OPTIMISM: Enabling Collaborative Implementation of Domain Specific Metaheuristic Optimization

Megan Hofmann
Khoury College of Computer Science, Northeastern University
Boston, MA, USA

Nayha Auradkar
Jessica Birchfield
Jerry Cao
Autumn Hughes
Paul G. Allen School of Computer Science, University of Washington
Seattle, WA, USA

Gene Kim
Stanford University
Stanford, CA, USA

Shriya Kurpad
Kathryn Lum
Kelly Mack
Paul G. Allen School of Computer Science, University of Washington
Seattle, WA, USA

Anisha Nilakantan
Human Computer Interaction Institute, Carnegie Mellon University
Pittsburgh, PA, USA

Margaret Ellen Seehorn
Grinnell College
Grinnell, IA, USA

Emily Warnock
Jennifer Mankoff
Paul G. Allen School of Computer Science, University of Washington
Seattle, WA, USA

Scott E. Hudson
Human Computer Interaction Institute, Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

For non-technical domain experts and designers it can be a substantial challenge to create designs that meet domain specific goals. This presents an opportunity to create specialized tools that produce optimized designs in the domain. However, implementing domain-specific optimization methods requires a rare combination of programming and domain expertise. Creating flexible design tools with re-configurable optimizers that can tackle a variety of problems in a domain requires even more domain and programming expertise. We present OPTIMISM, a toolkit which enables programmers and domain experts to collaboratively implement an optimization component of design tools. OPTIMISM supports the implementation of metaheuristic optimization methods by factoring them into easy to implement and reuse components: objectives that measure desirable qualities in the domain, modifiers which make useful changes to designs, design and modifier selectors which determine how the optimizer steps through the search space, and stopping criteria that determine when to return results. Implementing optimizers with OPTIMISM shifts the burden of domain expertise from programmers to domain experts.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3580904

CCS CONCEPTS

• **Human-centered computing**;

KEYWORDS

generative design; metaheuristic; optimization; toolkit

ACM Reference Format:
Megan Hofmann, Nayha Auradkar, Jessica Birchfield, Jerry Cao, Autumn Hughes, Gene Kim, Shriya Kurpad, Kathryn Lum, Kelly Mack, Anisha Nilakantan, Margaret Ellen Seehorn, Emily Warnock, Jennifer Mankoff, and Scott E. Hudson. 2023. OPTIMISM: Enabling Collaborative Implementation of Domain Specific Metaheuristic Optimization. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3544548.3580904

1 INTRODUCTION

Optimization methods can help users generate unique solutions while considering a variety of domain specific goals, but building optimizers requires the expertise and collaboration of programmers and domain experts who tailor these methods to the domain. This complex collaboration between programmers and domain experts is critical, but under-supported. Existing systems tend to focus on supporting the programmer by offering off-the-shelf implementations of standard and advanced optimization methods. However, the process of incorporating domain expertise to tailor these methods to a domain is left to the development team. There are opportunities to support the implementation of domain specific optimizers by enabling collaboration between domain experts and programmers.

Table 1: Summary of optimization methods distributed across literature survey. See Appendix A

Venue	Percentage of Total Papers at Venue							Paper
	Bayesian	Constraint Satisfaction	Topology	Convex	Stochastic	Heuristic	Metaheuristic	Count
CHI [3]	11.1	11.1	7.4	14.8	7.4	25.9	22.2	27
SCF [8]	0.0	0.0	33.3	16.7	0.0	33.3	16.7	6
TOG [2]	3.6	3.6	7.1	64.3	0.0	7.1	14.3	28
UIST [9]	0.0	9.1	0.0	9.1	9.1	36.4	36.4	11
Paper Count	**4**	**5**	**6**	**24**	**3**	**15**	**15**	**72**
Percentage of Total Papers	**5.6**	**6.9**	**8.3**	**33.3**	**4.2**	**20.8**	**20.8**	**100.0**
Example Paper	[80]	[100]	[153]	[108]	[43]	[11]	[39]	

We present the Optimization Programming Toolkit Integrating Metaheuristic Intuitive Search Methods. OPTIMISM helps non-technical *domain experts* and *programmers* collaboratively implement optimizers in diverse domains. *Designers*, who have domain expertise but do not program, access *optimizers* through an automatically generated GUI. OPTIMISM is domain agnostic and deconstructs many metaheuristic methods into a small set of pluggable operations called *objectives* and *modifiers*. These components help domain experts express their goals and modification strategies. OPTIMISM provides a domain agnostic library of pluggable components that help programmers rapidly prototype domain specific optimizers that apply objectives and modifiers.

We have developed OPTIMISM based on three principles. First, OPTIMISM empowers domain experts to participate in optimizer implementation. Second, OPTIMISM enables programmers to flexibly experiment with a variety of optimization methods with minimal additional coding. Third, OPTIMISM produces satisfactory and sufficient optimizers. OPTIMISM serves these principles through a generalized and simple framework that can implement a wide variety optimization methods. Rather than offering a one size fits all solution, OPTIMISM tailors to the unique properties of a domain.

We demonstrate five tools to show how OPTIMISM: (1) supported an Ophthalmologist and programmer in collaboratively building a cataract lens selection tool, (2) enabled us to derive a thumb splint optimizer from the domain expertise of occupational therapists', (3) assists blind designers in creating satisfactory and useful customized tactile graphics, (4) enables us to flexibly experiment with different optimizers that replicate an existing generative design tool [32], and (5) amplifies knitters' domain expertise.

2 RELATED WORK

Optimization methods have the the potential to make complex design tasks accessible to new users (e.g., people with disabilities [96, 143], clinicians [66, 69, 83]) because they can generate solutions that are tuned to fit highly specific needs (e.g., customized medical devices). While individually, these optimization methods can be simpler to develop, they are often out of the reach of domain experts who cannot program but would benefit from these tools. Further, optimization methods are a critical backbone to the growing domain of generative design for digital fabrication. In this section, we examine how optimization has been used in digital fabrication, the benefits and limitations of different optimization methods in these domains, and existing toolkits that support implementation of optimizers.

2.1 Optimization in Fabrication

Researchers have explored a wide range of digital fabrication optimization problems (e.g., fabricating "surface like objects" [32], balancing 3D models [20, 72, 116], improving model strength [128, 142], or generating deformable mechanisms [18, 22, 38, 99, 115, 148]). We analysed 210 research articles from four research venues relevant to human computer interaction and digital fabrication published between 2016 and 2021 that included author keywords related to digital fabrication (e.g., fabrication, 3D printing, laser cutting) or optimization (e.g., optimize, inverse design, generative design). We then narrowed our analysis to the 72 papers that contributed a optimization method for digital fabrication. We excluded papers that do not use an optimization method or described a generalized toolkit related to fabrication or optimization. We inductively categorized the broader categories of optimization used in this body of work (Table 1). Optimization methods depend on properties of the search space. For example, convex optimization methods are desirably efficient and tend to produce quality local-maxima but rely on domains being represented as a convex search space. Many non-technical domain experts struggle to understand domains in this way[15]. Similarly, constraint satisfaction methods require developers to express a domain as a solvable set of equations.

Even when the search space is not amenable to these methods, the diversity of viable, but potentially ill-suited, optimization methods is overwhelming. In poorly characterized domains, we can turn to heuristic, stochastic, and metaheuristic methods with make up 45.8% of the methods used in our survey. Heuristic methods (e.g., [29, 47, 51, 94, 119, 134, 135]) apply information about the domain to guide the search. Blum and Roli [24] call heuristic methods "intense" search methods because they narrowly apply domain specific strategies to find local maxima. Alternatively, stochastic methods like Monte Carlo methods [43], use randomized search patterns to jump over local maxima. Blum and Roli [24] call these methods "diverse" because they sample widely from the search space.

Metaheuristic methods combine heuristics and stochastic methods to create high-level, problem-agnostic strategies to guide a localized search process [129]. Different metaheuristics control the "intensity" and "diversity" [24] of the optimization method with: "intense" heuristic based search on one end of the spectrum and "diverse" random search methods on the other. Metaheuristic optimization trades speed and guarantees of optimality for flexibility. These methods work best in domains with where objectives can be clearly defined and a variety of simple strategies for improving designs (i.e., heuristics), but which strategy should be applied in any given

iteration is unclear. Essentially, metaheuristic methods are effective at sampling a wide variety of search strategies—doing the manual work of a designer but much faster. Like manual design, metaheuristic methods do not guarantee optimality or convergence on a solution. Despite this, these methods have been effective in a wide variety of fabrication domains (e.g., [13, 39, 43, 57, 68, 70, 87, 155]).

2.2 Multi-Objective Optimization

Often optimization involves multiple, conflicting objectives. Instead of reducing complex design tasks to one objective function, multi-objective optimization (MOO) methods allow users to explore the trade offs between objectives. MOO frames optimization problems as a design space defined by design parameters and a vector of objectives that map designs to an objective-space. The outer boundary of the objective space, where every design on the boundary cannot improve an objective without worsening another, is called the Pareto front. Notably, the size of the Pareto front expands exponentially relative to the number of objectives, making exploration of the Pareto front intractable for many high-dimension problems. Methods for finding the Pareto front is a growing area of research; like optimization methods more broadly, no method works well or guarantees optimality in all domains. The principle challenge of building a domain specific optimizer is to find an optimization method that is well matched to the domain and objective space and users need tools to help them create these unique optimizers.

To find the Pareto front, an optimization method must explore the design space by finding as many sample designs as possible that maximize different combinations of objectives. There are two main approaches to searching the design space [42]. The first are methods that decompose the search into many single-objective optimization problems [139] by either *scalarizing* (i.e., weighting and combining each objective) (e.g., weighted-sum method [97]) or constraining objectives (e.g., ϵ-constraint methods [59]). Through many runs of these single-objective optimizers, the multi-objective optimizer can sample new regions of the Pareto front. The second set of methods build up the population of discovered designs without being guided by a specific objective function using recombination and mutation of the population (e.g., evolutionary methods [37]). Particularly in discrete domains, both decomposition and population methods may apply metaheuristics when generating new results that select for some combination of: Pareto dominant designs, designs that dominate decomposed objective functions, and indicators of improvement [28, 90]. In many real world problems, where the number of objectives may be high (e.g., greater than three), the methods may be guided towards high priority regions of the Pareto front either by interaction with the user (e.g.,[121]) or by Bayesian methods [53].

Given a Pareto front, users are challenged to choose a solution. One approach is to only search for the section of the Pareto front that is relevant to the designer by narrowing the search with a priori rankings of objectives (e.g., user-set weights in scalarization) [103]. Depending on the domain, the user may not be able to provide quality weights and this can result in designs that do not meet their needs. However, these approaches one or only a few solutions making it trivial for users to select a solution if a satisfactory solution is discovered. Alternatively, a posteriori methods search for a wide region of the Pareto front and then ask users to make decisions based on discovered trade-offs between objectives [103]. Particularly with many objectives, this may result in the user having to choose from many possible solutions. Researchers have proposed a variety of visualization tools to help users make decisions (e.g., [78, 92, 131, 138]), however each of these visualizations presume that the designer understands key optimization concepts. For example, Smedberg and Bandaru's interactive knowledge discovery tools presumes that designers understand that optimization methods imply a mapping from a design space to an objective space [126]. In many relevant domains this assumption may not hold; for example, blind designers cannot access such visualizations for decision making [67] and prior work shows that clinicians prefer clinical CAD tools to follow a prescriptive model that generates only one effective solution [69]. Again, method selection depends on the domain and the needs and preferences of designers.

2.3 Toolkits to Support Optimization

To evaluate optimization toolkits, we can consider criteria described by Olsen [111]: "*flexibility*" to rapidly making design changes; "*expressive leverage*", accomplishing "*more by expressing less*"; "*expressive match*" of the toolkit model to the user's mental model of the problem; and "*ease of combination*" of simple primitives into a wide set of complex solutions. Krish describes optimization as consisting of: a search space or domain, a way of generating variation in that domain, and a method for evaluating designs in that domain [82]. A toolkit for optimization, ideally, gives developers flexible ways to express these components in a way that matches their mental model of the domain. Most optimization toolkits are inflexible and only offer established optimization algorithms, rather than recombinable primitives that enable programmers to prototype their own solutions. Consider, toolkits that support metaheuristic optimization (e.g., [5–7]), MOO (e.g., [23]), or optimization more broadly (e.g., [1, 4]). These systems are inflexible; limited to expressing existing algorithms. Further, their expressive match is dependent on how well standardized format fit users' mental models. These tools are designed for experienced programmers rather than domain experts.

Toolkits in the space of convex optimization [27] and Bayesian optimization [56] offer a promising approach. Burnell et al. developed GPKit [27] with an ethnographic study of experts who use convex optimization. The toolkit supports highly flexible development of a convex expression of a design space that has an expressive match with their users. These users have broader technical backgrounds but do not included domain experts who have little mathematical or programming experience. Similarly, Golovin et al's Google Vizier [56] helps programmers tune Bayesian optimization methods to solve new problems where characteristics of the domain space are unknown (e.g., cookie recipes, GUIs). Vizier is a highly flexible framework and can be leveraged to express a variety of black-box optimization problems. Our goal is to contribute to the same space of toolkits by focusing on heuristic methods, stochastic methods, and metaheuristic methods. By combining the expressive match of heuristic methods and the flexibility of stochastic methods, we aim to support new users in a wide set of domains.

3 OPTIMISM TOOLKIT

OPTIMISM[1] enables domain experts and programmers collaboratively implement a domain specific optimizer and automatically produce a GUI for designers. The domain expert and programmer provide OPTIMISM with domain specific heuristic libraries that express design strategies in the domain. These heuristics are iteratively applied by metaheuristic optimizers which programmers configure with elements of OPTIMISM's domain agnostic library. In the following sections, we describe OPTIMISM's: three guiding principles, three types of users, framework structure, and how it is used. We contextualize the system with the scenario of creating a cookie optimizer based on heuristics from a cooking show [102].

3.1 Guiding Principles

Empowerment of Domain Experts. Giving domain experts the flexibility to describe the domain and search strategies in a GUI.

