in (-\varepsilon, \varepsilon)$*. Taking* α *and* β *to be the KKT dual variables corresponding to* $\mathbf{x}^* := \mathbf{x}(0)$*, under the assumptions from Proposition 4.5 we have*

$$H\mathbf{x}'(0) \in \mathrm{Im}(DF^\top(\mathbf{x}^*)) \oplus \mathrm{Im}(DG_{K'}^\top(\mathbf{x}^*)). \tag{3}$$

where

$$\mathrm{H} := \sum_{i=1}^{d} \alpha_i H_{f_i}(\mathbf{x}^*) + \sum_{k=1}^{K'} \beta_k H_{g_k}(\mathbf{x}^*).$$

Furthermore,

$$DG_{K'}(\mathbf{x}^*)\mathbf{x}'(0) = 0. \tag{4}$$

Here H_u and D_u represent the Hessian and Jacobian of a function u, respectively; $DG_{K'}$ indicates the part of the Jacobian DG corresponding to the $K' \leq K$ active constraints. We prove this proposition in Appendix A.

Generically, these define a $(d-1)$-directional space of local exploration directions around $\mathbf{x}^*$. In particular, assuming H is invertible, (3) shows that $\mathbf{x}'(0)$ is in a $(d + K' - 1)$-dimensional space; the -1 comes from the last line of (2), which effectively shows that the row spaces of DF and $DG_{K'}$ are linearly dependent. Equation (4) reduces the dimensionality by K', as needed.

6 PARETO FRONT DISCOVERY

We now define an iterative algorithm for exploring the Pareto front, constructed from the above perturbation formula. Typical iterative approaches seek a diverse, dense set of solutions (points) that together approximate the Pareto front. In contrast, our algorithm represents the front piecewise-continuously as a set of manifolds. Our algorithm is therefore less sensitive to the uniformity of discovered points, so long as the manifolds expanded from these points completely cover the Pareto front. Instead, the distribution of points in our discovery algorithm will depend on the varying sizes and locations of the generated manifolds in objective space.

6.1 Data Structure

Assuming that performance metrics are positive, any point in the Pareto front will intersect a positive ray traced from the origin. From Proposition 4.4, any such ray will intersect at most one point in the Pareto front. Further, a Pareto point that intersects a given ray will have minimal distance to the origin when compared to all other points in performance space $\mathcal{S}$ that intersect this ray.

We therefore define the performance buffer as a $(d-1)$-dimensional array discretized using (hyper)spherical coordinates (see Figure 3). Inspired by the z-buffer used in rendering, a basic implementation of the performance buffer stores at each cell the point with minimum distance to the origin that intersects its corresponding ray. The performance buffer is updated at each iteration of the discovery algorithm, as new regions of the performance space are found.

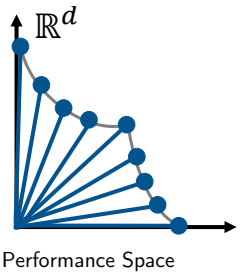

Fig. 3. The performance buffer: since a ray from the origin can only intersect one point in the Pareto front, we use a buffer discretized by (hyper)spherical coordinates for storage.

To reduce stochasticity of our final result, in practice we extend the basic implementation by storing a list of *candidate solutions* at each buffer cell $B(j)$, instead of storing only the single solution that has minimal distance to the origin. These solutions are included if their distance to the origin is within an allowed tolerance δ_B of the minimal distance over all solutions in $B(j)$. Maintaining this set of solutions is useful for extracting a *sparse* approximation of the Pareto front (§6.4), which may forego choosing the closest sample to the origin at some buffer cells in favor of a simpler set of manifolds covering the front. For performance, we bound the number of stored solutions per cell, keeping only the top K ($K = 50$ for all experiments).

6.2 Discovery Algorithm

6.2.1 Overview. We address diversity and convergence with an iterative procedure to discover the Pareto front. The algorithm is composed of three main steps (see Algorithm 3.2 and Figure 4). The first step is a *stochastic sampling* scheme that selects samples $\mathbf{x}_S^i$,

$i = 1, \ldots, N_S$ in the design space $\mathcal{X}$ (§6.2.2). The second step is a *local optimization* procedure that tries to push each sample $\mathbf{x}_s^i$ to a solution $\mathbf{x}_o^i$ on the Pareto set (§6.2.3). For each sample $\mathbf{x}_s^i$, a search direction $\mathbf{s}(\mathbf{x}_s^i) \in \mathbb{R}^d$ is selected to drive the local optimization scheme. Diverse directions are used to find a set of solutions that cover the different regions of the Pareto front. Finally, a *first-order approximation* of the Pareto front is extracted around $\mathbf{x}_o^i$ (§6.3). The perturbative formula in §5 is used to generate the $D \times (d-1)$ matrix M^i that defines an affine subspace $\mathcal{A}^i$ in design space that goes though $\mathbf{x}_o^i$.

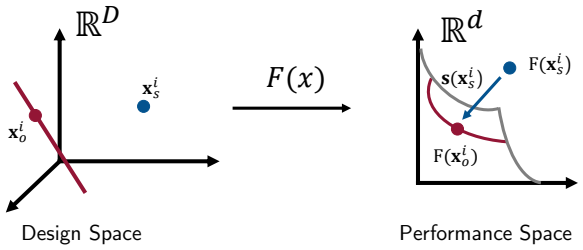

Fig. 4. A single iteration of the discovery algorithm: random samples $\mathbf{x}_s^i$ are generated from the current data on the buffer (illustrated in gray) and optimized for a search direction $\mathbf{s}(\mathbf{x}_s^i)$ (blue arrow). A first-order approximation around the result of this optimization, $\mathbf{x}_o^i$, generates the corresponding manifolds in both design and performance space (red lines) and the buffer is updated based on this new data.

The resulting manifold $F(\mathcal{A}^i)$ in performance space is then projected onto the buffer. If a point on $F(x^i) \in F(\mathcal{A}^i)$ is projected onto the buffer cell $B(j)$ and this point is considered a candidate according to the tolerance δ_B, then the buffer is updated. Each cell j on the buffer stores a list of solutions, each of which contains the point in design space, the corresponding map to performance space, and the corresponding affine subspace $\{(\mathbf{x}^i, F(\mathbf{x}^i), \mathcal{A}^i)\}_i$. The algorithm terminates if the buffer cells' average distance to the origin is not improved by δ_I after N_T iterations.

