Generative Design of Sheet Metal Structures

AMIR BARDA, Tel Aviv University, Israel
GUY TEVET, Tel Aviv University, Israel
ADRIANA SCHULZ, University of Washington, USA
AMIT H. BERMANO, Tel Aviv University, Israel

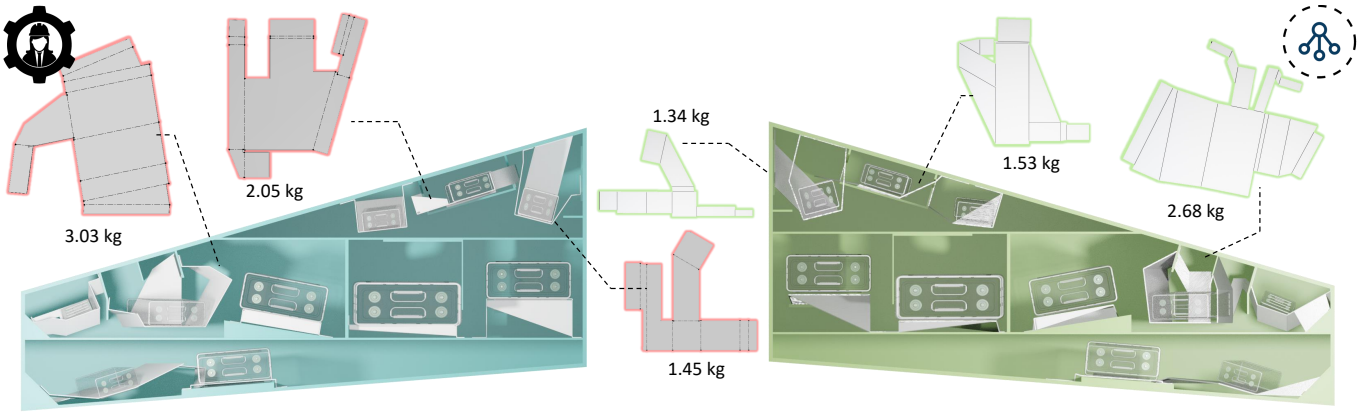


Fig. 1. We present a framework for the automatic generation of load bearing sheet metal parts from high level specifications. This image illustrates an aerospace application of our method: an avionics bay environment with multiple subsystems that require sheet metal parts to connect to the structure. On the left is the result of a domain expert, on the right is the output of our system, with selected flat patterns. Our method is able to fully automate this design task, which otherwise requires significant time and domain expertise. Further analysis shows that our system produced lighter parts and was significantly faster.

Sheet Metal (SM) fabrication is perhaps one of the most common metalworking technique.

Despite its prevalence, SM design is manual and costly, with rigorous practices that restrict the search space, yielding suboptimal results.

In contrast, we present a framework for the first automatic design of SM parts. Focusing on load bearing applications, our novel system generates a high-performing manufacturable SM that adheres to the numerous constraints that SM design entails:

The resulting part minimizes manufacturing costs while adhering to structural, spatial, and manufacturing constraints. In other words, the part should be strong enough, not disturb the environment, and adhere to the manufacturing process. These desiderata sum up to an elaborate, sparse, and expensive search space.

Our generative approach is a carefully designed exploration process, comprising two steps. In *Segment Discovery* connections from the input load to attachable regions are accumulated, and during *Segment Composition* the most performing valid combination is searched for.

For Discovery, we define a slim grammar, and sample it for parts using a Markov-Chain Monte Carlo (MCMC) approach, ran in intercommunicating instances (i.e, chains) for diversity. This, followed by a short continuous optimization, enables building a diverse and high-quality library of substructures. During Composition, a valid and minimal cost combination of the curated substructures is selected. To improve compliance significantly without additional manufacturing costs, we reinforce candidate parts onto themselves — a unique SM capability called *self-riveting*. we provide our code and data in https://github.com/amir90/AutoSheetMetal.

We show our generative approach produces viable parts for numerous scenarios. We compare our system against a human expert and observe improvements in both part quality and design time. We further analyze our pipeline's steps with respect to resulting quality, and have fabricated some results for validation.

We hope our system will stretch the field of SM design, replacing costly expert hours with minutes of standard CPU, making this cheap and reliable manufacturing method accessible to anyone.

Authors' addresses: Amir Barda, amirbarda@mail.tau.ac.il, Tel Aviv University, Tel Aviv, Israel; Guy Tevet, guy.tvt@gmail.com, Tel Aviv University, Tel Aviv, Israel; Adriana Schulz, adriana@cs.washington.edu, University of Washington, Seattle, USA; Amit H. Bermano, amit.bermano@gmail.com, Tel Aviv University, Tel Aviv, Israel.