Flexible Optimizer Construction. Optimizers can be reconfigured through modular, easy to understand, components.

Produce Satisficing Designs. Optimizers efficiently (e.g., faster than manual design) produce satisfactory and sufficient (e.g., equivalent to or better than manual results) designs [26].

3.2 OPTIMISM's Three Users

We distinguish between three types of users: domain-experts, programmers, and designers . In a development stage, a programmer and domain expert collaboratively develop a domain specific optimizer by refining design representations and testing different optimizer configurations. In the design phase, independent of the domain expert and programmer, the designer uses this optimizer to produce designs that satisfy their needs. Notably, we do not assume that any of these users are familiar with optimization concepts. Programmers are only expected to have basic programming experience (e.g., implementing classes and functions) and neither domain-experts nor designers need to program. Instead, they must have relevant knowledge from a domain such as an understanding of relevant design variables, goals, and their relationships. Domain-experts and designers have the same backgrounds and are only distinguished by the phase they participate in.

3.3 System Overview and Definitions

We have developed simple abstractions that can capture metaheuristic optimization methods. OPTIMISM optimizers consist of pluggable components (Table 2) from a domain specific heuristic library and a domain agnostic library. The optimizers are seeded with *design representations* and then follow an iterative optimization process (Algorithm 1). First, these designs are scored by an *objective function* which domain experts and designers construct from domain specific *objectives*. The evaluated designs enter a *design population* which has a limited capacity set by the programmer. As that capacity is exceeded, random designs, with a bias towards poor performing designs, are pruned from the population. Second, a domain agnostic *design selector* chooses a design from the population for the next iteration. Then a domain agnostic *modifier selector* chooses a domain specific *modifier* to apply to the selected design. The modifier changes the design, producing a new design which is evaluated and added to the population; modifiers step through the design space. When the population of designs meets a *stopping criteria* it is returned to the designer. Otherwise, the cycle repeats.

[1]Framework available at https://github.com/mhofmann-Khoury/optimism-toolkit

Input *seeds*: a set of starting designs provided by designers
Output $\mathbb{D}$: the population of generated designs
$\mathbb{D} \leftarrow \{\}$;
for $d \in seeds$ **do** // Evaluate Seed Designs
  $s_d \leftarrow evaluate(d)$;
  add d to $\mathbb{D}$ sorted by s_d;
end
while *not* $stop(\mathbb{D})$ **do** // Main Optimization Loop
  $d \leftarrow select\_design(\mathbb{D})$;
  $m \leftarrow select\_modifier(d, \mathbb{H}, \mathbb{D})$;
  $d' \leftarrow m(d)$;
  $s_{d'} \leftarrow evaluate(d')$;
  add d' to $\mathbb{D}$ sorted by $s_{d'}$;
  prune $\mathbb{D}$ to population_cap;
end
return $\mathbb{D}$;

Algorithm 1: The iterative structure of an optimizer.

In the following sections, we describe the components of an optimizer organized by their source libraries. OPTIMISM's abstractions help to minimize the workload and level of technical expertise needed from the domain expert. Many aspects of OPTIMISM are domain agnostic allowing them to be readily reused and swapped out when creating an optimizer without input from a domain expert. A key exception is the heuristic library which organizes domain specific code into easy to implement and modular components.

4 DOMAIN SPECIFIC HEURISTIC LIBRARY

Domain experts and programmers implement the domain specific components of their optimizer and organize them in a heuristic library. The domain expert first describes how to represent designs and the programmer creates a corresponding *design representation*. OPTIMISM does not require a specific structure for representations since they are only accessed by objectives and modifiers implemented by the programmer. In most of our demonstrations, a simple set of parameters was a sufficient representation. For example, Chef Alton describes cookies as recipes and Steve implements a Python class with a parameter for each ingredient amount.

Next, the domain expert describes how to evaluate low-level design goals. Continuing the example, Chef Alton describes ways of estimating key properties of a recipe (e.g., melting-point, acidity) that will affect the texture of the cookie. Adjusting these properties will help Alton design different cookie textures; for instance he notes that crispy cookies have "a relatively low melting temperature so the batter spreads before setting" [102]. Each of these properties is defined by proportions of key ingredients which Steve can evaluate with simple functions. The domain expert also describes how they modify designs. For example, Alton explains how to adjust the proportions of ingredients to modify different key properties and Steve can implement this as adjustments to ingredient amounts.

Table 2: A summary of components of an OPTIMISM optimizer.

Term	Definition	Notation	Example
Domain Specific Heuristic Library	Library of domain specific elements implemented by domain experts and programmers.		
Design Representation	A data structure that represents designs in a domain	d	A set of parameters that define a cookie recipe
Objective	A function that evaluates how well a design meets a specified criteria or design goal	$0 \leq o(d) \leq 1 \forall o \in \mathbb{O}$	A function that compares a cookie recipe's estimated melting point to a target value set by a designer.
Objective Function	A function that evaluates the weighted set of objectives to assess a design's quality	$evaluate(d)$	see Equation 1
Modifier	A function that creates a new design by modifying a generated design.	$m(d) = d' \forall m \in \mathbb{M}$	A function that sets the fat used in a recipe to butter
Heuristic Map	A set of weights between modifiers and objectives that express how effective a modifier is expected to be at improving an objective	$\alpha_{m \to o} \in \mathbb{H}$	A designer sets a weight of 2.0 between a low-melting point objective and a butter modifier
Domain Agnostic Metaheuristic Library	Library of domain agnostic components of a metaheuristic optimizer		
Design Population	A data structure that organizes generated designs	$\mathbb{D}$	see subsection 5.1
Design Selector	A function that selects a design from the design population	$select\_design(\mathbb{D}) = d$	A function which returns the highest scoring design that has been generated.
Modifier Selector	A function that selects a modifier to use on a design	$select\_modifier(d, \mathbb{H}, \mathbb{D}) = m$	A function that selects the modifier that is expected to most improve the design
Stopping Criteria	A function that determines if the optimization results should be returned	$stop(\mathbb{D}) = \{True, False\}$	A function that returns True if the design scores more a threshold value.

4.1 Objectives

The optimizer's objective function measures how well designs meet the designer's goals. OPTIMISM supports metaheuristic methods that reduce multi-objective optimization problems into scalarized objective functions weighted by the designer's preferences. Programmers and domain experts collaboratively build a library of objectives. Designers combine these objectives to define their specific design goals. Consider some cookie objectives that can be combined to make crispy cookies. Alton explains that crispiness is increased by lowering the batter's melting point. A high portion of low-melting point fats (e.g., butter) and lower acidity will reduce the melting point [102]. Thus, objectives that measure the portion of butter and acidity can maximizes crispiness.

In OPTIMISM, designers construct scalarized multi-objective functions by defining a weighting a set of objectives based on their importance. Optimizers seek to maximize this scalarized function. Objectives are functions, implemented by the programmer and domain expert, that estimates how well a design performs under some criteria by returning a normalized value between 0 and 1. Given a set of objectives, $o \in \mathrm{O}$, with weights β_o, the customized objective function will be the weighted sum of objectives' score (Equation 1). As we will demonstrate, domain experts can often define simple objectives for a variety of important criteria that designers can combine to define a variety of multi-objective problems.

$$\max_d f(d) = \sum_{o \in \mathbb{O}} \beta_o o(d) \tag{1}$$

4.2 Modifiers

Modifiers express how designs can be improved to maximize the objective function. Modifiers are functions that take in a design and produce a new, slightly different, design. They take small steps through the search space similar to steps taken by a designer when prototyping. Consider four simple cookie modifiers: two change the proportion of butter to shortening and two that change the proportion of baking soda to baking powder. These modifiers will directly influence the melting point of the batter and thus the crispiness of the cookies. Note that the modifiers change proportions of ingredients, not just amounts. This ensures that the amount of ingredients never gets out of proportion (e.g., cookies made only of butter). Ensuring modifiers do not violate design requirements is left up to the development team.

4.3 Heuristic Maps

Heuristics express rules of thumb for improving objective scores. Different modifiers will affect different objectives; heuristics encapsulate this relationship. In OPTIMISM, a modifiers tendency to improve an objective is expressed with an heuristic weight. That is a modifier, m, that should improve an objective, o, has a heuristic weight $\alpha_{m->o}$. We use heuristic weights to measure the expected value of applying a modifier to a design. The expected value is the sum of potential increases in objectives' scores between a design d and the resulting design from applying a modifiers, $m(d) = d'$, multiplied by the heuristic weight (Equation 2).

$$E(d', m) = \sum_{o \in \mathbb{O}} \alpha_{m->o} \beta_o (1 - o(d)) \tag{2}$$

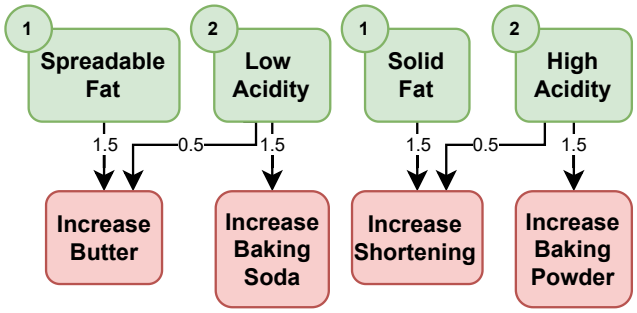


Figure 1: Example Heuristic Map

Input *seeds*: Sample designs provided by domain experts
Input $\mathbb{O}$: Objectives
Input $\mathbb{M}$: Modifiers
Input *Rounds*: the number of iterations to generate samples
Output $\mathbb{H}$: The estimated heuristic map
$current\_round \leftarrow seeds$;
$designs\_by\_modifier \leftarrow \{\}$;
$o\_increases\_by\_m \leftarrow \{\}$;
for $round \in Rounds$ **do**
  $next\_round \leftarrow \{\}$;
  for $d \in current\_round$ **do**
    for $m \in \mathbb{M}$ **do**
      $d' \leftarrow m(d)$;
      add d' to $designs\_by\_modifier[m]$;
      add d' to $next\_round$;
      for $o \in \mathbb{O}$ **do**
        if $o(d) < o(d')$ **then**
          increment $o\_increases\_by\_m[o][m]$;
        end
      end
    end
  end
  $current\_round \leftarrow next\_round$;
end
$\mathbb{H} \leftarrow \{\}$;
for $m \in \mathbb{M}$ **do**
  for $o \in \mathbb{O}$ **do**
    $\alpha_{m->o} = P(o_{\uparrow}|m)$; // See Equation 3
    if $\alpha_{m->o} > 0$ **then**
      $\mathbb{H}[o][m] \leftarrow \alpha_{m->o}$;
    end
  end
end
return $\mathbb{H}$;

Algorithm 2: Sample Data Generation.

$$P(o_{\uparrow}|m) = \frac{|d' \in D_m | o(d) < o(d')|}{|D_m|} \quad (3)$$

We make the relationship between objectives and modifiers explicit in a *heuristic map* (Figure 1) composed of these weighted pairs. Each modifier in a heuristic map can be mapped to multiple objectives and visa-versa. Again, consider the cookie example. Increasing the proportion of butter will likely improve a spreadable fat objective. Similarly, increasing the proportion of baking soda should improve a low-acidity objective. To a lesser extent, because Chef Alton notes that butter is slightly acidic, increasing the butter should also improve the low-acidity objective. Domain experts and designers, without the help of programmers, can either manually construct heuristic maps in the automatically generated GUI or have OPTIMISM generate them from a curated set of seed designs.

Our tuning algorithm estimates heuristic weights given a set of designs, D_m, generated by applying a modifier to a previously generated design. We set the heuristic weight to the proportion of new designs that increased the objective score over all designs generated by the modifier (Equation 3). Domain experts seed the tuning method with curated designs. Over multiple rounds, we apply each modifier to the designs and track increases in each objective's score (Algorithm 2).

The heuristic library is the foundation of domain specific optimizers. The highly flexible structures of composing objective functions from weighted objectives, modifiers as simple strategies for improving designs, and heuristic maps that associate modifiers with the objectives scaffolds heuristic implementation in a way that is accessible to domain experts and programmers.

4.4 Implementing Heuristic Libraries

```
registry = make_registrar()
@registry(Heuristic_Component.Objective,
          "high_solid_fat")
@inverse_objective
@registry(Heuristic_Component.Objective,)
def high_spreadable_fat(cookie: Cookie) -> float:
    return cookie.butter_cup / 1.0
@registry(Heuristic_Component.Modifier)
def add_butter(cookie:Cookie, increment:float=0.25):
    cookie.shortening_cups -= increment
    cookie.butter_cup += increment
@registry(Heuristic_Component.Modifier)
def add_shortening(cookie:Cookie,
                   increment:float=0.25):
    cookie.shortening_cups += increment
    cookie.butter_cup -= increment
```

Figure 2: Example cookie heuristic library code registered to the cookie heuristic `registry` using Python decorators.

To build a domain specific heuristic library, the programmer writes objective and modifier functions with a signature that starts with a parameter for the design (e.g., a cookie recipe) being evaluated or modified and any additional default parameters. Then, the programmer registers these components in the library using a python decorator (e.g., Figure 2.2). The decorator stores a pointer to the function in a domain specific registry (Figure 2.1). When the heuristic map interface is generated, it will reference this registry and make each registered element available with the given

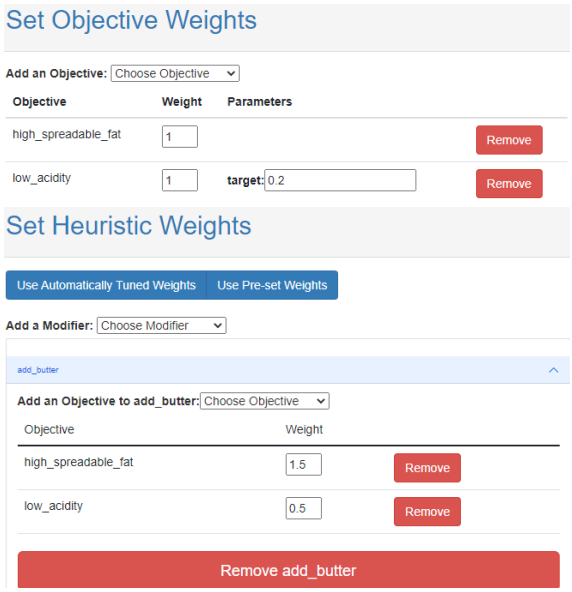


Figure 3: Objective and heuristic weights are set in a GUI.

function-name or an optional identifier provided in the decorator (e.g., "high_solid_fat"). We provide a specialized set of decorators which can modify the output of objectives and modifiers based on common patterns. For instance, the inverse decorator (Figure 2.4) will return the inverse value of an objective (e.g., $inv(o, d) = 1 - o(d)$). A domain specific GUI is generated from the registry (Figure 3) with objectives listed in a drop down menu enabling designers to assign weights to objectives for a specific optimization. Additional parameters can be modified in the Objective Parameters column of the weighting table. Similarly, designers create a heuristic map by associating modifiers with objectives.