Algorithm 1 Pareto set discovery given performance metrics F and design constraints that define $\mathcal{X}$.

procedure ParetoFrontDiscovery($\mathcal{X}$, F)
  B: performance buffer array
  $B(i) \leftarrow \emptyset, \forall i$
  do
    $\mathbf{x}_s^0, \ldots, \mathbf{x}_s^{N_s} \leftarrow$ stochasticSampling(B, F, $\mathcal{X}$)
    for each $\mathbf{x}_s^i$ **do**
      $D(\mathbf{x}_s^i) \leftarrow$ selectDirection(B, $\mathbf{x}_s^i$)
      $\mathbf{x}_o^i \leftarrow$ localOptimization($D(\mathbf{x}_s^i)$, F, $\mathcal{X}$)
      $M^i \leftarrow$ firstOrderApproximation($\mathbf{x}_o^i$, F, $\mathcal{X}$)
      updateBuffer(B, $F(M^i)$)
    if buffer not updated on past N_i iterations **then**
      break
  while within computation budget
  return B

6.2.2 Stochastic Sampling. We use stochastic sampling to initialize each iteration of the algorithm to avoid local minima. Since the performance buffer stores the current approximation of the Pareto front, we use these points as initial guesses. We uniformly sample buffer cells at random. For each selected cell j, we take the point $\mathbf{x}^j$ with minimal distance to the origin and perturb it as follows:

$$\mathbf{x}_s = \mathbf{x}^j + \frac{1}{2^{\delta_p}}\mathbf{d}_p$$

where $\mathbf{d}_p$ is a uniform random unit vector that defines a stochastic direction and δ_p is a uniform random number in $[0, \delta_P]$ used for scaling. The exponential factor trades off between exploration and exploitation—small scaling factors are typically preferred to explore local neighborhoods, but occasional larger values are desired to diversify the solutions. We clamp the result to ensure $\mathbf{x}_p \in \mathcal{X}$. In the first iteration, points are sampled uniformly from $\mathcal{X}$.

6.2.3 Local Optimization. Each of the sampled points $\mathbf{x}_s$ is then optimized for Pareto optimality. A scalarization scheme is used to convert this multi-objective optimization problem into a single-objective problem which can be solved for each point. Previous work on multi-objective optimization proposes assorted scalarization functions that diversify the solutions across the Pareto front [Das and Dennis 1998]; diversification is essential to avoid having solutions cluster in certain areas, failing to provide a good representation for the shape.

We find that the following scalarization function is most effective in our applications:

$$\mathbf{x}_o = \arg\min_{\mathbf{x} \in \mathcal{X}} \|F(\mathbf{x}) - \mathbf{z}(\mathbf{x}_s)\|^2 \tag{5}$$

where $\mathbf{z}(\mathbf{x}_s) \in \mathbb{R}^d$ is a reference point defined for each sample. This quadratic expression is inspired by previous work [Zeleny 1973] and allows for discovery of solutions on non-convex regions of the Pareto front.

We use the performance buffer discretization to specify a unit-length search direction $\mathbf{s}(\mathbf{x}_p)$ for pushing $\mathbf{x}_s$ towards the Pareto front (see Figure 5). This suggests choosing the reference point $\mathbf{z}(\mathbf{x}_p)$ as:

$$\mathbf{z}(\mathbf{x}_s) = \mathbf{x}_s + \mathbf{s}(\mathbf{x}_s)C(\mathbf{x}_s), \tag{6}$$

where $C(\mathbf{x}_s) = \delta_s \|\mathbf{x}_s\|$ is a scaling factor depending on the distance to the origin. This scaling factor is important for diversity since setting the reference point too far from the Pareto front will make results cluster around specific solutions.

As shown in Figure 5, the buffer discretization defines a search direction for each buffer cell j. For further diversity, instead of setting $\mathbf{s}(\mathbf{x}_s)$ as the search direction for the buffer cell j where $\mathbf{x}_s$ get projected, we select the search direction assigned to a cell on the neighborhood of cell j selected uniformly at random. The neighborhood of a cell is defined by all the cells that are within distance δ_N.

6.3 First-Order Approximation

For each point $\mathbf{x}_o^i$, we use the result in Proposition 5.1 to find $d-1$ directions for local exploration stored in a matrix M^i.

As discussed in §5, equations (3) and (4) generically define a $d-1$ dimensional space. In practice, however, one must consider two

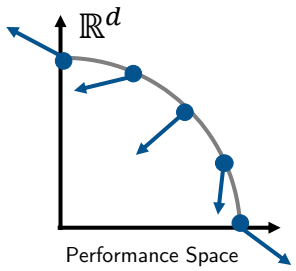

Fig. 5. Search directions for buffer cells for $d = 2$. For diversity, different regions of the performance space get assigned different search directions. We use the buffer discretization to define these directions as illustrated in the figure.

special cases: (1) when $\mathrm{Im}(DF^\top(\mathbf{x}^*)) \oplus \mathrm{Im}(DG_{K'}^\top(\mathbf{x}^*))$ is low rank and (2) when H is low rank.

The first special case occurs when two performance objectives agree. In this case, the Pareto front locally will be represented by a manifold with dimensionality smaller than $d-1$. Consider, for example, a two-objective problem where the performance metrics are the weight and material cost of a single-material model. Since these objectives are not conflicting, the Pareto front is defined by a single point where the volume is minimized. Typically this is not the case for interesting problems in multiobjective optimization where the challenge is due to conflicting objectives. Therefore, in our implementation, we assume that this does not happen.

The second special case is more common, since it results from having design variables that do not affect the performance. While design variables that do not affect the performance at any configuration can be easily discarded in a pre-processing step, it is common to have design variables that have overall impact but locally are ineffective. In such cases, equations (3) and (4) define a space with dimensionality higher than $d-1$. This means that locally the Pareto *set* (design space) has higher dimensionality. Since the Pareto *front* (performance space) can never have dimensionality greater than $d-1$ (see Proposition 4.4), however, this means that there are multiple affine subspaces that locally map to the Pareto front. In our implementation, we deal with these cases by selecting $d-1$ directions uniformly at random.

6.3.1 Storage. Given M^i, which defines an affine subspace around $\mathbf{x}_o^i$, we find an orthonormal frame and uniformly sample on a grid defined by this frame in design space. We set the grid size to be large enough to reach the boundaries of $\mathcal{X}$ and discard points that are not in $\mathcal{X}$. We then map all valid points to performance space using F and project the results onto the buffer. For $d = 2$, this projection is done by interpolating line segments, and for $d = 3$, we define a triangle mesh and use barycentric coordinates for interpolation.