0730-0301/2023/8-ART116 $15.00
https://doi.org/10.1145/3592444

CCS Concepts: • **Applied Computing** → **Physical sciences and engineering**; • **Modeling and simulation** → *Model development and analysis.*

Additional Key Words and Phrases: Computational Fabrication, Generative Design, Sheet Metal

ACM Reference Format:
Amir Barda, Guy Tevet, Adriana Schulz, and Amit H. Bermano. 2023. Generative Design of Sheet Metal Structures. *ACM Trans. Graph.* 42, 4, Article 116 (August 2023), 13 pages. https://doi.org/10.1145/3592444

1 INTRODUCTION

Sheet Metal (SM) fabrication is a widespread metalworking technique, with uses in many industries, including construction, aerospace, industrial machinery, consumer products, medical implants, and transportation. In its most popular variant, the fabrication process consists of a series of cut and bend operations performed on a flat metal sheet. SM fabrication offers a fast and inexpensive process that wastes significantly less material than other machining techniques, due to the fact that the raw material is a sheet instead of block. Despite its prevalence, SM design has seen scant refinement over the years. Most work in the field aids the designer through interaction and analysis; to our awareness, none automatically suggests how to issue a functional and manufacturable SM part from its requirements. For this reason, SM design is currently laborious, taking hours for an expert engineer to design and test even moderately elaborate parts [Research 2019]. To mitigate design complexity, experts have adopted rigid practices, such as preferring to use right bend angles, which make the design conceptually easier, but doing so restricts the search space and impairs performance, in terms of strength to weight ratio.

In this paper, we introduce a novel generative approach to the design of SM parts. As we demonstrate, this approach yields parts that adhere to the functional requirements. Additionally, our design time was faster and the resulting parts were lighter, compared to those of a SM design expert. Generating SM parts is challenging, with an elaborate search space and numerous considerations that must be taken into account. Hence, we restrict our focus to the fundamental application of *load-bearing fixtures*. Even for this application, however, many requirements must still be respected: minimal material use, sufficient strength to support the given load, collision avoidance with the environment and adherence to SM manufacturing constraints.

Accounting for all requirements produces a complex

and highly constrained search space that induces computationally expensive operations. In addition, the space representation must also be of a mixed continuous/discrete nature, including both a *topological component* (the discrete number and structure of the cutting and bending operations) and a *geometrical component* (continuous bend angles and cut dimensions). More specifically, the SM design process must take into account the following constraints, in addition to considering manufacturing costs:

- **Environment constraints.** The environment where the fixture is placed imposes *collision* (restricted regions) and *connectivity* (attachable regions) constraints.
- **Manufacturing constraints.** To be manufactured, the shape must be *cut from a single sheet* of material and be *sequentially bent* in a collision-free manner.
- **Structural constraints**. Mechanical performance (or strength), which we express in terms of stress and compliance. Note these constraints require *physical simulation* for evaluation.

We posit that such complexity has deterred the automation of SM designs despite its prevalent need.

As a running example for our generative design approach, we consider the prominent setting of the aerospace industry. Here, SM structures are used to connect electronic subsystems to other structural elements of the airframe: an aircraft typically carries tens to hundreds of such custom sheet metal fixtures, riveted to the airframe (see Figure 1)

We apply three key insights to address the challenges of automatically producing SM designs that respect all aforementioned constraints. First, *fixtures are hierarchical compositions of substructures, with each one connecting the load to a point of support.* This insight lets us decompose the problem into two phases, discovery and composition. In the *discovery* phase, we create a library of components that satisfy spatial and manufacturing constraints and are likely to achieve optimal trade-offs between manufacturing cost and strength. In the *composition* phase, we search for a composition of library substructures that meet structural constraints while minimizing costs.

However, building such a library is still difficult, even when considering a single substructure: the search space remains large, and valid solutions are sparse. To manage the search space, our second insight both *exploits the search space structure and decomposes the search process into sparse discrete sampling and local continuous optimization.* For the former, we use stochastic search over a fabrication-oriented grammar; we select a Markov-Chain Monte Carlo (MCMC) approach with a sampling scheme that avoids collisions and a likelihood function that rewards diverse connections to attachable areas. For the latter, we propose a physics-inspired contraction process that respects environmental constraints while minimizing material usage. Together, these steps ensure we populate our library with parts that are *valid* and *locally optimal*.

Diversity, it turns out, is a difficult goal to achieve. Sampling strategies typically introduce bias, tending to find "easier" solutions more often, yielding similar parts, and failing to fully span the space of possible candidate solutions. Accordingly, our third insight is to *jointly search the space for candidates using multiple concurrent MCMC chains that communicate with each other.* Searching for substructures independently typically results in rather simple structures connecting the load to the frame, limiting solution expressiveness. An immediate solution is to allow new sub-structures to spawn from previously found ones, in addition to the load itself.