5 METAHEURISTIC LIBRARY

Metaheuristic optimization combines domain specific heuristics with stochastic search strategies to form a wide variety of optimization methods. OPTIMISM separates the domain specific and domain agnostic components of these methods to reduce the burden on development teams. Metaheuristic methods are characterized by how designs are chosen and modified in each iteration. Simulated annealing, for example, is defined by how designs are selected in each iteration, but design modification is left up to the programmer. Monte-Carlo Markov chains, on the other hand, are defined by the probabilities of moving from one state to the next (i.e., modifier selection) but cannot jump around the discovered design space. Making these different strategies pluggable enables programmers to mix and match methods until they find one suited to the domain. Programmers may identify established methods that are well suited to their domain or compose new methods.

We present OPTIMISM's domain agnostic library which defines a optimizer's metaheuristic strategy. In OPTIMISM, metaheuristics consist of three components: *stopping criteria*, *design selectors*, and *modifier selectors*. Stopping criteria determine when to stop iterating. A design selector chooses a design to modify in an iteration. A modifier selector determines which modifier to apply to that design. Each of these components make their decisions based on the scalarized objective function and objective scores, the heuristic map, and information collected during optimization in a *design population*. Our domain agnostic library enables programmers to implement a wide variety of optimizers by mixing and matching these elements without writing any new optimization code.

5.1 Design Population

Table 3: The types of information maintained by an optimizer's Design-Population.

Value	Key to Generated Designs
Iterations	Iteration which generated the design.
Scores	Objective function score of design. (i.e., $f(d)$, see Equation 1)
Score Differences	Difference in score from prior design. (i.e., $f(d') - f(d)$).
Objective Scores	Individual Objective scores of design. (i.e., $o(d) \forall o \in \mathbb{O}$).
Objective Score Differences	Differences in objective scores from prior design. (i.e., $o(d') - o(d)$)
Modifier Used	Modifier used to generate design.

OPTIMISM organizes and sorts generated designs in a domain agnostic format called the design population. It collects data about how well a design performed under the objective-function and individual objectives, the change from the prior design, the iteration in which the design was generated, and the modifier that generated it (Table 3). The design population is self-pruning. When it reaches a maximum size it will randomly remove a design to maintain that capacity. Pruning is biased to remove the worst performing designs.

The design population will be returned to the designer after optimization so that they can access and organize designs based on different criteria. Additionally, after optimization, we use Deb's [42] algorithm for finding the non-dominated front of a set of designs to calculate the Pareto set. Designers can view designs on the Pareto set or sorted by the scalarized objective function score.

5.2 Stopping Criteria

The domain agnostic library provides four parameterizable stopping criteria that determine when the optimizer should stop iterating and return the results to the designer (Table 4). We derive three these from common stopping criteria across our literature survey. The fourth method introduces the concept of Pareto dominance: designs that cannot improve individual objectives without trading off other objectives. By stopping when many iterations have not returned a

Table 4: Programmer parameterizable stopping criteria

Stopping Criteria	Programmer's Parameters	Definition
Exhausted Iterations	I: maximum iterations	Returns true when the I is reached.
Matched Threshold Score	N: Design count s: threshold score	Returns true when N designs have been generated with Objective Function scores $\geq s$
Scalarized Objective Function Converged	I: iterations Δ: score changes	Returns true if I iterations have passed with no more than Δ difference in Scalarized Objective Function Scores
Pareto Set Converged	I: iterations Δ: score changes	Returns true if I iterations have passed without discovering a design that dominates the prior design (i.e., $d' \leq d$).

Pareto dominant design, this stopping criteria returns results when a region of the Pareto set has likely been discovered. Each stopping criteria is a function that, given the design population, will return true if the optimizer should stop. It is trivial to construct a variety of more complex stopping criteria by logically combining the results of these categories. For instance, we may want to continue searching for a local-maxima after a threshold score has been met. To do this we can combine the results of a threshold and convergence stopping criteria so it only stops when both criteria are met. Similarly, we can set the optimizer to stop at a maximum number of iterations by returning the value of the combined stopping criteria (i.e., threshold and convergence) or an exhausted-iterations stopping criteria.

5.3 Design and Modifier Selectors

Selecting a design and modifier are critical steps in each iteration of an optimizer and define the metaheuristic method. Particularly in bumpy domains with many local maxima, randomness is introduced into this process. These methods are non-deterministic and this makes it difficult to identify strategies that will efficiently satisfice the designer's needs. By switching out pluggable design and modifier selectors, programmers can test a variety of methods. Additionally, a narrow subset of these selectors can be accessed by curious domain experts and designers through a GUI (Figure 5).

Design and modifier selectors follow the same procedure shown in Algorithm 3. The optimizer inputs a set of values sorted by the programmer's criteria. Sorting functions are specific to either design or modifier selection. Additionally, a selector is defined by a selection probability function, P. In many cases this function returns a static probability threshold. Some metaheuristic methods use more complex probability functions. For example, simulated annealing increases the probability of selecting high-performing designs as the number of iterations increases (i.e., the size of the design population). This *cooling function* will converge on a high scoring region later in the optimization process. OPTIMISM includes a variety of parameterizable cooling schedules based on existing methods [10]. The selector creates a probability threshold by passing the current state of the design population and heuristic map to the probability function. Given this probability threshold, each value is considered in the sorted order. If a random variable is less than or equal to the threshold, the current value is selected. If the threshold is never met the last value in the sorted set is selected.

5.3.1 Design Selectors. Design selection is critical to the optimization process. An optimizer that chooses high scoring designs quickly climbs towards a local maximum, but can miss distant, higher-scoring regions. Alternatively, choosing a design randomly will jump to a new regions but will not climb to a local maximum. Standard optimization algorithms balance between selecting quality designs and randomly searching by considering factors such as how long the optimizer has been running, the quality of previously discovered designs, and how often each design has been visited. OPTIMISM's library of design selectors provides a general, domain agnostic strategy for managing these trade-offs.

The primary way to customize design selectors is by changing the sorting order of the designs. Most often we choose a design with a bias towards those that are performing the best. This is true for a variety of standard metaheuristic methods (e.g., evolutionary

```
1  Input values: The set of designs or modifiers to select from
2  Input P: A probability function to set the selection
     threshold
3  Input sort: A sorting function for the values
4  Output value: A selected design or modifier to use in the
     optimizer iteration threshold ← P(𝔻, ℍ) ;
5  sorted_values ← sort(values, 𝔻, ℍ) ;
6  for v ∈ sorted_values do
7  |   p̂ ← random value between 0 and 1;
8  |   if p̂ ≤ threshold then
9  |   |   return v;
10 |   end
11 end
12 return last value in sorted_values;
```

Algorithm 3: Selector Algorithm

methods [52]). This is done by sorting the designs from highest to lowest objective function scores. Programmers can further refine this by sorting by individual objective scores (e.g., Guided Local Search [137]). Other metaheuristic methods (e.g., Tabu search [55], ant-colony optimization [46]) select designs based on their visitation history. For these cases, the population can be sorted based on how many times the same design was generated. Similarly, we can bias the selector towards recent designs by sorting by the iteration that generated each design (e.g., iterative local search [93]).

5.3.2 Modifier Selectors. While design selectors offer a variety of strategies for jumping around the discovered search space, the optimization process is only as diverse as the designs that are generated by modifiers. Each modifier in the heuristic map will generate a neighboring design to the currently selected design. Modifier selectors decide which neighbor to generate and add to the design population. Choosing a modifier with each iteration is independent from design selection, but equally important. Modifier selectors select (see Algorithm 3) modifiers from the heuristic map. OPTIMISM offers three sorting values ($v(m)$) for modifiers.

Actual Value. A modifier's actual-value measures the difference between the current design's objective function (Equation 4) or individual objective (Equation 4) scores. Since this requires the execution of each modifier the resulting designs are cached and used once a modifier is selected.

$$v(m) = f(m(d)) - f(d) \tag{4a}$$

$$v(m) = o(m(d)) - o(d) \tag{4b}$$

Expected Value. To avoid executing all modifiers we estimate how much a modifier will improve the design using Equation 2 and the heuristic weights. The expected values can serve as the probability of state changes in methods such as Monte-Carlo Markov chains.

Historical Value. The design population tracks both the changes in objective scores and which modifiers are used to create designs. From this information, we can keep track of how often, and by how much, each modifier has improved prior designs. Just as we did

```
optimizer = Optimizer(registry.heuristic_map,
                      biased_towards_best_design,
                      reached_threshold_score,
                      biased_to_expected_best_mod)
design_population = optimizer.optimize([Cookie()])
```

Figure 4: Sample Optimizer Metaheuristic configuration.

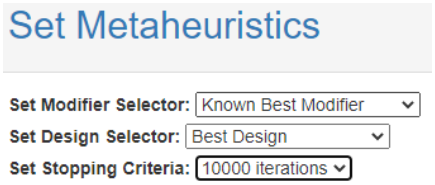


Figure 5: GUI for adjusting metaheuristics.

in the heuristic map tuning algorithm (see Algorithm 2), we can use this information to estimate the probability that a modifier will improve the design (see Equation 3). This sorting criteria is useful for implementing methods like Ant-Colony optimization where we want to bias modifier selection towards those that have a history of improving designs.

6 DEVELOPING WITH OPTIMISM

Once the domain expert and programmer has implemented a design representation and heuristic library, configuring an optimizer only requires the programmer to assign a stopping criteria, design selector, and modifier selector (Figure 4). Programmers choose these components from OPTIMISM's library and can refine their behavior by adjusting their parameters. Notably, building different optimizers only requires a change in these pluggable components and no new code. Figure 5 shows a simple GUI for switching out design selectors, this gives curious domain experts and designers a limited ability to adjust metaheuristics, however they cannot adjust selector parameters without programming. In most cases, we expect programmers will do this work directly in code.

Because it is easy to create a wide variety of optimizers with different combinations of metaheuristics, the new challenge is to find a satisficing optimizer. The programmer must try out different metaheuristics until they find one that converges, at least, faster than a designer could manually produce a design and results in designs that are, at least, as good as what designers produce. To make this process easier, we provide a evaluation tools to experiment with different optimizers. The first tool runs an optimizer many times with random scalarizations of the objectives and collects the resulting scores and convergence times. Using this data, programmers can compare the efficacy and efficiency of many different optimizers. We use this in our demonstrations to compare different optimizers in the domains. The second evaluation tool collects the discovered Pareto set from many runs of an optimizer with random scalarizations, similar to an all-weighted sums method for estimating the Pareto front. This enables programmer to examine the trade offs between different objectives and compare how effectively different optimizers discover the Pareto set. Figure 6 shows a visualization of the Pareto front of chewy vs cakey cookies.

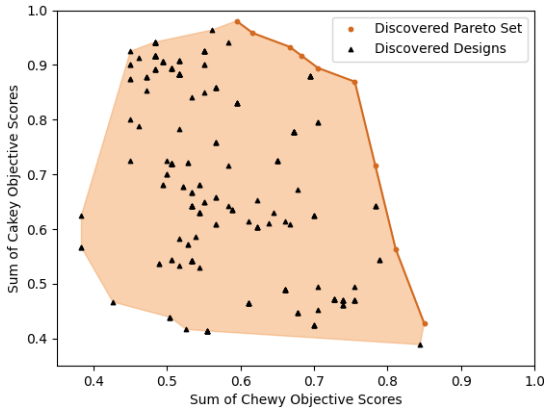


Figure 6: Discovered Pareto front of cookie objectives.

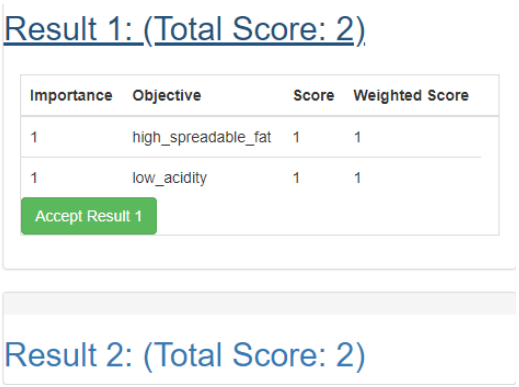


Figure 7: GUI for displaying optimization results and sample cookies generated with different objective functions.

7 DESIGNING WITH OPTIMIZERS

The optimizers produced by teams of domain experts and programmers capture and represent substantial expertise. While designers can use default objective functions and heuristic maps provided by the domain expert, we expect they will often have domain expertise and will use OPTIMISM's GUI to modify the search process. The key difference between domain experts and designers is that designers do not have access to a programmer. They cannot program new objectives or modifiers, nor can they change the design representation structure. Any designer could use Chef Alton's optimizer with minimal expertise, however a curious and clever baker could also recombine objectives and heuristics to generate new cookies. For example, using objectives that measure solidity of fat and require higher acidity to create puffy cakey cookies [25]. Mixing and matching low-level objectives and heuristics gives designers greater flexibility in what the optimizers can produce. Designs generated by an optimizer are displayed in a table sorted by objective function scores (Figure 7). Designers select the results to view them in a domain specific format (e.g., cookie recipes, web pages, 3D models, vector graphics, knitting instructions). More advanced visualizations of the optimization results are left to future work (e.g., visualizing the Pareto set).