As previously discussed, the buffer stores all of the solutions that are within a given tolerance. For each solution $(\mathbf{x}, F(\mathbf{x}), \mathcal{A})$ mapped to a cell j in the buffer we compare its distance to the origin with the minimal distance stored in the solutions for cell j. If the result is within δ_B, we append it to the solution list for that cell; otherwise it is rejected. If the solution is closer to the origin than any other solution on the buffer, we traverse the list rejecting all candidate solutions that are no longer within tolerance.

6.4 Sparse Approximation

After the Pareto front has been discovered, the final step is to select for each cell on the buffer a single solution from the list of candidate points. Our goal is to assign a unique value to each buffer cell to minimize discontinuities in design space while maintaining optimality within tolerance. In cases with many design variables, it is possible to have more than one solution in design space that maps to the same point in the Pareto front. We aim at selecting between these solutions so that adjacent buffer cells map to solutions that are close in design space. Further, we want a sparse set of first-order approximations that accurately represents the Pareto front.

Since points that are represented by the same first-order approximation are close in design space, we can optimize for both of these objectives by solving a labeling problem. Each label l^i corresponds to a linear subspace $\mathcal{A}^i$ on the set of spaces found by the discovery algorithm. Our goal is to choose a label for each buffer cell j so that: (1) the label of a cells is similar to the label of its neighbors and (2) the assigned label for a given cell is on the list of solutions $B(j)$, with priority given to solutions with smaller distance to the origin. This can be expressed and solved as a graph-cut problem.

Compared to an approach that takes the best value in each buffer cell, our graph-cut algorithm finds a sparse set of first-order approximations, providing local continuity at the expense of additional approximation error. We define the approximation error $e_A(j,i)$ associated to assigning label l^i to cell $B(j)$ as the difference between the distance to the origin of the candidate solution on $\mathcal{A}^i$ and the minimal distance to the origin of all solutions in $B(j)$. The graph-cut formulation aims at segmenting the buffer into large continuous regions while minimizing this error. From the buffer construction, the error is bounded by a user-defined tolerance, δ_B. In typical applications, engineering safety factors should be used to determine this tolerance.

We use the technique described in [Boykov et al. 2001] and the provided implementation. The unary term $E_U(j,i)$ is set to $e_A(j,i)/\delta_B$ if $\mathcal{A}^i$ is on the list of candidate solutions for $B(j)$ and C_{inf} otherwise. The binary term $E_B(j,k)$ is set to 1 for every point if j and k are within a δ_R neighborhood from each other. In our experiments, we set $C_{\mathrm{inf}} = 10$ and $\delta_R n = 2$. A post-processing step is performed to filter out outliers.

6.5 Visualization

The performance buffer provides a discretization of the points on the Pareto front but may also contain points that are not Pareto-optimal. The final step of the algorithm is to remove all buffer cells that fall into this latter category. This can be done by simply checking for dominance based on Definition 4.3.

For visualization, we embed the buffer in a $(d-1)$-dimensional space to allow for easy exploration (see Figure 1). For $d = 2$, the embedding is a line, and for $d = 3$ the embedding is a triangle. In both cases, the extreme points correspond to maximizing a given performance metric. Since we handle minimization problems, each metric is optimized at the opposite vertex for $d = 2$ or edge for $d = 3$.

We define a color map for the embedding to highlight different affine subspaces. We color the embedded shape assigned to each point as a hue based on the corresponding affine subspace and a value based on the distance between neighbors. This allows us to see the transitions in the design space.

The pre-computed one-to-one mapping between piecewise linear regions in design space and their corresponding manifolds in performance space allows us to generate the geometry that corresponds to each performance trade-off in real time, allowing for interactive navigation of this embedded space.

7 RESULTS

We have implemented the algorithm above and run experiments for $d = 2$ and $d = 3$ with design parameters varying from $D = 3$ to $D = 21$ depending on the problem. For all of the experiments, we rescale the problems so that both $\mathcal{X} \in [0,1]^D$ and $F(\mathcal{X}) \in [0,1]^d$ and use the same parameter settings for all experiments: $\delta_B = 10^{-2}$, $\delta_I = 10^{-4}$, $\delta_S = 0.3$, $\delta_P = 10$, and $\delta_N = 0.2|B|$, where $|B|$ is the buffer size.

7.1 Experiments

We test the proposed algorithm against a set of well-established benchmark problems for multi-objective optimization, each of which covers a number of criteria for discovering the Pareto front.

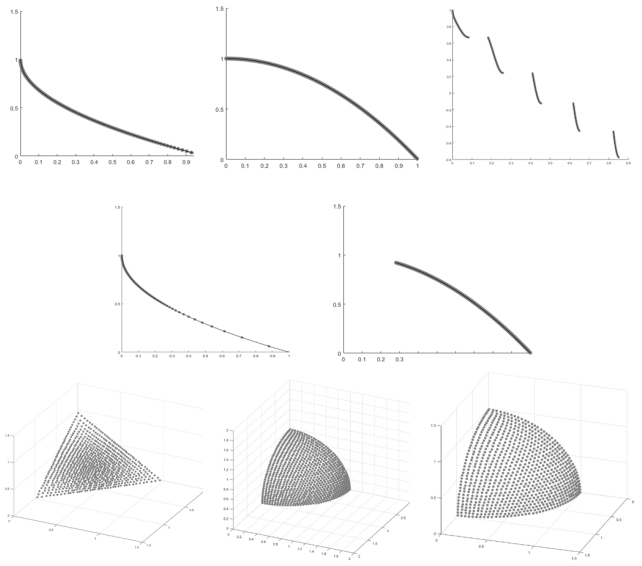

Fig. 6. Nondominated solutions for various well-established benchmark problems using our proposed approach. Top two rows: Solutions for the five real-valued ZDT problems. Bottom row: Solutions for the first three DTLZ problems with three objectives. Our approach was able to converge to the ground truth Pareto front in all cases.

The ZDT test suite [Zitzler et al. 2000] is perhaps the most widely-applied test suite for multi-objective optimization. This is due largely to the fact that the five real-valued tests in the set cover a broad range of geometric forms in both the Pareto front and Pareto set (concave, convex, disconnected). Additionally, these tests highlight the difficulties of multimodality (presence of multiple local minima) and, because they have well-defined optimal solutions, they are easily verifiable. The DTLZ test suite [Deb et al. 2002] offers little new in the way of geometric complexity, but it does provide the option to optimize for more than two objectives. This is an especially rare attribute for well-established test suites and is a crucial requirement for the scope of our project. In particular, the reference point mapping and manifold generation become less intuitive for higher dimensional problems. For this reason, we use the DTLZ suite to validate and visualize the results of our method in three dimensions.

Figure 6 shows the results of the points stored in the performance buffer over the true front which is known for these test functions, validating that our proposed method manages to converge to the correct solution for these examples.

These test functions provide an empirical validation that our method can avoid local minima, discovering the true front even for multi-objective optimization problems that are designed to be challenging. Compared to state of the art methods for Pareto front discovery (according to a recent survey [Zhang and Xing 2017]), our method generates a collection of manifolds, as opposed to point samples on the front (see Figure 7 of [Zhang and Li 2007], which contains an identical experiment to our Figure 6). The computation time, measured by the number of function evaluations, is at least comparable to the state of the art. For example, Zhang and Li report $\sim$10000 function evaluations for ZDT1 with 15 parameters, while our method uses only 6100 function evaluations. In addition, our final representation is more compact—since all ZDT benchmark examples can be represented by a single manifold—and is amenable to parallelization.

Since the Pareto set for each of the ZDT and DTLZ functions lies on a single affine subspace, our method recovers the entire front after finding a single solution that is Pareto-optimal. Therefore, to stress-test our approach and demonstrate that it can generically approximate the Pareto front by piecewise linear regions in design space, we test our method on a "Fourier benchmark": functions defined by linear combinations of sines. We define each performance metric f_j as

$$f_j(\mathbf{x}) = \sum_{i=1}^{D} \sum_{k=1}^{K} \alpha_{i,k} \sin(k x_i + B_{i,k}), \tag{7}$$

where $\alpha_{i,k}, \beta_{i,k} \in [0,1]$ are selected uniformly at random. We ran experiments for varying numbers of design and performance parameters. The results in Figure 7 illustrate the different affine subspaces that are used to construct the Pareto fronts, denoted in different colors.

Figure 8 compares our solution to a method that first discovers Pareto-optimal points and then uses a piecewise-linear interpolation in design space. Our approach, in addition to discovering points on the front more efficiently, has the critical advantage of being faithful to the topology of the front and its relationship to the preimage in design space (the Pareto set). As shown in the figure, there is no guarantee that interpolating the preimages of two solutions adjacent in objective space (blue points) will produce another Pareto-optimal

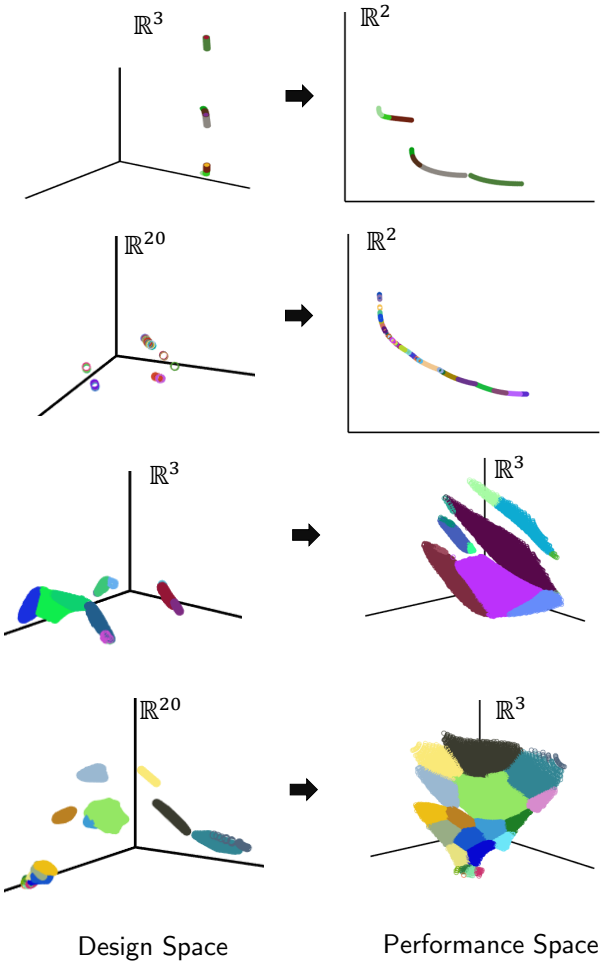

Fig. 7. Results for Pareto front discovery on Fourier benchmark. The figure illustrates how the Pareto front can be covered by a set of individually smooth regions which are mapped via F from affine subspaces in the design space. Each corresponding pair in design and performance space is illustrated in a different color. In high-dimensional cases in design space, dimensionality-reduction via principal component analysis is applied for visualization purposes.

solution (red points). Our method for generating space approximations (Section 6.4) automatically detects these discontinuities in the Pareto set.

The buffer construction and graph-cut algorithm guarantee that the results of this sparse approximation are optimal within a tolerance (δ_B) if each candidate point on $B(j)$ that has minimal distance to the origin is on the true Pareto front. This result, however, depends not only on our ability to avoid local minima, which was empirically validated on the ZDT and DTLZ benchmarks, but also on the first-order approximation. The quality of the first-order approximation depends on how well the Pareto set can be approximated by a linear function. Since the Pareto sets of the ZDT and DTLZ benchmarks are linear, we quantitatively evaluate the proposed local linear approximation on the Fourier benchmark. The result

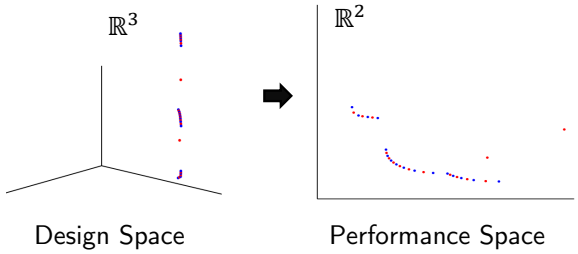

Fig. 8. A direct piecewise-linear interpolation result of the example shown on the first row of Figure 7. The optimal solution is chosen from the list of points at each buffer cell (blue points) and a denser sampling is generated by linearly interpolating the preimage of neighboring points in performance space (red points). Since the Pareto front is comprised of distinct manifolds, the linearly interpolated points have no guarantee of Pareto optimality. For illustrative purposes, the interpolation is performed on a sparse set of the discovered solutions.

is shown in Figure 9, which, illustrates how far away the optimal candidate point $\mathbf{x}_j^*$ on each buffer cell is from the actual Pareto front.

To measure this distance, we ran an additional local optimization that tries to push each point $\mathbf{x}_j^*$ the direction $\mathbf{d}_N^j$ normal to the front at $F(\mathbf{x}_j^*)$:

$$\min_{\mathbf{x}\in\mathcal{X}} \left\| F(\mathbf{x}) - \left(F(\mathbf{x}_j^*) + \delta_N \mathbf{d}_N^j \right) \right\|, \tag{8}$$

where we set $\delta_N = \delta_B = 10^{-2}$. Figure 9 shows that the approximation error (distance in performance space) for the problem showed in the the first row of Figure 7 is below 4.0×10^{-4} in the worst case and below 10^{-5} for over 97% of points. This result shows that for general functions defined by Fourier series, our method can robustly approximate the front. This is done by iteratively adding more local manifolds until the improvement on an average cell is bellow $\delta_I = 10^{-4}$ (stopping parameter). For the example in Figure 9 each buffer cell has, on average, 10.42 candidate points, which correspond to first order approximations that are within error $\delta_B = 10^{-2}$ of each other.