Table 5: Summary of final optimizer configurations across all demonstrative tools.

Demonstration	Design-Selector	Modifier-Selector	Stopping-Criteria
Cookies	Best Design	Expected Best Modifier	Pareto set has not changed in 100 iterations
Cataract Lenses	Highest score wtih increasing probability	Best-known modifier with 85% probability	Converge after 10 iterations with <5% change
Splints	Highest score wtih increasing probability	Expected Best Modifier	Reached threshold 95% perfect score
Tactile Graphics	Highest score with 85% probability	Modifier with greatest history of success	1000 iterations
Tile-Decors	Highest score with increasing probability	Best-known modifier for most important objective	Reached threshold 95% perfect score
Knitting	Highest score with 85% probability	Expected best modifier with 85% probability	10,000 iterations

8 DEMONSTRATIONS

We used OPTIMISM to build five domain specific optimizers, two of which have been published in prior work [67, 68]. Each domain is built on the domain expertise of non-technical experts which we source by analysing design patterns from online communities [64], collaborating with researchers and users in the domain [67], and by building on prior community engagements [69]. In our evaluation of these demonstrations we will refer to objective function scores as percentages of the maximum possible score (i.e., score of 1 on all objectives) to normalize the results across many different domains and objective functions. The final configurations of each optimizer are shown in Table 5.

8.1 Cataract Lens Selection

We recruited David, an Ophthalmologist, and Brian[2], a programmer, to build an optimizer that selects prosthetic lenses for cataract surgery. Over two weeks they coordinated by email to build their optimizer. As David explained, choosing a lens depends on a variety of factors such as patient outcomes, available lenses, and error tolerances. Cataract lens are defined by an a-constant dependent on lens models, and an inter-ocular-lens power (IOL) which comes in 0.5 increments. Brian implemented this design representation with two parameters. David explained that he assesses lenses based on a variety of formulas [63, 118] that predict the patient's resulting refraction value (i.e., their glasses prescription). These formulas will predict different values and David weights them depending on information about the patient. Brian encoded these formulas as objectives that compare the predicted refraction to David's target refraction. Brian implemented three modifiers that chose a random a-constant from available models and incremented/decremented the IOL. We provided Brian with example code from other demonstrations and we were available to answer his questions about implementing the library and optimizer over email. After some trial and error, he chose a default design selector that sorts design by objective function scores and selects designs with increasing probability over time. His modifier selector sorts modifiers by their actual-value and selects with a static 85% probability. He uses actual-values instead of expected-values because he felt that this would increase David's confidence in the results and came at little cost to efficiency since his three modifiers are very fast. He used a convergence stopping-criteria that halts when 10 samples have stayed with in a 5% difference of objective function scores.

[2]Participant selected psuedonyms

Brian and David had no prior experience implementing metaheuristic optimizers. Despite their lack of experience, their optimizer selected appropriate lenses. David provided de-identified patient data and lens selections for 10 prior patients. We compared his choices and those made by the optimizer. OPTIMISM selected the same a-constant with 100% accuracy and the correct IOL with 80% accuracy. In the two samples where the IOL differed from David's decision, it was by a single increment. David explained that this was a safe margin of error similar to the different choices two Ophthalmologists would make. Further, in the two cases where the optimizer differed from David, his choice was the second highest scoring option and visible to him in the GUI.

8.2 Occupational Therapy Splints

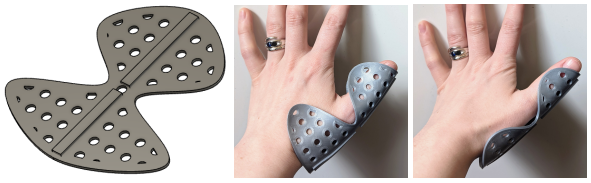


Figure 8: Splints are thermoformed to the patient's hand.

OPTIMISM's modifiable objective functions can enable an Occupational Therapist (OT) to customize splints to a patient's needs. We built a thumb splint optimizer based on field notes from a six month field study in OT clinics [69]. We represent splints with a set of parameters from a standardized splint pattern used by those OTs. We provide modifiers that increment and decrement each parameter by 1mm and derived objectives based on the OT's expertise:

Fit to Patient. The smaller the difference between a splint's parameters and corresponding patient measurements, the better it will fit. Fit is critical for ensuring splints properly restrict movement.

Restriction. Splints restrict movement of specific joints to support healing. OTs estimate the restriction by the width of the *wings* of the splint. Increasing width increases restriction.

Durability. Splints are more likely to break where wider wings and cooling holes introduce material strain. OTs estimate durability with cooling-hole density and inversely to the wing widths.

Comfort. Restriction and durability reduce comfort which leads to abandonment. OTs estimate comfort by the density of cooling holes and reductions of the lower-wing which can irritate the wrist.

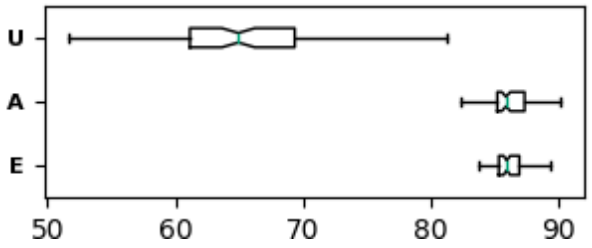


Figure 9: Percentage of perfect score across Uniform (U), Automatic (A), and Expert's (E) weights.

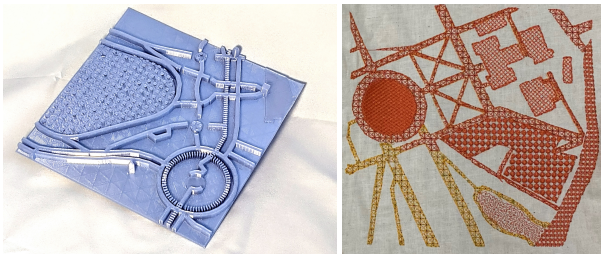


Figure 10: Sample optimized tactile maps optimized for 3D printing and machine embroidery techniques.

By changing the objective weights, we can quickly customize splints. Traditionally, OTs will spend at least one entire patient appointment manually sizing a splint. This optimizer generates the same sizing data in less than a second, allowing the OT to focus on other important aspects of patient care. Figure 8 shows two splints generated using the same patient measurements. The left splint has an increased weight on the restriction objective which resulted in wider wings. The right splint decreased this weight and increased the weight on comfort, causing the lower wing to narrow. Using this optimizer, OTs could easily produce a variety of splints to test out with patients.

Heuristic weights play a significant role in optimizer results because they express the relationship between objectives and modifiers. To demonstrate this, we compared three splint heuristic maps across 100 optimization trials using our evaluation tool. One heuristic map used weights we inferred from our study notes. The second, used weights by seeding our heuristic map tuning method with the sample splints created by the OTs. The third weighted all objectives and modifiers equally. Figure 9 shows how each of these heuristic maps performed. A Welch's Anova tests shows a significant ($F = 439.79, p < 0.01$) effect on the splints' objective scores. A Games Howell Post-hoc analysis shows a significant difference ($p < 0.01$) between the expert and uniform ($T = 29.3$) and tuned and uniform ($T = 29.2$) conditions, but not the expert and tuned conditions ($T = 0.9$). This shows the value of expressing domain expertise directly through heuristic maps or indirectly by seeding our heuristic map tuning algorithm.

The splinting domain demonstrates the value domain experts designing heuristics. While there may be a more efficient or effective optimization methods, we were able to build a splinting tool that produces satisficing results based on a description of the splint design process. By structuring the optimization around heuristics, we can embed the OT's design process into a tool. This reduces manual iterations across many appointments to an optimization that consistently converges in under a second. The resulting design is ready to print and provide to the patient. Further, OPTIMISM's multi-objective structure enables the OTs to quickly explore objective trade offs either by browsing the Pareto set after a single optimization or by adjusting weights and running the optimizer multiple times.

8.3 Tactile Graphic Optimization

Beyond customizing objective functions, modifying heuristic strategies can have significant benefits. In this demonstration we discuss prior work on Maptimizer [67], a tool built with OPTIMISM that generates tactile maps that support blind navigation (Figure 10), and a modification of its underlying optimizer to create a new tool for optimizing machine embroidered tactile graphics [122]. Maptimizer was developed by an interdisciplinary team of programmers and accessibility experts. These tools give blind designers significant control over the optimization of tactile graphics so that they, without the help of a programmer, can express what information is most critical to them (objectives) and how to best represent that information (modifiers). Each designer will have different priorities and preferences that they can express by changing objective and heuristic weights. In this optimizer, objectives measure how much high-value information is included in a graphic, how well specific pieces of information are conveyed, and how cluttered the graphic is. For implementation details refer to [67].

In the prior study of Maptimizer [67], six blind designers provided objective and heuristic weights in a simple web form, a precursor to OPTIMISM's automatically generated GUI. The resulting maps generated with these weights produced maps that were optimized for each individual participant and for four unique locations. In a variety of navigation tasks, the optimized maps out performed both manually customized maps and standardized maps. Without being able to reconfigure objectives and heuristics, the resulting maps could not adapt to each participant's needs.

OPTIMISM's flexibility enables domain experts and programmers to easily expand the scope of existing optimizers. Two undergraduate researchers who were not involved in the creation of Maptimizer (A8, A13), were able to create a new optimizer to generate tactile graphics optimized for machine embroidery using Maptimizer's optimizer code. They did this by programming new objectives that ensured the graphics have continuity of embroidered texture across designer-specified regions and contrast between overlapping and neighboring regions. Additionally, they modified the original 3D printed tactile graphic representation to generate SVG for machine embroidery. These simple additions substantially expanded the scope of the original tool to accommodate a new manufacturing technique. Usually, modifying a generative design tool to accommodate a new manufacturing technique with unique constraints would require substantial effort but by factoring objectives and modifiers specific to tactile graphics and 3D printing into separate sections of the heuristic map, A8 and A13 had little difficulty extending the system.

Table 6: Tile-decor optimizer configurations. A-D vary design selector. D-G vary modifier selector. G was the best optimizer

Optimizer	Design-Selector	Modifier-Selector
A	Select random design	Select modifier with highest expected value
B	Select highest scoring design	Select modifier with highest expected value
C	Select highest scoring design with 0.85 probability	Select modifier with highest expected value
D	Select highest scoring design with probability increasing with iterations	Select modifier with highest expected value
E	Select highest scoring design with probability increasing with iterations	Select best-known modifier for objective function
F	Select highest scoring design with probability increasing with iterations	Select best-known modifier for lowest scoring objective
G	**Select highest scoring design with increasing probability**	**Select best-known modifier for most important objective**

Figure 11: Chen et al's [32] original tile decor (left) and our equivalent tile decor (right)

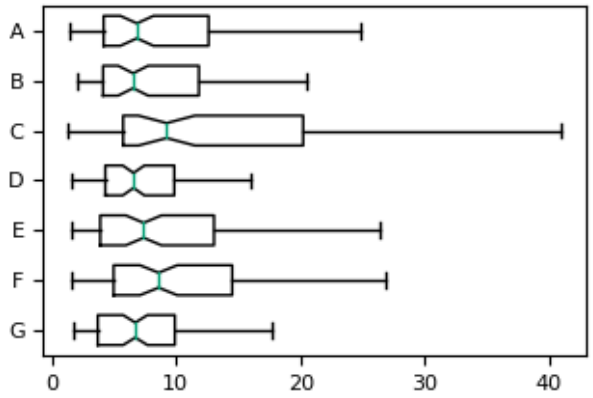
8.4 Replicating Tile-Decors

In this demonstration, we replicate Chen et al's [32] heuristic based optimization of "*objects composed of connected tiles*" and increase its efficiency and efficacy with OPTIMISM. Like OPTIMISM's objective functions, their objective function is a weighted scalarization of four objectives: minimize neighborhood distance, maximize surface approximation, maximize hinge-placement, and minimize repulsion. They apply four modifications: randomly placing tiles, attracting tiles, repulsing tiles, and scaling tiles. We have replicated this method without OPTIMISM and created seven OPTIMISM optimizers that used different metaheuristics.

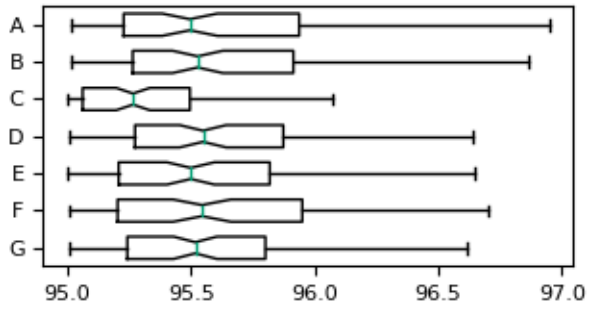
Table 7: Results of Games Howel Post-hoc analysis comparing tile decor experiments. *Indicates significance (p<0.05).

	A	B	C	D	E	F	G
A Score		-0.2	***4.8****	-0.1	0.6	0.2	0.6
A Time		0.6	***-3.02****	0.8	-0.4	-1.4	1.5
B Score	-0.2		***5.3****	0.1	0.8	0.4	0.8
B Time	0.6		***-3.5****	0.2	-1.0	-2.0	1.0
C Score	***4.8****	***5.3****		***-5.6****	***-4.4****	***-4.8****	***-4.9****
C Time	***-3.02****	***-3.5****		***3.6****	2.6	1.7	***4.1****
D Score	-0.1	0.1	***-5.6****		0.8	0.4	0.7
D Time	-0.4	-1.0	2.6		-1.1	-2.1	0.7
E Score	0.6	0.8	***-4.4****	0.8		-0.4	-0.1
E Time	-1.4	-2.0	1.7	-2.1		-0.9	1.8
F Score	0.2	0.4	***-4.8****	0.4	-0.4		0.4
F Time	-1.4	-2.0	1.7	-2.1	-0.9		2.7
G Score	0.6	0.8	***-4.9****	0.7	-0.1	0.4	
G Time	1.5	1.0	***4.1****	0.7	1.8	-1.4	

By plugging in different design and modifier selectors we were able to test metaheuristic strategies without modifying the domain specific code. Using our evaluation tool, we conducted seven experiments (Table 6) where we generated a packed, cylindrical, tiled surface (Figure 11) for 100 optimizations. We kept the stopping condition constant, halting the optimization when a tile-decor was discovered that achieve 95% of a perfect objective score or 10000 iterations were exceeded. The first four experiments vary the modifier selector (Table 6 A-D). A Welch's Anova showed a significant effect of varying the modifier selector ($p < 0.01$) on scores ($F = 17.4$) and optimization time ($F = 4.6$). We then varied design selectors (Table 6 D-G) and a Welch's Anova showed no significant effect on score ($p > 0.1, F = 0.3$) but did find an significant effect on time



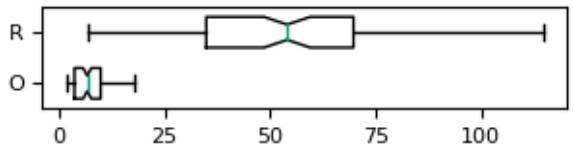
(a) Convergence time in seconds.