Finally, we show the behavior of our first-order approximation on the bi-objective Kursawe problem [Kursawe 1991], which has a disconnected, asymmetric Pareto front. Figure 10 shows the expanded curve using our algorithm (red) alongside a number of curves expanded in random directions (gray). Our generated direction in design space (black) approximates the shape of the true Pareto front in objective space better than its randomized counterparts that are not obtained using the perturbative formula.

7.2 Design Applications

We additionally run experiments on several CAD models, each with a specified set of design and performance metrics. We use three models from [Schulz et al. 2017] that are assessed using a combination of pre-computed physical simulations and geometric analysis. We also experiment with higher-dimensional design spaces using a twelve-parameter boat parametrized by cage-based deformation [Ju

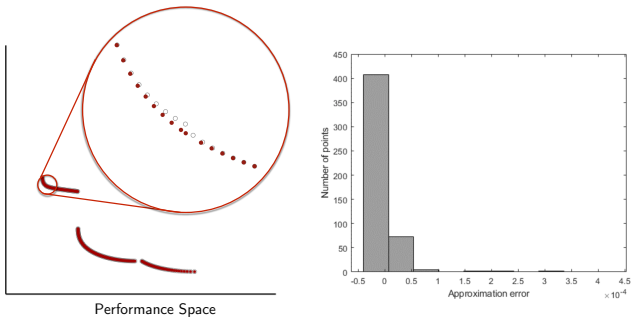

Fig. 9. Performance Space Assessment of the approximation error associated with our first-order expansion method on the example shown on the first row of Figure 7. Left: plot in performance space of the discovered Pareto front (grey) and results of the same points after performing an additional local optimization (maroon). Right: histogram showing the approximation error for points on the discovered Pareto front.

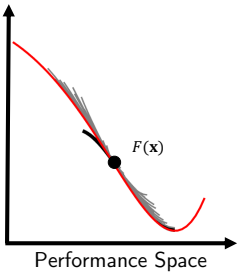

Fig. 10. First-order approximation for a point $\mathbf{x}$ on the Pareto front of the bi-objective Kursawe problem [Kursawe 1991] (true front shown in black). The first-order approximation defined by our method (red) is compared to the mapping of affine spaces around $\mathbf{x}$ generated using directions chosen uniformly at random (gray).

et al. 2005] and a twenty-one-parameter lamp designed in CAD. Figures 1 and 11 show the resulting solutions for each of these experiments in both design space and performance space, along with the corresponding embedding and (selected) mesh samples. We also implement a simple interface that provides an interactive visualization of geometries corresponding to each point in the embedding (see the supplemental video).

The top row of Figure 11 shows a wrench example with three design parameters: handle length, head thickness, and fillet radius (the rounding along the edges that connects the head to the handle). The measured performance metrics are stress for a given torque (maximum von Mises stress computed with FEA, implementation from [Schulz et al. 2017]), mass, and force required to generate a given torque. In this example, most of the front can be expressed as a single affine space where the fillet radius and the length are maximized. This result is due to the fact that these parameters have a large impact on minimizing the stress and force for a given torque, while the negative impact on the mass is largely negligible. Only in a very small region of the Pareto front do solutions corresponding to a small fillet radius appear. For these regions, the head and fillet radii are minimized and the optimal trade-offs between mass and force/torque are achieved by varying the length of the handle. While the design space for this model is low-dimensional (only three variables), it highlights the strength of our method in exposing relationships between design parameters and performance metrics that are not evident without analysis. Furthermore, it allows us to simplify the solution space by understanding that there are only two affine regions in design space which yield Pareto-optimal performance trade-offs.

The second row of Figure 11 shows a brake hub with three design variables: the angle of the inner hole, spoke thickness, and the thickness of the rim. The performance metrics for this model are stress (calculated by simulating the impact of directional forces and heat distribution), mass, and heat dissipation (approximated by the thickness of the rim). Our method highlights a strong discontinuity in design space represented by the black line that divides the green patch. Figure 12 shows two models on opposite sides of this divide. We observe that, while the performance parameters vary smoothly across this gap, the spoke thickness almost doubles. This result exposes a property of the measured stress which depends on both the heat distribution and directional forces from the brake; the reason why two very different designs can be so close in performance space is that the spoke thickness affects the heat distribution and the force-imposed object deformation in opposite ways. Our method exposes these different design configurations which, in fact, yield comparable performance results, providing engineers with a better understanding of the model's properties.

The third row of Figure 11 illustrates a bike frame with four design variables that has been engineered to minimize mass, drag, and stress. What is interesting about this result is that the embedding is composed of a set of disconnected regions. This happens because of the geometry of the envelope of $F(\mathcal{X})$. Assuming that F is continuous and $\mathcal{X}$ is connected, the projection of $F(\mathcal{X})$ onto the buffer is also always connected. As previously discussed, however, not all points represented by the buffer are Pareto-optimal. For this example in particular, after we remove the suboptimal point from the buffer, we are left with a large empty region in the center of the embedding. These discontinuities in performance space reveal regions of trade-off where a small variation in design space will only slightly worsen one metric but significantly improve another. Such exposed properties can be very useful in aiding designers to decide between certain success metrics and define trade-offs.

The fourth row of Figure 11 illustrates a toy boat parameterized by a cage with twelve design variables and two performance metrics. The performance metrics are buoyancy (approximated by volumetric maximization) and drag from a frontal wind. This example shows how our method can find a locally smooth approximation of the Pareto front for high-dimensional design space. The resulting boats all have maximal length but the shape of the projection onto a frontal plane varies between solutions that represent different trade-offs between mass and drag.

Finally, Figure 1 illustrates a lamp with 21 parameters that was designed using a CAD package. Three parameters are used to define the position and orientation of each of the lamp's seven beams. The performance metrics for this model are: stability (measured by the distance of the projection of the center of mass to the center of the

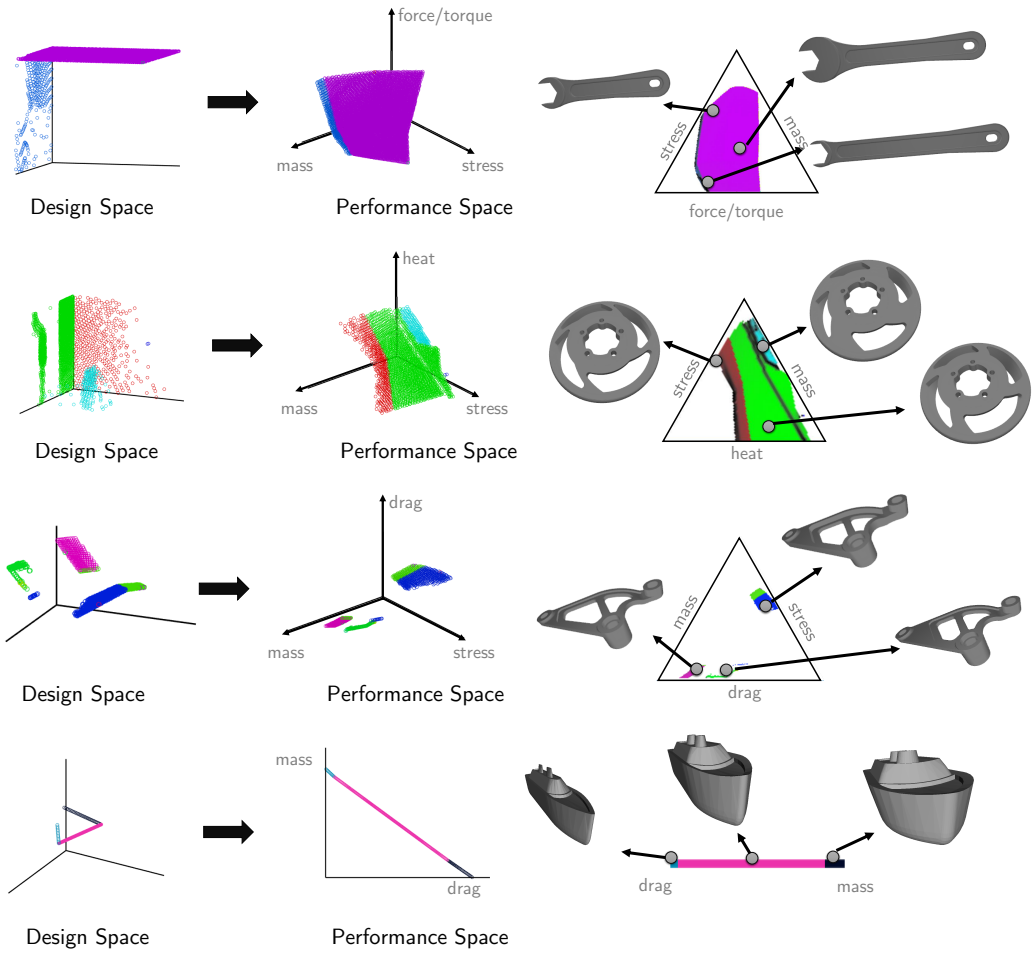

Fig. 11. Examples of CAD models processed using our proposed technique. From left to right: Pareto-optimal points in design space (illustrated with multi-dimensional scaling via principal component analysis for models with more than three design parameters); Pareto-optimal points in performance space; the resulting embedding and illustrations of geometric results for some sampled points. Across all figures, the different colors correspond to different regions resulting from local expansion in design space.

base), mass, and average distance between the lamp beams and a predetermined focal point that should be illuminated. As shown in the figure, our method returns an approximation of the Pareto front with many disconnected regions in the design space. This behavior arises due to the presence of many points in the design space that are mapped to the same regions on the Pareto front (i.e., there are many points in the Pareto front which can be locally approximated by manifolds of higher dimension than $d-1$). An example of this is shown in Figure 13. This example highlights the utility of creating a sparse representation of the Pareto front with local manifolds. Since multiple points map to the same point on the Pareto front, a discrete approach would result in an approximation containing many disparate points across the design space. This discontinuous representation, however, provides little in the way of performance trade-off insights. Our method, on the other hand, creates locally smooth regions around areas of similar geometry. This configuration in turn allows designers to directly observe which parameters have a significant effect on local performance and which ones do not, a particularly useful feature in higher-dimensional cases.

8 CONCLUSION

Real-world design problems can rarely be squeezed into one dimension. Instead, the process of engineering a physical object requires navigating a complex and potentially even disconnected space of candidate configurations. The algorithm and accompanying interactive tool presented in this paper represent a significant effort toward the larger goal of efficiently estimating and navigating the space of relevant designs for a given problem. Our technique efficiently reveals this space in a wide variety of scenarios, from benchmarks in multi-objective optimization to parameterized CAD models paired with expensive physical simulation tools. Its versatility indicates

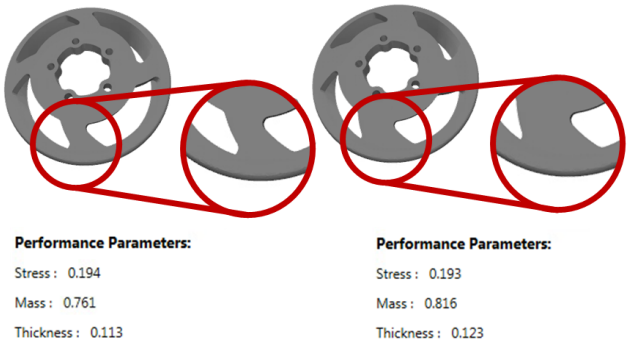

Fig. 12. Example of two variations of the brake hub that have similar performance metrics but very different design parameters. Our system can expose these relationships providing intuition to designers about the structural nature of the model. The metrics are shown under a normalization for easy comparison—the Pareto front is rescaled to lie between zero and one.

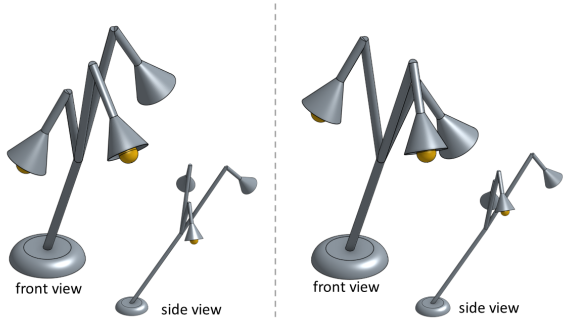

Fig. 13. Example of two different lamps (front and side view) that have the same performance across all metrics. Due to the large dimensionality of the design space, these two configurations have the same stability, mass, and distance to the focal point.

broad applicability across use cases in computational fabrication and beyond.