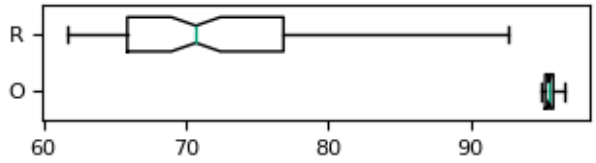
(b) Percentage of perfect scores.

Figure 12: Comparison of different metaheuristic configurations by time and score. Experiment descriptions in Table 6

($p < 0.05, F = 3.1$). A Games Howell post-hoc analysis the shows differences across experiments (Table 7, Figure 12). In comparison to our replication of Chen et al's original algorithm (Figure 13a, 13b), our optimizer converged on significantly ($p < 0.01$) higher scores ($T = 39.8$) and converged significantly faster ($T = -16.12$) based on a Welch's T-test. Further, experimenting with different metaheuristics gave us unique insights into the optimization problem. Unlike, Chen et al's staged optimization method, our optimizer switched between increasing different objectives with each iteration. This revealed a trade-off between attraction and repulsion objectives that is poorly supported by a multi-stage method.



(a) Convergence time in seconds.



(b) Percentage of perfect Score.

Figure 13: Box plots comparing OPTIMISM (O) vs replicated (R) optimizers

8.5 Optimizing Knitted Textures

Our final demonstration examines an optimizer built with an early version of OPTIMISM, KnitGIST [68], which uses the strategies of hand knitters to generate complex, machine-knittable textures. KnitGIST exemplifies how OPTIMISM amplifies the expertise of the domain experts and programmers that build optimizers by making the a highly-re-configurable optimizer accessible to non-programming designers. Since KnitGIST was implemented, OPTIMISM has been expanded to include pluggable and parameterizable metaheuristics (e.g., design selectors, modifier selectors, stopping criteria), a tuning algorithm for defining heuristic maps, and an automatically generated GUI for configuring optimizers. However, KnitGIST still represents OPTIMISM's core optimization structure in a complex domain where designers need re-configurable optimizers to generate functional and attractive designs.

Knitters design these patterns by considering the complex relationship between different stitches and their physical and aesthetic properties (e.g., curl, elasticity, opacity, color). For example, knitters will tediously switch the orientation of stitches to affect elasticity and curl. Similarly, knitters change the color of stitches to create colored images (Figure 14), a new addition to the KnitGIST heuristic library. However, the manual process of adjusting textures to achieve these properties is often tedious and difficult. KnitGIST's heuristic library has the most programmatically complex objectives of our demonstrations (e.g., estimations of physical properties, a decision tree classification of texture aesthetics). However, like other demonstrations, it uses simple modifiers derived from common design practice in the domain. KnitGIST amplifies knitters' domain expertise. Embedding knitters' design strategies into KnitGIST through heuristic maps enables designers to manage the complex stitch structures without manual tweaking or programming.

Figure 14: KnitGIST colorwork samples.

9 DISCUSSION

OPTIMISM structure of metaheuristic optimizers gives domain experts a format to make their expertise explicit. Through the development of our five demonstrations and exemplar cookie optimizer we have developed a better understanding of what domains are best suited to OPTIMISM and the flexibility of this framework.

9.1 Suitable Optimization Domains

Our primary goal is to expand the role and agency of domain experts who help build optimizers. Through our demonstrations we have focused on domain experts in under resourced domains such as healthcare, accessibility, and craft communities because these areas are rarely the focus of optimization research. Even introducing small efficiencies in the design process can have out sized impacts (e.g., increased time with patients). We observe that these designers often rapidly prototype by making small modifications to designs and using rule-of-thumb objectives. Which modification comes with each iteration is guided by intuition. When these strategies are applied manually they are time consuming and designers are limited to the number of iterations they can do in a limited amount of time. Chefs tweak ingredient proportions [102]. Ophthalmologists examine different lens values. OTs trim splints [69]. Knitters switch out stitches [68]. OPTIMISM has limited support for domain experts in the development stage beyond the shared language of design representations, objectives, and modifiers and the GUI for adjusting heuristics. There remains opportunities to define design representations, objectives, and modifiers without programming. While this may not be feasible across all domains, interfaces for visually programming parameterized designs or graph structures could build on OPTIMISM's core optimization library. OPTIMISM

empowers domain experts to collaborate with programmers by providing a shared language and by trading guarantees about optimality or efficiency for greater expressive match to design practice. Heuristics make the relationships between modifiers and objectives explicit while replacing designer's intuition with metaheuristics.

OPTIMISM may be amenable to domains where manual design practices are insufficient (e.g., tile-decors, knitted textures); however, OPTIMISM's utility is more limited. Without clear design strategies, domain experts may need more technical expertise to use OPTIMISM. While many domains may require more complex solutions from programmers, our diverse set of demonstrations show how simple heuristics still produce useful tools. In most cases, objectives only compare or average a subset of design parameters and corresponding modifiers increment and decrement these parameters. These easy to program objectives and modifiers may not always be sufficient and programmers can provide more complex functions. The simplicity of these library elements has secondary advantages. First, they are easier for domain experts and designers to understand and reason about. Second, by using many simpler objectives and modifiers, domain experts and designers can mix an match more elements in the GUI rather than relying on the programmer. Third, these are efficient methods and can be executed in many iterations without slowing the optimizer. OPTIMISM is not appropriate for domains that require slower heuristics (e.g., simulation, user-interaction). In cases where optimality is critical, OPTIMISM will not produce sufficient optimizers because metaheuristic methods cannot provide these guarantees. OPTIMISM is meant to lower the floor and entry barriers to optimization. OPTIMISM is complementary to existing optimization toolkits for convex [27], Bayesian [56], and multi-objective [23] optimization.

9.2 Flexibility of Optimizers

While heuristics have an expressive match to designer's practices, programmers often struggle to find an optimization method or metaheuristic that matches a domain. Given the bumpiness and lack of convexity of many domains, it is unclear which methods will work best. The only option is to test many different methods and evaluate the differing results. Without the modularity of OPTIMISM, building each unique optimizer to test different metaheuristics is a substantial burden for programmers. OPTIMISM supports flexible prototyping of optimizers with re-configurable metaheuristics and heuristics. Using simple evaluation tools that run many iterations of these optimizers with randomized objective weights, the programmer can evaluate the effects of different heuristics and metaheuristics on objective scores and convergence times. We demonstrated this in the construction and evaluation of splint and tile-decor optimizers.

As a programming toolkit, OPTIMISM is a jumping off point to build more advanced optimization methods. By structuring objectives as weighted set, we provide a common structure for multi-objective optimization problems. Recall, that decomposition methods for exploring of the Pareto front (e.g., all-weighted sums method) require optimizers that search for solutions to specific scalarizations of the multi-objective problem. As we showed with our Cookie optimizer, OPTIMISM has all of the components needed to implement these types of methods. The design population at the core of an optimizer and that is returned to the designer collects critical information from disparate design representations into a common format. With the the design population, programmers could build multi-objective optimization methods that make decisions based on objective scores, history of improvement, and Pareto dominance. Programmers could build these more advanced optimizers using the same heuristic structures they designed with domain-experts. The optimizers OPTIMISM currently produces are, alone, useful tools in new domains, however methods that produce many Pareto optimal results rather than results sorted by a weighted objective function will require advances in interactive decision making (e.g., [126]) and visualizations of the objective space (e.g.,[78]).

10 LIMITATIONS

We have evaluated OPTIMISM though independent case studies with varying degrees of involvement from programmers, domain experts, and programmers and at different stages in the implementation of OPTIMISM. The time needed to implement optimizers varied depending on the iteration of the toolkit. Indeed, limitations discovered in prior work [67, 68] directly influenced key features. For example, blind designers who used Maptimizer provided critical feedback that influenced the design of the automatically generated GUIs. Ultimately, the last demonstration to be implemented, cataract lenses, was built in two weeks by a team that had no prior experience with OPTIMISM. The programmer, Brian, benefited substantially from optimizer patterns he observed in existing optimizers and was able to adapt them to a novel domain without our support. Without further, longitudinal, case studies we are limited in our analysis of OPTIMISM's learn-ability by different users or its expressive match to novel domains. Many of our demonstrations are motivated by prior work on clinical CAD tools [69] that are used by clinicians and people with disabilities. In these domains, it is essential that extensive, iterative design practices be reduced to quick (e.g., under a minute) calculations and return only a few effective results. In prior work with people with disabilities and clinicians, its clear that that they would prefer one satisfactory result over the opportunity to—and burden of—evaluating many discovered designs with trade-offs (e.g., the Pareto front) [65, 69]. However, this is not representative of all domains that OPTIMISM could be applied to and future research should expand to different domain experts and programmers with varied expertise in optimization.

11 CONCLUSION

We contribute a toolkit for building generative design tools in unique domains. OPTIMISM is designed to: empower domain experts in the collaborative process of building domain specific optimizers, support flexible rapid prototyping of optimizers, and produce satisfying and sufficient designs. Unlike other optimization toolkits, OPTIMISM separates the roles of programmers who implement optimizers and domain experts who guide optimizers to good results. They do this collaboratively by building a library of heuristics that are applied by flexible metaheuristics that can accommodate a wide variety of domains.

ACKNOWLEDGMENTS

We thank the participants and collaborators who have used OPTIMISM to build the systems we have described. This work was funded by: Google; the Center for Research and Education on Accessible Technology and Experiences (CREATE); a NIDILRR ARRT Training grant 90ARCP0005-01-00; and the National Science Foundation (IIS-1907337, IIS-1718651, 2031801, CHS-1907337, FMitF-1836813).