While our algorithm as-is can be plugged into many existing engineering pipelines, we also anticipate several avenues of research that can extend the basic model proposed here. One important consideration comes from the human-computer interaction. Now that we can efficiently uncover and parameterize the Pareto front, what is the best way to display it to an engineer who must digest the space of candidate designs? Such a study can use our tool as a starting point, adjusting e.g. the embedding of the performance buffer to best reflect intuitive notions of proximity in performance space. On the opposite side of the spectrum between human interaction and automation, we also anticipate that our differentiable representation of the Pareto front can be incorporated into "hyperparameter" selection techniques that use a secondary function to choose between different points on the Pareto front.

Our study also suggests several intriguing mathematical and algorithmic challenges. While our benchmark study indicates that our algorithm yields a smooth and complete picture of the Pareto front—at least in realistic scenarios where the front is representable computationally—additional theoretical analysis could reveal convergence rates and/or the likelihood that our strategy will reveal the entire Pareto front. Of course, such theoretical analysis will probably require stronger assumptions on the performance metrics than are needed in practice, e.g. convexity or Lipschitz bounds, to rule out pathological cases; the challenge will be to select a theoretical model that is reflective of the scenarios we observe in our examples from CAD and engineering design. Further analysis could also extend the first-order approximation to handle cases in which the derivatives of the active constraints at a given point are themselves linearly dependent. Algorithmically, a clear next-step will be to extend our methodology to the regime where parameters are discrete or performance measures are not twice differentiable.

An additional challenge for the future spanning both technical and human-oriented aspects is to cope with higher dimensionalities for performance space. Currently our technique is designed for the $d = 2$ and $d = 3$ cases, for which the Pareto front is readily embedded in a display tool. Considering larger values of d will require several technical developments. For the sampling algorithm, the "curse of dimensionality" implies that sampling may take more iterations to converge; we anticipate that our manifold-based strategy will serve as a key component reducing computational burden in this regime by exploiting local structure to discover larger pieces of the front at a time. After uncovering the higher-dimensional Pareto front, we will also need to design a means of displaying the result in a fashion similar to Figure 11. While MDS embedding of the Pareto front may suffice, a more careful parameterization that preserves relevant relationships between the performance metrics may be desirable.

Even in the absence of these improvements, we anticipate incorporation of our exploration algorithm and visualization tool into software for CAD and 3D modeling. By helping engineers and designers understand the possible ways to trade off between performance metrics, we hope to alleviate the dependence on unintuitive and often brittle parameters currently permeating design software.

A PROOF OF PROPOSITION 5.1

Since $\mathbf{x}(t) \in \mathcal{P}$ for all $t \in (-\varepsilon, \varepsilon)$, each point $\mathbf{x}(t)$ must satisfy the KKT conditions (2). Hence, we can assume the existence of time-varying dual variables $\alpha(t) : (-\varepsilon, \varepsilon) \to \mathbb{R}^d$ and $\beta(t) : (-\varepsilon, \varepsilon) \to \mathbb{R}^K$. Generically these functions are differentiable in t.

For a given critical point $\mathbf{x}^* = \mathbf{x}(0)$, without loss of generality permute the constraints so that the first K' inequality constraints are active, i.e. $g_k(\mathbf{x}^*) = 0$ for all $k \leq K'$, and that the remaining constraints are inactive, i.e. $g_k(\mathbf{x}^*) < 0$ for all $k > K'$.

Applying the complementary slackness condition in (2), we must have $\beta_k(0) = 0$ for all inactive constraints $g_k(\mathbf{x}^*)$. By continuity, if a constraint is inactive at $t = 0$ it must remain inactive in a nonempty open interval surrounding $t = 0$. After possibly restricting ε, we can assume $\beta_k(t) \equiv 0$ for all $k > K'$ and $t \in (-\varepsilon, \varepsilon)$.

Collecting our observations so far, we rewrite the KKT conditions (2) as follows:

$$\left\{\begin{array}{l} \alpha_i(t) \geq 0 \;\; \forall i \in \{1,\ldots,d\}, t \in (-\varepsilon,\varepsilon) \\ \beta_j(t) \geq 0 \;\; \forall j \in \{1,\ldots,K'\}, t \in (-\varepsilon,\varepsilon) \\ \beta_j g_j(\mathbf{x}(t)) = 0 \;\; \forall j \in \{1,\ldots,k\}, t \in (-\varepsilon,\varepsilon) \\ \sum_{i=1}^{d} \alpha_i(t) = 1, t \in (-\varepsilon,\varepsilon) \\ \sum_{i=1}^{d} \alpha_i(t)\nabla f_i(\mathbf{x}(t)) + \sum_{j=1}^{K'} \beta_j(t)\nabla g_j(\mathbf{x}(t)) = 0 \;\; \forall t \in (-\varepsilon,\varepsilon) \end{array}\right\} \tag{9}$$

Note this form effectively ignores the inactive constraints since they do not figure into the problem near $t = 0$.

Our next task is to differentiate the final condition in (9) with respect to t at $t = 0$. Define:

$$h(t) := \sum_{i=1}^{d} \alpha_i(t)\nabla f_i(\mathbf{x}(t)) + \sum_{j=1}^{K'} \beta_j(t)\nabla g_j(\mathbf{x}(t))$$

Then,

$$h'(t) = DF^\top(\mathbf{x}(t))\alpha'(t) + \left(\sum_{i=1}^{d} \alpha_i(t) H_{f_i}(\mathbf{x}(t))\right)\mathbf{x}'(t) + DG^\top(\mathbf{x}(t))\beta'(t) + \left(\sum_{k=1}^{K'} \beta_k(t) H_{g_k}(\mathbf{x}(t))\right)\mathbf{x}'(t)$$

Here, Du indicates the Jacobian and H_u indicates the Hessian of a function $u(\mathbf{x})$.

Evaluating at $t = 0$ and recalling $\mathbf{x}(0) = \mathbf{x}^*$ shows

$$h'(0) = DF^\top(\mathbf{x}^*)\alpha'(0) + \left(\sum_{i=1}^{d} \alpha_i(0) H_{f_i}(\mathbf{x}^*)\right)\mathbf{x}'(0) + DG_{K'}^\top(\mathbf{x}^*)\beta'(0) + \left(\sum_{k=1}^{K'} \beta_k(0) H_{g_k}(\mathbf{x}^*)\right)\mathbf{x}'(0)$$

We use $DG_{K'}$ to denote the part of the Jacobian of G corresponding to active constraints. Defining $\mathrm{H} := \sum \alpha_i(0) H_{f_i}(\mathbf{x}^*) + \sum \beta_k(0) H_{g_k}(\mathbf{x}^*)$ allows us to simplify our expression to

$$h'(0) = DF^\top(x^*)\alpha'(0) + DG_{K'}^\top(x^*)\beta'(0) + \mathrm{H}\mathbf{x}'(0). \tag{10}$$

From the KKT conditions, we know $h(t) \equiv 0$—and hence $h'(t) \equiv 0$—for all $t \in (-\varepsilon, \varepsilon)$. Rearranging slightly shows

$$H\mathbf{x}'(0) \in \mathrm{Im}(DF^\top(\mathbf{x}^*)) \oplus \mathrm{Im}(DG_{K'}^\top(\mathbf{x}^*)). \tag{11}$$

We obtain an additional property of $\mathbf{x}'(0)$ by revisiting the complementary slackness condition in (9), which shows $\beta_k(t) g_k(\mathbf{x}(t)) \equiv 0$ for $t \in (-\varepsilon, \varepsilon)$ and $k \in \{1,\ldots,K'\}$. Again differentiating both sides with respect to t shows

$$0 = \beta_k'(t) g_k(\mathbf{x}(t)) + \beta_k(t)\nabla g_k(\mathbf{x}(t))^\top \mathbf{x}'(t).$$

Recall $g_k(\mathbf{x}^*) = 0$ since constraint k is active; we furthermore can assume $\beta_k(0) \neq 0$ since the constraint is active. Hence the first term vanishes and at $t = 0$ we are left with

$$\nabla g_k(\mathbf{x}^*)^\top \mathbf{x}'(0) = 0 \;\forall k \in \{1,\ldots,K'\}, \tag{12}$$

Combining different k's shows

$$DG_{K'}(\mathbf{x}^*)\mathbf{x}'(0) = 0,$$

as desired.

ACKNOWLEDGMENTS

The authors would like to thank Desai Chen for helpful suggestions and discussions. This work is supported by the Defense Advanced Research Projects Agency, under grant N66001-15-C-4030 and the National Science Foundation, under grant CMMI-1644558 and grant IIS-1409286. J. Solomon acknowledges the generous support of Army Research Office grant W911NF-12-R-0011 ("Smooth Modeling of Flows on Graphs"), from the MIT Research Support Committee ("Structured Optimization for Geometric Problems"), and from the Skoltech–MIT Next Generation Program ("Simulation and Transfer Learning for Deep 3D Geometric Data Analysis").

REFERENCES

Shailen Agrawal and Michiel van de Panne. 2013. Pareto Optimal Control for Natural and Supernatural Motions. (2013).

Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. 2015. LinkEdit: Interactive Linkage Editing Using Symbolic Kinematics. *ACM Transactions on Graphics* 34, 4 (July 2015), 99:1–99:8.

Sunith Bandaru and Kalyanmoy Deb. 2015. Temporal innovization: Evolution of design principles using multi-objective optimization. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics))*, Vol. 9018. Springer Verlag, 79–93.

Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. 2015. Computational Design of Metallophone Contact Sounds. *ACM Transactions on Graphics* 34, 6 (Oct. 2015), 223:1–223:13.

Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. 2010. Design and Fabrication of Materials with Desired Deformation Behavior. *ACM Transactions on Graphics* 29, 4, Article 63 (July 2010), 10 pages.

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23, 11 (2001), 1222–1239.

Indraneel Das and J. E. Dennis. 1998. Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization* 8, 3 (1998), 631–657.

Kalyanmoy Deb and Kalyanmoy Deb. 2014. Multi-objective Optimization. In *Search Methodologies*. Springer US, Boston, MA, 403–449.

Kalyanmoy Deb and Aravind Srinivasan. 2006. Innovization: Innovating Design Principles Through Optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06)*. ACM, New York, NY, USA, 1629–1636.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. 2002. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC 2002)*. IEEE Press, 825–830.

Yue Dong, Jiaping Wang, Fabio Pellacini, Xin Tong, and Baining Guo. 2010. Fabricating Spatially-varying Subsurface Scattering. *ACM Transactions on Graphics* 29, 4, Article 62 (July 2010), 10 pages.

Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational Multicopter Design. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 227:1–227:10.

Claus Hillermeier. 2001. *Nonlinear multiobjective optimization: a generalized homotopy approach*. Vol. 135. Springer Science & Business Media.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 561–566.

Frank Kursawe. 1991. A Variant of Evolution Strategies for Vector Optimization. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature (PPSN I)*. Springer-Verlag, London, UK, UK, 193–197.

Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-printable Robotic Creatures. *ACM Transactions on Graphics* 34, 6 (Oct. 2015).

A. Messac, A. Ismail-Yahaya, and C.A. Mattson. 2003. The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization* 25, 2 (01 Jul 2003), 86–98.

Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2015. Reduced-order Shape Optimization Using Offset Surfaces. *ACM Transactions on Graphics* 34, 4 (July 2015).

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Transactions on Graphics* 32, 4 (July 2013), 81:1–81:10.

Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. 2017. Retrieval on Parametric Shape Collections. *ACM Transactions on Graphics* 36, 1 (Jan. 2017), 11:1–11:14.

Maria Shugrina, Ariel Shamir, and Wojciech Matusik. 2015. Fab Forms: Customizable Objects for Fabrication with Validity and Geometry Caching. *ACM Transactions on Graphics* 34, 4 (July 2015), 100:1–100:12.

Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. 2012. Guided Exploration of Physically Valid Shapes for Furniture Design. *ACM Transactions on Graphics* 31, 4 (2012).

Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes. *ACM Transactions on Graphics* 33, 4 (July 2014), 65:1–65:10.

Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. 2012. Structural Optimization of 3D Masonry Buildings. *ACM Transactions on Graphics* 31, 6 (2012), 159:1–159:11.

Siu-Man Yau, Eitan Grinspun, Vijay Karamcheti, and Denis Zorin. 2006. Sim-X: parallel system software for interactive multi-experiment computational studies. In *20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece.*

M. Zeleny. 1973. Compromise Programming. In *Multiple Criteria Decision Making*, J. Cochrane and M. Zeleny (Eds.). University of South Carolina Press, Columbia, 262–301.

J. Zhang and L. Xing. 2017. A Survey of Multiobjective Evolutionary Algorithms. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Vol. 1. 93–100.

Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation* 11, 6 (2007), 712–731.

Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation* 1, 1 (2011), 32–49.

Bo Zhu, Mélina Skouras, Desai Chen, and Wojciech Matusik. 2017. Two-Scale Topology Optimization with Microstructures. *ACM Transactions on Graphics* 36, 5, Article 164 (July 2017), 16 pages.

E. Zitzler, K. Deb, and L. Thiele. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation* 8, 2 (2000), 173–195.