REFERENCES

[1] 2020. TensorFlow Model Optimization. https://www.tensorflow.org/model_optimization [Online; accessed 1. Apr. 2021].
[2] 2021. *ACM Trans. Graph.* Vol. 40. Association for Computing Machinery, New York, NY, USA.
[3] 2021. *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan). Association for Computing Machinery, New York, NY, USA.
[4] 2021. Maple Global Optimization Toolbox Powered by Optimus - Optimization Software - Maplesoft. https://www.maplesoft.com/products/toolboxes/globaloptimization [Online; accessed 1. Apr. 2021].
[5] 2021. Opt4J. https://sdarg.github.io/opt4j [Online; accessed 1. Apr. 2021].
[6] 2021. Optimization Algorithm Toolkit (OAT). https://www.onworks.net/software/app-optimization-algorithm-toolkit-oat [Online; accessed 1. Apr. 2021].
[7] 2021. Optimization Toolbox. https://www.mathworks.com/products/optimization.html [Online; accessed 1. Apr. 2021].
[8] 2021. *SCF '21: Symposium on Computational Fabrication* (Virtual Event, USA). Association for Computing Machinery, New York, NY, USA.
[9] 2021. *UIST '21: The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA). Association for Computing Machinery, New York, NY, USA.
[10] 2022. A Comparison of Cooling Schedules for Simulated Annealing (Artificial Intelligence). http://what-when-how.com/artificial-intelligence/a-comparison-of-cooling-schedules-for-simulated-annealing-artificial-intelligence [Online; accessed 8. Feb. 2022].
[11] Muhammad Abdullah, Martin Taraz, Yannis Kommana, Shohei Katakura, Robert Kovacs, Jotaro Shigeyama, Thijs Roumen, and Patrick Baudisch. 2021. *FastForce: Real-Time Reinforcement of Laser-Cut Structures.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445466
[12] Muhammad Abdullah, Martin Taraz, Yannis Kommana, Shohei Katakura, Robert Kovacs, Jotaro Shigeyama, Thijs Roumen, and Patrick Baudisch. 2021. FastForce: Real-Time Reinforcement of Laser-Cut Structures. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 673, 12 pages. https://doi.org/10.1145/3411764.3445466
[13] Maneesh Agrawala and Chris Stolte. 2001. Rendering Effective Route Maps: Improving Usability through Generalization. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 241–249. https://doi.org/10.1145/383259.383286
[14] Rahul Arora, Alec Jacobson, Timothy R. Langlois, Yijiang Huang, Caitlin Mueller, Wojciech Matusik, Ariel Shamir, Karan Singh, and David I. W. Levin. 2019. Volumetric Michell Trusses for Parametric Design & Fabrication. In *Proceedings of the ACM Symposium on Computational Fabrication* (Pittsburgh, Pennsylvania) *(SCF '19)*. Association for Computing Machinery, New York, NY, USA, Article 6, 13 pages. https://doi.org/10.1145/3328939.3328999
[15] C. Audet and W. Hare. 2017. *Derivative-Free and Blackbox Optimization.* Springer International Publishing. https://books.google.com/books?id=ejVBDwAAQBAJ
[16] Thomas Auzinger, Wolfgang Heidrich, and Bernd Bickel. 2018. Computational Design of Nanostructural Color for Additive Manufacturing. *ACM Trans. Graph.* 37, 4, Article 159 (jul 2018), 16 pages. https://doi.org/10.1145/3197517.3201376
[17] Vahid Babaei, Kiril Vidimče, Michael Foshey, Alexandre Kaspar, Piotr Didyk, and Wojciech Matusik. 2017. Color Contoning for 3D Printing. *ACM Trans. Graph.* 36, 4, Article 124 (jul 2017), 15 pages. https://doi.org/10.1145/3072959.3073605
[18] Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. 2012. Fabricating Articulated Characters from Skinned Meshes. *ACM Trans. Graph.* 31, 4, Article 47 (July 2012), 9 pages. https://doi.org/10.1145/2185520.2185543
[19] Moritz Bächer, Benjamin Hepp, Fabrizio Pece, Paul G. Kry, Bernd Bickel, Bernhard Thomaszewski, and Otmar Hilliges. 2016. DefSense: Computational Design of Customized Deformable Input Devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3806–3816. https://doi.org/10.1145/2858036.2858354
[20] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-It: Optimizing Moment of Inertia for Spinnable Objects. *ACM Trans. Graph.* 33, 4, Article 96 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601157
[21] Shajay Bhooshan, Tom Van Mele, and Philippe Block. 2020. Morph & Slerp: Shape Description for 3D Printing of Concrete. In *Symposium on Computational Fabrication* (Virtual Event, USA) *(SCF '20)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. https://doi.org/10.1145/3424630.3425413
[22] 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 Trans. Graph.* 29, 4, Article 63 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778800
[23] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567
[24] Christian Blum and Andrea Roli. 2003. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.* 35, 3 (2003), 268–308. https://doi.org/10.1145/937503.937505
[25] Alton Brown. 2020. The Puffy Chocolate Chip Cookie. https://altonbrown.com/recipes/the-puffy [Online; accessed 6. Apr. 2021].
[26] Reva Brown. 2004. Consideration of the origin of Herbert Simon's theory of "satisficing" (1933-1947). *Management Decision* 42, 10 (Dec. 2004), 1240–1256. https://doi.org/10.1108/00251740410568944
[27] Edward Burnell, Nicole B. Damen, and Warren Hoburg. 2020. GPkit: A Human-Centered Approach to Convex Optimization in Engineering Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376412
[28] Massimiliano Caramia and Paolo Dell'Olmo. 2020. Multi-objective Optimization. In *Multi-objective Management in Freight Logistics: Increasing Capacity, Service Level, Sustainability, and Safety with Optimization Algorithms.* Springer, Cham, Switzerland, 21–51. https://doi.org/10.1007/978-3-030-50812-8_2
[29] Ruei-Che Chang, Chih-An Tsao, Fang-Ying Liao, Seraphina Yong, Tom Yeh, and Bing-Yu Chen. 2021. Daedalus in the Dark: Designing for Non-Visual Accessible Construction of Laser-Cut Architecture. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 344–358. https://doi.org/10.1145/3472749.3474754
[30] Ruei-Che Chang, Chih-An Tsao, Fang-Ying Liao, Seraphina Yong, Tom Yeh, and Bing-Yu Chen. 2021. Daedalus in the Dark: Designing for Non-Visual Accessible Construction of Laser-Cut Architecture. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 344–358. https://doi.org/10.1145/3472749.3474754
[31] Desai Chen, David I. W. Levin, Wojciech Matusik, and Danny M. Kaufman. 2017. Dynamics-Aware Numerical Coarsening for Fabrication Design. *ACM Trans. Graph.* 36, 4, Article 84 (jul 2017), 15 pages. https://doi.org/10.1145/3072959.3073669
[32] Weikai Chen, Yuexin Ma, Sylvain Lefebvre, Shiqing Xin, Jonàs Martínez, and wenping wang. 2017. Fabricable Tile Decors. *ACM Trans. Graph.* 36, 6, Article 175 (Nov. 2017), 15 pages. https://doi.org/10.1145/3130800.3130817
[33] Xiang 'Anthony' Chen, Stelian Coros, and Scott E. Hudson. 2018. Medley: A Library of Embeddables to Explore Rich Material Properties for 3D Printed Objects. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173736
[34] Xiang 'Anthony' Chen, Stelian Coros, Jennifer Mankoff, and Scott E. Hudson. 2015. Encore: 3D Printed Augmentation of Everyday Objects with Printed-Over, Affixed and Interlocked Attachments. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 73–82. https://doi.org/10.1145/2807442.2807498
[35] Xiang 'Anthony' Chen, Ye Tao, Guanyun Wang, Runchang Kang, Tovi Grossman, Stelian Coros, and Scott E. Hudson. 2018. Forte: User-Driven Generative Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174070
[36] Subramanian Chidambaram, Yunbo Zhang, Venkatraghavan Sundararajan, Niklas Elmqvist, and Karthik Ramani. 2019. Shape Structuralizer: Design, Fabrication, and User-Driven Iterative Refinement of 3D Mesh Models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300893
[37] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Springer US. https://link.springer.com/book/10.1007/978-0-387-36797-2
[38] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Trans. Graph.* 32, 4, Article 83 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461953

[39] Sebastian Cucerca, Piotr Didyk, Hans-Peter Seidel, and Vahid Babaei. 2020. Computational Image Marking on Metals via Laser Induced Heating. *ACM Trans. Graph.* 39, 4, Article 70 (jul 2020), 12 pages. https://doi.org/10.1145/3386569.3392423

[40] Sebastian Cucerca, Piotr Didyk, Hans-Peter Seidel, and Vahid Babaei. 2020. Computational Image Marking on Metals via Laser Induced Heating. *ACM Trans. Graph.* 39, 4, Article 70 (jul 2020), 12 pages. https://doi.org/10.1145/3386569.3392423

[41] Niraj Ramesh Dayama, Kashyap Todi, Taru Saarelainen, and Antti Oulasvirta. 2020. *GRIDS: Interactive Layout Design with Integer Programming.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376553

[42] Kalyanmoy Deb and Kalyanmoy Deb. 2014. *Multi-objective Optimization.* Springer US, Boston, MA, 403–449. https://doi.org/10.1007/978-1-4614-6940-7_15

[43] Ruta Desai, James McCann, and Stelian Coros. 2018. Assembly-Aware Design of Printable Electromechanical Devices. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18).* Association for Computing Machinery, New York, NY, USA, 457–472. https://doi.org/10.1145/3242587.3242655

[44] Ruta Desai, James McCann, and Stelian Coros. 2018. Assembly-Aware Design of Printable Electromechanical Devices. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18).* Association for Computing Machinery, New York, NY, USA, 457–472. https://doi.org/10.1145/3242587.3242655

[45] Nikan Doosti, Julian Panetta, and Vahid Babaei. 2021. Topology Optimization via Frequency Tuning of Neural Design Representations. In *Symposium on Computational Fabrication* (Virtual Event, USA) *(SCF '21).* Association for Computing Machinery, New York, NY, USA, Article 1, 9 pages. https://doi.org/10.1145/3485114.3485124

[46] M. Dorigo, M. Birattari, and T. Stutzle. 2006. Ant colony optimization. *IEEE Computational Intelligence Magazine* 1, 4 (2006), 28–39. https://doi.org/10.1109/MCI.2006.329691

[47] Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational Multicopter Design. *ACM Trans. Graph.* 35, 6, Article 227 (nov 2016), 10 pages. https://doi.org/10.1145/2980179.2982427

[48] Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational Multicopter Design. *ACM Trans. Graph.* 35, 6, Article 227 (nov 2016), 10 pages. https://doi.org/10.1145/2980179.2982427

[49] Peitong Duan, Casimir Wierzynski, and Lama Nachman. 2020. *Optimizing User Interface Layouts via Gradient Descent.* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376589

[50] John J. Dudley, Jason T. Jacques, and Per Ola Kristensson. 2019. Crowdsourcing Interface Feature Design with Bayesian Optimization. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19).* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300482

[51] Jérémie Dumas, Jean Hergel, and Sylvain Lefebvre. 2014. Bridging the Gap: Automated Steady Scaffoldings for 3D Printing. *ACM Trans. Graph.* 33, 4, Article 98 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601153

[52] Michael T. M. Emmerich and André H. Deutz. 2018. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Nat. Comput.* 17, 3 (Sep 2018), 585–609. https://doi.org/10.1007/s11047-018-9685-y

[53] Paulo Paneque Galuzio, Emerson Hochsteiner de Vasconcelos Segundo, Leandro dos Santos Coelho, and Viviana Cocco Mariani. 2020. MOBOpt — multi-objective Bayesian optimization. *SoftwareX* 12 (2020), 100520. https://doi.org/10.1016/j.softx.2020.100520

[54] Christoph Gebhardt and Otmar Hilliges. 2021. Optimization-Based User Support for Cinematographic Quadrotor Camera Target Framing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21).* Association for Computing Machinery, New York, NY, USA, Article 586, 13 pages. https://doi.org/10.1145/3411764.3445568

[55] Fred Glover and Manuel Laguna. 1998. *Tabu Search.* Springer US, Boston, MA, 2093–2229. https://doi.org/10.1007/978-1-4613-0303-9_33

[56] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. 2017. Google Vizier: A Service for Black-Box Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17).* Association for Computing Machinery, New York, NY, USA, 1487–1495. https://doi.org/10.1145/3097983.3098043

[57] Floraine Grabler, Maneesh Agrawala, Robert W. Sumner, and Mark Pauly. 2008. Automatic Generation of Tourist Maps. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–11. https://doi.org/10.1145/1360612.1360699

[58] Jianzhe Gu, Vidya Narayanan, Guanyun Wang, Danli Luo, Harshika Jain, Kexin Lu, Fang Qin, Sijia Wang, James McCann, and Lining Yao. 2020. Inverse Design Tool for Asymmetrical Self-Rising Surfaces with Color Texture. In *Symposium on Computational Fabrication* (Virtual Event, USA) *(SCF '20).* Association for Computing Machinery, New York, NY, USA, Article 14, 12 pages. https://doi.org/10.1145/3424630.3425420

[59] Yacov Haimes. 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE transactions on systems, man, and cybernetics* 1, 3 (1971), 296–297.

[60] Yue Hao, Yun-hyeong Kim, and Jyh-Ming Lien. 2018. Synthesis of Fast and Collision-Free Folding of Polyhedral Nets. In *Proceedings of the 2nd ACM Symposium on Computational Fabrication* (Cambridge, Massachusetts) *(SCF '18).* Association for Computing Machinery, New York, NY, USA, Article 2, 10 pages. https://doi.org/10.1145/3213512.3213517

[61] Yue Hao and Jyh-Ming Lien. 2019. Computational Laser Forming Origami of Convex Surfaces. In *Proceedings of the ACM Symposium on Computational Fabrication* (Pittsburgh, Pennsylvania) *(SCF '19).* Association for Computing Machinery, New York, NY, USA, Article 9, 11 pages. https://doi.org/10.1145/3328939.3329006

[62] Jean Hergel, Kevin Hinz, Sylvain Lefebvre, and Bernhard Thomaszewski. 2019. Extrusion-Based Ceramics Printing with Strictly-Continuous Deposition. *ACM Trans. Graph.* 38, 6, Article 194 (nov 2019), 11 pages. https://doi.org/10.1145/3355089.3356509

[63] Kenneth J. Hoffer. 1993. The Hoffer Q formula: A comparison of theoretic and regression formulas. *Journal of Cataract & Refractive Surgery* 19, 6 (1993), 700–712. https://doi.org/10.1016/S0886-3350(13)80338-0

[64] Megan Hofmann, Lea Albaugh, Ticha Sethapakadi, Jessica Hodgins, Scott E. Hudson, James McCann, and Jennifer Mankoff. 2019. KnitPicking Textures: Programming and Modifying Complex Knitted Textures for Machine and Hand Knitting. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19).* Association for Computing Machinery, New York, NY, USA, 5–16. https://doi.org/10.1145/3332165.3347886

[65] Megan Hofmann, Jeffrey Harris, Scott E. Hudson, and Jennifer Mankoff. 2016. Helping Hands: Requirements for a Prototyping Methodology for Upper-Limb Prosthetics Users. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16).* Association for Computing Machinery, New York, NY, USA, 1769–1780. https://doi.org/10.1145/2858036.2858340

[66] Megan Hofmann, Udaya Lakshmi, Kelly Mack, Rosa I. Arriaga, Scott E. Hudson, and Jennifer Mankoff. 2022. Making a Medical Maker's Playbook: An Ethnographic Study of Safety-Critical Collective Design by Makers in Response to COVID-19. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW1, Article 101 (apr 2022), 26 pages. https://doi.org/10.1145/3512948

[67] Megan Hofmann, Kelly Mack, Jessica Birchfield, Jerry Cao, Autumn Hughes, Shriya Kurpad, Kathryn J Lum, Emily Warnock, Anat Caspi, Scott E. Hudson, and Jennifer Mankoff. 2022. Maptimizer: Using Optimization to Tailor Tactile Maps to Users Needs. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans LA, USA) *(CHI '22).* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3491102.3517436

[68] Megan Hofmann, Jennifer Mankoff, and Scott E. Hudson. 2020. KnitGIST: A Programming Synthesis Toolkit for Generating Functional Machine-Knitting Textures. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20).* Association for Computing Machinery, New York, NY, USA, 1234–1247. https://doi.org/10.1145/3379337.3415590

[69] Megan Hofmann, Kristin Williams, Toni Kaplan, Stephanie Valencia, Gabriella Hann, Scott E. Hudson, Jennifer Mankoff, and Patrick Carrington. 2019. "Occupational Therapy is Making": Clinical Rapid Prototyping and Digital Fabrication. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19).* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300544

[70] Alexandra Ion, David Lindlbauer, Philipp Herholz, Marc Alexa, and Patrick Baudisch. 2019. *Understanding Metamaterial Mechanisms.* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300877

[71] Alexandra Ion, David Lindlbauer, Philipp Herholz, Marc Alexa, and Patrick Baudisch. 2019. Understanding Metamaterial Mechanisms. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19).* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300877

[72] Caigui Jiang, Chengcheng Tang, Hans-Peter Seidel, and Peter Wonka. 2017. Design and Volume Optimization of Space Structures. *ACM Trans. Graph.* 36, 4, Article 159 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073619

[73] Yue Jiang, Wolfgang Stuerzlinger, Matthias Zwicker, and Christof Lutteroth. 2020. *ORCSolver: An Efficient Solver for Adaptive GUI Layout with OR-Constraints.* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376610

[74] Yuhua Jin, Isabel Qamar, Michael Wessely, and Stefanie Mueller. 2020. PhotoChromeleon: Re-Programmable Multi-Color Textures Using Photochromic Dyes. In *ACM SIGGRAPH 2020 Emerging Technologies* (Virtual Event, USA) *(SIGGRAPH '20).* Association for Computing Machinery, New York, NY, USA, Article 7, 2 pages. https://doi.org/10.1145/3388534.3407296

[75] Florian Kadner, Yannik Keller, and Constantin Rothkopf. 2021. AdaptiFont: Increasing Individuals' Reading Speed with a Generative Font Model and Bayesian Optimization. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 585, 11 pages. https://doi.org/10.1145/3411764.3445140

[76] Martin Kilian, Aron Monszpart, and Niloy J. Mitra. 2017. String Actuated Curved Folded Surfaces. *ACM Trans. Graph.* 36, 4, Article 64a (may 2017), 13 pages. https://doi.org/10.1145/3072959.3015460

[77] Mina Konaković, Keenan Crane, Bailin Deng, Sofien Bouaziz, Daniel Piker, and Mark Pauly. 2016. Beyond Developable: Computational Design and Fabrication with Auxetic Materials. *ACM Trans. Graph.* 35, 4, Article 89 (jul 2016), 11 pages. https://doi.org/10.1145/2897824.2925944

[78] Roozbeh Haghnazar Koochaksaraei, Ivan Reinaldo Meneghini, Vitor Nazário Coelho, and Frederico Gadelha Guimarães. 2017. A new visualization method in many-objective optimization with chord diagram and angular mapping. *Knowledge-Based Systems* 138 (2017), 134–154. https://doi.org/10.1016/j.knosys.2017.09.035

[79] Yuki Koyama and Masataka Goto. 2018. OptiMo: Optimization-Guided Motion Editing for Keyframe Character Animation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173735

[80] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential Line Search for Efficient Visual Design Optimization by Crowds. *ACM Trans. Graph.* 36, 4, Article 48 (jul 2017), 11 pages. https://doi.org/10.1145/3072959.3073598

[81] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential Line Search for Efficient Visual Design Optimization by Crowds. *ACM Trans. Graph.* 36, 4, Article 48 (jul 2017), 11 pages. https://doi.org/10.1145/3072959.3073598

[82] Sivam Krish. 2011. A practical generative design method. *Computer-Aided Design* 43, 1 (2011), 88–100. https://doi.org/10.1016/j.cad.2010.09.009

[83] Udaya Lakshmi, Megan Hofmann, Stephanie Valencia, Lauren Wilcox, Jennifer Mankoff, and Rosa I. Arriaga. 2019. "Point-of-Care Manufacturing": Maker Perspectives on Digital Fabrication in Medical Practice. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 91 (nov 2019), 23 pages. https://doi.org/10.1145/3359193

[84] Thomas Langerak, Juan José Zárate, Velko Vechev, David Lindlbauer, Daniele Panozzo, and Otmar Hilliges. 2020. *Optimal Control for Electromagnetic Haptic Guidance Systems.* Association for Computing Machinery, New York, NY, USA, 951–965. https://doi.org/10.1145/3379337.3415593

[85] DoYoung Lee, Jiwan Kim, and Ian Oakley. 2021. FingerText: Exploring and Optimizing Performance for Wearable, Mobile and One-Handed Typing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 283, 15 pages. https://doi.org/10.1145/3411764.3445106

[86] Danny Leen, Tom Veuskens, Kris Luyten, and Raf Ramakers. 2019. JigFab: Computational Fabrication of Constraints to Facilitate Woodworking with Power Tools. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300386

[87] Dingzeyu Li, David I. W. Levin, Wojciech Matusik, and Changxi Zheng. 2016. Acoustic Voxels: Computational Optimization of Modular Acoustic Filters. *ACM Trans. Graph.* 35, 4, Article 88 (jul 2016), 12 pages. https://doi.org/10.1145/2897824.2925960

[88] Dingzeyu Li, David I. W. Levin, Wojciech Matusik, and Changxi Zheng. 2016. Acoustic Voxels: Computational Optimization of Modular Acoustic Filters. *ACM Trans. Graph.* 35, 4, Article 88 (jul 2016), 12 pages. https://doi.org/10.1145/2897824.2925960

[89] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. 2019. Context-Aware Online Adaptation of Mixed Reality Interfaces. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 147–160. https://doi.org/10.1145/3332165.3347945

[90] Qi Liu, Xiaofeng Li, Haitao Liu, and Zhaoxia Guo. 2020. Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art. *Applied Soft Computing* 93 (2020), 106382. https://doi.org/10.1016/j.asoc.2020.106382

[91] J. Derek Lomas, Jodi Forlizzi, Nikhil Poonwala, Nirmal Patel, Sharan Shodhan, Kishan Patel, Ken Koedinger, and Emma Brunskill. 2016. Interface Design Optimization as a Multi-Armed Bandit Problem. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4142–4153. https://doi.org/10.1145/2858036.2858425

[92] Alexander V. Lotov and Kaisa Miettinen. 2008. Visualizing the Pareto Frontier. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer, Berlin, Germany, 213–243. https://doi.org/10.1007/978-3-540-88908-3_9

[93] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. 2003. Iterated Local Search. In *Handbook of Metaheuristics*. Springer, Boston, MA, Boston, MA, USA, 320–353. https://doi.org/10.1007/0-306-48056-5_11

[94] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-Last: Strength to Weight 3D Printed Objects. *ACM Trans. Graph.* 33, 4, Article 97 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601168

[95] Granit Luzhnica and Eduardo Veas. 2019. Optimising Encoding for Vibrotactile Skin Reading. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300465

[96] Jennifer Mankoff, Megan Hofmann, Xiang 'Anthony' Chen, Scott E. Hudson, Amy Hurst, and Jeeeun Kim. 2019. Consumer-Grade Fabrication and Its Potential to Revolutionize Accessibility. *Commun. ACM* 62, 10 (sep 2019), 64–75. https://doi.org/10.1145/3339824

[97] R. Timothy Marler and Jasbir S. Arora. 2010. The weighted sum method for multi-objective optimization: new insights. *Struct. Multidiscip. Optim.* 41, 6 (June 2010), 853–862. https://doi.org/10.1007/s00158-009-0460-7

[98] Jess McIntosh, Hubert Dariusz Zajac, Andreea Nicoleta Stefan, Joanna Bergström, and Kasper Hornbæk. 2020. *Iteratively Adapting Avatars Using Task-Integrated Optimisation.* Association for Computing Machinery, New York, NY, USA, 709–721. https://doi.org/10.1145/3379337.3415832

[99] Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-Printable Robotic Creatures. *ACM Trans. Graph.* 34, 6, Article 216 (Oct. 2015), 9 pages. https://doi.org/10.1145/2816795.2818137

[100] Vittorio Megaro, Jonas Zehnder, Moritz Bächer, Stelian Coros, Markus Gross, and Bernhard Thomaszewski. 2017. A Computational Design Tool for Compliant Mechanisms. *ACM Trans. Graph.* 36, 4, Article 82 (jul 2017), 12 pages. https://doi.org/10.1145/3072959.3073636

[101] Vittorio Megaro, Jonas Zehnder, Moritz Bächer, Stelian Coros, Markus Gross, and Bernhard Thomaszewski. 2017. A Computational Design Tool for Compliant Mechanisms. *ACM Trans. Graph.* 36, 4, Article 82 (jul 2017), 12 pages. https://doi.org/10.1145/3072959.3073636

[102] Michael Menninger. 2014. Three Chips for Sister marsha Transcript. http://www.goodeatsfanpage.com/season3/cookie/cookietranscript.htm

[103] K. Miettinen. 2012. *Nonlinear Multiobjective Optimization*. Springer US. https://books.google.com/books?id=bnzjBwAAQBAJ

[104] Eder Miguel, Mathias Lepoutre, and Bernd Bickel. 2016. Computational Design of Stable Planar-Rod Structures. *ACM Trans. Graph.* 35, 4, Article 86 (jul 2016), 11 pages. https://doi.org/10.1145/2897824.2925978

[105] Roberto A. Montano Murillo, Sriram Subramanian, and Diego Martinez Plasencia. 2017. Erg-O: Ergonomic Optimization of Immersive Virtual Environments. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 759–771. https://doi.org/10.1145/3126594.3126605

[106] Rafael Morales, Asier Marzo, Sriram Subramanian, and Diego Martínez. 2019. LeviProps: Animating Levitated Optimized Fabric Structures Using Holographic Acoustic Tweezers. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 651–661. https://doi.org/10.1145/3332165.3347882

[107] Przemyslaw Musialski, Christian Hafner, Florian Rist, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2016. Non-Linear Shape Optimization Using Local Subspace Projections. *ACM Trans. Graph.* 35, 4, Article 87 (jul 2016), 13 pages. https://doi.org/10.1145/2897824.2925886

[108] Thomas Klaus Nindel, Tomáš Iser, Tobias Rittig, Alexander Wilkie, and Jaroslav Křivánek. 2021. A Gradient-Based Framework for 3D Print Appearance Optimization. *ACM Trans. Graph.* 40, 4, Article 178 (jul 2021), 15 pages. https://doi.org/10.1145/3450626.3459844

[109] Thomas Klaus Nindel, Tomáš Iser, Tobias Rittig, Alexander Wilkie, and Jaroslav Křivánek. 2021. A Gradient-Based Framework for 3D Print Appearance Optimization. *ACM Trans. Graph.* 40, 4, Article 178 (jul 2021), 15 pages. https://doi.org/10.1145/3450626.3459844

[110] Masa Ogata and Yuki Koyama. 2021. A Computational Approach to Magnetic Force Feedback Design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 284, 12 pages. https://doi.org/10.1145/3411764.3445631

[111] Dan R. Olsen. 2007. Evaluating User Interface Systems Research. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA) *(UIST '07)*. Association for Computing Machinery, New York, NY, USA, 251–258. https://doi.org/10.1145/1294211.1294256

[112] Julian Panetta, Florin Isvoranu, Tian Chen, Emmanuel Siéfert, Benoît Roman, and Mark Pauly. 2021. Computational Inverse Design of Surface-Based Inflatables. *ACM Trans. Graph.* 40, 4, Article 40 (jul 2021), 14 pages. https://doi.org/10.1145/3450626.3459789

[113] Seonwook Park, Christoph Gebhardt, Roman Rädle, Anna Maria Feit, Hana Vrzakova, Niraj Ramesh Dayama, Hui-Shyong Yeo, Clemens N. Klokmose, Aaron Quigley, Antti Oulasvirta, and Otmar Hilliges. 2018. AdaM: Adapting Multi-User Interfaces for Collaborative Environments in Real-Time. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3173758

[114] Davide Pellis, Martin Kilian, Helmut Pottmann, and Mark Pauly. 2021. Computational Design of Weingarten Surfaces. *ACM Trans. Graph.* 40, 4, Article 114 (jul 2021), 11 pages. https://doi.org/10.1145/3450626.3459939

[115] Michal Piovarči, David I. W. Levin, Jason Rebello, Desai Chen, Roman Ďurikovič, Hanspeter Pfister, Wojciech Matusik, and Piotr Didyk. 2016. An Interaction-Aware, Perceptual Model for Non-Linear Elastic Objects. *ACM Trans. Graph.* 35, 4, Article 55 (July 2016), 13 pages. https://doi.org/10.1145/2897824.2925885

[116] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Trans. Graph.* 32, 4, Article 81 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2461957

[117] Yingying Ren, Julian Panetta, Tian Chen, Florin Isvoranu, Samuel Poincloux, Christopher Brandt, Alison Martin, and Mark Pauly. 2021. 3D Weaving with Curved Ribbons. *ACM Trans. Graph.* 40, 4, Article 127 (jul 2021), 15 pages. https://doi.org/10.1145/3450626.3459788

[118] Timothy V Roberts, Chris Hodge, Gerard Sutton, Michael Lawless, and contributors to the Vision Eye Institute IOL outcomes registry. 2018. Comparison of Hill-radial basis function, Barrett Universal and current third generation formulas for the calculation of intraocular lens power during cataract surgery. *Clinical & Experimental Ophthalmology* 46, 3 (2018), 240–246. https://doi.org/10.1111/ceo.13034 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/ceo.13034

[119] Thijs Roumen, Yannis Kommana, Ingo Apel, Conrad Lempert, Markus Brand, Erik Brendel, Laurenz Seidel, Lukas Rambold, Carl Goedecken, Pascal Crenzin, Ben Hurdelhey, Muhammad Abdullah, and Patrick Baudisch. 2021. *Assembler3: 3D Reconstruction of Laser-Cut Models.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445453

[120] Thijs Roumen, Yannis Kommana, Ingo Apel, Conrad Lempert, Markus Brand, Erik Brendel, Laurenz Seidel, Lukas Rambold, Carl Goedecken, Pascal Crenzin, Ben Hurdelhey, Muhammad Abdullah, and Patrick Baudisch. 2021. Assembler3: 3D Reconstruction of Laser-Cut Models. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 672, 11 pages. https://doi.org/10.1145/3411764.3445453

[121] Oliver Schütze, Oliver Cuate, Adanay Martín, Sebastian Peitz, and Michael Dellnitz. 2020. Pareto Explorer: a global/local exploration tool for many-objective optimization problems. *Engineering Optimization* 52, 5 (2020), 832–855. https://doi.org/10.1080/0305215X.2019.1617286 arXiv:https://doi.org/10.1080/0305215X.2019.1617286

[122] Margaret Ellen Seehorn, Gene S-H Kim, Aashaka Desai, Megan Hofmann, and Jennifer Mankoff. 2022. Enhancing Access to High Quality Tangible Information through Machine Embroidered Tactile Graphics. In *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication* (Seattle, WA, USA) *(SCF '22)*. Association for Computing Machinery, New York, NY, USA, Article 23, 3 pages. https://doi.org/10.1145/3559400.3565586

[123] Ticha Sethapakdi, Daniel Anderson, Adrian Reginald Chua Sy, and Stefanie Mueller. 2021. Fabricaide: Fabrication-Aware Design for 2D Cutting Machines. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 664, 12 pages. https://doi.org/10.1145/3411764.3445345

[124] Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Trans. Graph.* 36, 4, Article 38 (jul 2017), 11 pages. https://doi.org/10.1145/3072959.3073618

[125] Madlaina Signer, Alexandra Ion, and Olga Sorkine-Hornung. 2021. Developable Metamaterials: Mass-Fabricable Metamaterials by Laser-Cutting Elastic Structures. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 674, 13 pages. https://doi.org/10.1145/3411764.3445666

[126] Henrik Smedberg and Sunith Bandaru. 2022. Interactive knowledge discovery and knowledge visualization for decision support in multi-objective optimization. *European Journal of Operational Research* (2022). https://doi.org/10.1016/j.ejor.2022.09.008

[127] Haichuan Song, Jonàs Martínez, Pierre Bedell, Noémie Vennin, and Sylvain Lefebvre. 2019. Colored Fused Filament Fabrication. *ACM Trans. Graph.* 38, 5, Article 141 (jun 2019), 11 pages. https://doi.org/10.1145/3183793

[128] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. 2012. Stress Relief: Improving Structural Strength of 3D Printable Objects. *ACM Trans. Graph.* 31, 4, Article 48 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185544

[129] T. Stützle. 1999. Local search algorithms for combinatorial problems - analysis, improvements, and new applications. In *DISKI*.

[130] Denis Sumin, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, Jaroslav Křivánek, Karol Myszkowski, and Tim Weyrich. 2019. Geometry-Aware Scattering Compensation for 3D Printing. *ACM Trans. Graph.* 38, 4, Article 111 (jul 2019), 14 pages. https://doi.org/10.1145/3306346.3322992

[131] AKM Khaled Ahsan Talukder and Kalyanmoy Deb. 2020. PaletteViz: A Visualization Method for Functional Understanding of High-Dimensional Pareto-Optimal Data-Sets to Aid Multi-Criteria Decision Making. *IEEE Computational Intelligence Magazine* 15, 2 (2020), 36–48. https://doi.org/10.1109/MCI.2020.2976184

[132] Davi Colli Tozoni, Jérémie Dumas, Zhongshi Jiang, Julian Panetta, Daniele Panozzo, and Denis Zorin. 2020. A Low-Parametric Rhombic Microstructure Family for Irregular Lattices. *ACM Trans. Graph.* 39, 4, Article 101 (jul 2020), 20 pages. https://doi.org/10.1145/3386569.3392451

[133] Davi Colli Tozoni, Yunfan Zhou, and Denis Zorin. 2021. Optimizing Contact-Based Assemblies. *ACM Trans. Graph.* 40, 6, Article 269 (dec 2021), 19 pages. https://doi.org/10.1145/3478513.3480552

[134] Nobuyuki Umentani, Takeo Igarashi, and Niloy J. Mitra. 2015. Guided Exploration of Physically Valid Shapes for Furniture Design. *Commun. ACM* 58, 9 (Aug. 2015), 116–124. https://doi.org/10.1145/2801945

[135] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: Interactive Design and Optimization of Free-Formed Free-Flight Model Airplanes. *ACM Trans. Graph.* 33, 4, Article 65 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601129

[136] Tom Valkeneers, Danny Leen, Daniel Ashbrook, and Raf Ramakers. 2019. StackMold: Rapid Prototyping of Functional Multi-Material Objects with Selective Levels of Surface Details. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 687–699. https://doi.org/10.1145/3332165.3347915

[137] Christos Voudouris and Edward P. K. Tsang. 2003. Guided Local Search. In *Handbook of Metaheuristics*. Springer, Boston, MA, Boston, MA, USA, 185–218. https://doi.org/10.1007/0-306-48056-5_7

[138] David J. Walker. 2018. Visualisation with treemaps and sunbursts in many-objective optimisation. *Genet. Program. Evolvable Mach.* 19, 3 (Sept. 2018), 421–452. https://doi.org/10.1007/s10710-018-9329-0

[139] Rui Wang, Qingfu Zhang, and Tao Zhang. 2016. Decomposition-Based Algorithms Using Pareto Adaptive Scalarizing Methods. *IEEE Transactions on Evolutionary Computation* 20, 6 (2016), 821–837. https://doi.org/10.1109/TEVC.2016.2521175

[140] Michael Wessely, Yuhua Jin, Cattalyya Nuengsigkapian, Aleksei Kashapov, Isabel P. S. Qamar, Dzmitry Tsetserukou, and Stefanie Mueller. 2021. ChromoUpdate: Fast Design Iteration of Photochromic Color Textures Using Grayscale Previews and Local Color Updates. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 666, 13 pages. https://doi.org/10.1145/3411764.3445391

[141] Emily Whiting, Nada Ouf, Liane Makatura, Christos Mousas, Zhenyu Shu, and Ladislav Kavan. 2017. Environment-Scale Fabrication: Replicating Outdoor Climbing Experiences. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1794–1804. https://doi.org/10.1145/3025453.3025465

[142] Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. 2012. Structural Optimization of 3D Masonry Buildings. *ACM Trans. Graph.* 31, 6, Article 159 (Nov. 2012), 11 pages. https://doi.org/10.1145/2366145.2366178

[143] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-Based Design: Concept, Principles and Examples. *ACM Trans. Access. Comput.* 3, 3, Article 9 (apr 2011), 27 pages. https://doi.org/10.1145/1952383.1952384

[144] Katja Wolff and Olga Sorkine-Hornung. 2019. Wallpaper Pattern Alignment along Garment Seams. *ACM Trans. Graph.* 38, 4, Article 62 (jul 2019), 12 pages. https://doi.org/10.1145/3306346.3322991

[145] Jiaxian Yao, Danny M. Kaufman, Yotam Gingold, and Maneesh Agrawala. 2017. Interactive Design and Stability Analysis of Decorative Joinery for Furniture. *ACM Trans. Graph.* 36, 4, Article 157a (mar 2017), 16 pages. https://doi.org/10.1145/3072959.3054740

[146] Christopher Yu, Keenan Crane, and Stelian Coros. 2017. Computational Design of Telescoping Structures. *ACM Trans. Graph.* 36, 4, Article 83 (jul 2017), 9 pages. https://doi.org/10.1145/3072959.3073673

[147] Ya-Ting Yue, Xiaolong Zhang, Yongliang Yang, Gang Ren, Yi-King Choi, and Wenping Wang. 2017. WireDraw: 3D Wire Sculpturing Guided with Mixed Reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3693–3704. https://doi.org/10.1145/3025453.3025792

[148] Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-Aware Retargeting of Mechanisms to 3D Shapes. *ACM*

Trans. Graph. 36, 4, Article 81 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073710

[149] Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-Aware Retargeting of Mechanisms to 3D Shapes. *ACM Trans. Graph.* 36, 4, Article 81 (jul 2017), 13 pages. https://doi.org/10.1145/3072959.3073710

[150] Xiaoting Zhang, Guoxin Fang, Chengkai Dai, Jouke Verlinden, Jun Wu, Emily Whiting, and Charlie C.L. Wang. 2017. Thermal-Comfort Design of Personalized Casts. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 243–254. https://doi.org/10.1145/3126594.3126600

[151] Xiaoting Zhang, Guoxin Fang, Melina Skouras, Gwenda Gieseler, Charlie C. L. Wang, and Emily Whiting. 2019. Computational Design of Fabric Formwork. *ACM Trans. Graph.* 38, 4, Article 109 (jul 2019), 13 pages. https://doi.org/10.1145/3306346.3322988

[152] Yongqi Zhang, Biao Xie, Haikun Huang, Elisa Ogawa, Tongjian You, and Lap-Fai Yu. 2019. Pose-Guided Level Design. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300784

[153] Bo Zhu, Mélina Skouras, Desai Chen, and Wojciech Matusik. 2017. Two-Scale Topology Optimization with Microstructures. *ACM Trans. Graph.* 36, 4, Article 120b (jul 2017), 16 pages. https://doi.org/10.1145/3072959.3095815

[154] Bo Zhu, Mélina Skouras, Desai Chen, and Wojciech Matusik. 2017. Two-Scale Topology Optimization with Microstructures. *ACM Trans. Graph.* 36, 4, Article 120b (jul 2017), 16 pages. https://doi.org/10.1145/3072959.3095815

[155] Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-Guided Mechanical Toy Modeling. *ACM Trans. Graph.* 31, 6, Article 127 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366146

[156] Amit Zoran and Dror Cohen. 2018. Digital Konditorei: Programmable Taste Structures Using a Modular Mold. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3173574.3173974

A DIGITAL FABRICATION SURVEY

(1) Abdullah, Taraz, Kommana, Katakura, Kovacs, Shigeyama, Roumen, and Baudisch [12]
(2) Arora, Jacobson, Langlois, Huang, Mueller, Matusik, Shamir, Singh, and Levin [14]
(3) Auzinger, Heidrich, and Bickel [16]
(4) Babaei, Vidimče, Foshey, Kaspar, Didyk, and Matusik [17]
(5) Bächer, Hepp, Pece, Kry, Bickel, Thomaszewski, and Hilliges [19]
(6) Bhooshan, Van Mele, and Block [21]
(7) Chang, Tsao, Liao, Yong, Yeh, and Chen [30]
(8) Chen, Coros, and Hudson [33]
(9) Chen, Coros, Mankoff, and Hudson [34]
(10) Chen, Levin, Matusik, and Kaufman [31]
(11) Chen, Tao, Wang, Kang, Grossman, Coros, and Hudson [35]
(12) Chidambaram, Zhang, Sundararajan, Elmqvist, and Ramani [36]
(13) Cucerca, Didyk, Seidel, and Babaei [40]
(14) Dayama, Todi, Saarelainen, and Oulasvirta [41]
(15) Desai, McCann, and Coros [44]
(16) Doosti, Panetta, and Babaei [45]
(17) Duan, Wierzynski, and Nachman [49]
(18) Dudley, Jacques, and Kristensson [50]
(19) Du, Schulz, Zhu, Bickel, and Matusik [48]
(20) Gebhardt and Hilliges [54]
(21) Gu, Narayanan, Wang, Luo, Jain, Lu, Qin, Wang, McCann, and Yao [58]
(22) Hao, Kim, and Lien [60]
(23) Hao and Lien [61]
(24) Hergel, Hinz, Lefebvre, and Thomaszewski [62]
(25) Ion, Lindlbauer, Herholz, Alexa, and Baudisch [71]
(26) Jiang, Stuerzlinger, Zwicker, and Lutteroth [73]
(27) Jin, Qamar, Wessely, and Mueller [74]
(28) Kadner, Keller, and Rothkopf [75]
(29) Kilian, Monszpart, and Mitra [76]
(30) Konaković, Crane, Deng, Bouaziz, Piker, and Pauly [77]
(31) Koyama and Goto [79]
(32) Koyama, Sato, Sakamoto, and Igarashi [81]
(33) Langerak et al. [84]
(34) Lee, Kim, and Oakley [85]
(35) Leen, Veuskens, Luyten, and Ramakers [86]
(36) Li, Levin, Matusik, and Zheng [88]
(37) Lindlbauer, Feit, and Hilliges [89]
(38) Lomas, Forlizzi, Poonwala, Patel, Shodhan, Patel, Koedinger, and Brunskill [91]
(39) Luzhnica and Veas [95]
(40) McIntosh, Zajac, Stefan, Bergström, and Hornbæk [98]
(41) Megaro, Zehnder, Bächer, Coros, Gross, and Thomaszewski [101]
(42) Miguel, Lepoutre, and Bickel [104]
(43) Montano Murillo et al. [105]
(44) Morales, Marzo, Subramanian, and Martínez [106]
(45) Musialski, Hafner, Rist, Birsak, Wimmer, and Kobbelt [107]
(46) Nindel, Iser, Rittig, Wilkie, and Křivánek [109]
(47) Ogata and Koyama [110]
(48) Panetta, Isvoranu, Chen, Siéfert, Roman, and Pauly [112]
(49) Park, Gebhardt, Rädle, Feit, Vrzakova, Dayama, Yeo, Klokmose, Quigley, Oulasvirta, and Hilliges [113]
(50) Pellis, Kilian, Pottmann, and Pauly [114]
(51) Ren, Panetta, Chen, Isvoranu, Poincloux, Brandt, Martin, and Pauly [117]
(52) Roumen, Kommana, Apel, Lempert, Brand, Brendel, Seidel, Rambold, Goedecken, Crenzin, Hurdelhey, Abdullah, and Baudisch [120]
(53) Sethapakdi, Anderson, Sy, and Mueller [123]
(54) Shtengel, Poranne, Sorkine-Hornung, Kovalsky, and Lipman [124]
(55) Signer, Ion, and Sorkine-Hornung [125]
(56) Song, Martínez, Bedell, Vennin, and Lefebvre [127]
(57) Sumin, Rittig, Babaei, Nindel, Wilkie, Didyk, Bickel, Křivánek, Myszkowski, and Weyrich [130]
(58) Tozoni, Dumas, Jiang, Panetta, Panozzo, and Zorin [132]
(59) Tozoni, Zhou, and Zorin [133]
(60) Valkeneers, Leen, Ashbrook, and Ramakers [136]
(61) Wessely et al. [140]
(62) Whiting, Ouf, Makatura, Mousas, Shu, and Kavan [141]
(63) Wolff and Sorkine-Hornung [144]
(64) Yao, Kaufman, Gingold, and Agrawala [145]
(65) Yu, Crane, and Coros [146]
(66) Yue, Zhang, Yang, Ren, Choi, and Wang [147]
(67) Zhang, Auzinger, Ceylan, Li, and Bickel [149]
(68) Zhang, Fang, Dai, Verlinden, Wu, Whiting, and Wang [150]
(69) Zhang, Fang, Skouras, Gieseler, Wang, and Whiting [151]
(70) Zhang, Xie, Huang, Ogawa, You, and Yu [152]
(71) Zhu, Skouras, Chen, and Matusik [154]
(72) Zoran and Cohen [156]