This approach, however, biases results towards the earlier found structures, again limiting expressiveness and diversity. We hence propose combining both approaches: we run multiple MCMC instances (or *chains*), where each beam allows starting new structures from older ones. Running several chains at the same time helps in avoiding committing to parts of the solution early. To further ensure the different chains yield different results, we encourage searching regions that are already under-explored, across all chains.

We validate system performance on several environments of varying complexity and show that the parts generated adhere to the aforementioned environment, manufacturing, and structural constraints. We further compare our results to those achievable by a human expert, showing that our system yields better expressiveness in terms of generating non-trivial parts with comparatively lower weight and and processing time.

Finally, we demonstrate manufacturability by physically realizing several SM parts designed by the system.

Ours is the first system to establish the viability of a fully automated solution for sheet metal design, exchanging hours of human expert

labor with minutes of standard CPU computation. We anticipate that the system's ease of use could bring the realm of metal sheet design closer to novice users and makers, as well.

The remainder of this paper presents Related Work (Section 2); the Methods we choose for generative design, including specification of our design objectives, design space, and exploration algorithm (Section 3); the Discovery process for segments (Section 4); the Composition process for segments (Section 5); Results (Section 6), which includes evaluation findings as well as design limitations and future work opportunities; and our Conclusion (Section 7).

2 RELATED WORK

Our work leverages ideas from prior work on generative design and procedural modeling.

2.1 Computational Design for Sheet Metal

Due to its widespread use, modern CAD systems (e.g., Solidwork, Onshape, SolidEdge, NX) offer specific tools that guide designers in creating sheet metal parts. These include 3D/2D interfaces for flat pattern design and automatic insertion of relief holes for fabrication. Most research in computational design of SM focuses on algorithms that support these interactive tools [Herrmann and Delalio 2001; Jonsson et al. 2020; Naranje 2010]. There has also been progress on feature detection for SM design to reverse engineer a 3D model into a plan for sheet metal fabrication [Salunkhe et al. 2019].

Initial work considered generative capabilities for sheet metal, but it still requires manual effort and does not target functional parts. [Soman and Campbell 2002] define a 17-rule grammar for SM parts but does not apply it to generative design from a high-level specification. [Patel and Campbell 2008, 2010] use a grammar and a genetic algorithm for automatic design of basic sheet metal parts with certain geometrical goals, but they do not account for functionality objectives.

Our method, on the other hand, offers an end-to-end solution for generating complete sheet metal fabrication plans (i.e., flat patterns) that can be directly employed since the resulting parts are optimized for fabrication and comply with requested physical performance specifications. Our approach decouples the exploration and optimization phases, which lets us concentrate on just a single objective (i.e., making connections) during the exploration phase, making optimization more feasible, robust, and efficient. This mono-objective exploration formulation makes genetic algorithms, such as [Yuan et al. 2014], used in previous works ineffective, as there is no "Pareto front" to be found.

2.2 Generative Design

Generative design is the programmatic approach to synthesizing a design given a user's specifications of how the resulting model should perform. Fully automated design has been a long-stretch goal in computer-aided design over the past several decades. One of its fundamental challenges is constraining the search space to manufacturable artifacts. Further, different fabrication methods impose design constraints that can be complicated to navigate. Recently, significant advances in generative design are being prompted by the increased popularity of additive manufacturing; the freedom of form of this manufacturing technique has enabled successful application of topology optimization [Bendsoe and Sigmund 2013; Liu et al. 2018], an active research area that is extending the capabilities of generative design in terms of materials, resolution [Zhu et al. 2017] and functionality [Du et al. 2020]. Notably, [Zhu et al. 2017] also deals with design space exploration in the voxel domain to design parts for 3D printing, and give a space exploration procedure that also relies on a combination of stochastic and continuous optimization steps that are run iteratively. Moreover, similar to our approach, their stochastic step employs a formulation that combines a term that promotes samples with desirable properties multiplied by another term that penalizes over-sampled regions of the design space, thus promoting diversity. Unfortunately, the development of generative approaches for some manufacturing methods are lagging far behind others. While some research progress is being made in subtractive methods (such as CNC machining [Morris et al. 2020]) as well as in generative design of millable or castable topologies (such as Frustum's GENERATE) for commercial products, these methods do not directly extend to forming processes such as those used for sheet metal.

2.3 2D and 3D Fabrication

Sheet metal creates 3D shapes using bends from a 2D flat pattern. Our work is therefore related to the vast literature on 2D to 3D fabrication. Most prior work in this area takes as input a 3D shape and searches for an optimal decomposition into 2D planes for fabrication [Chen et al. 2013; Guseinov et al. 2020; Hildebrand et al. 2012]. Some techniques here include computational geometry algorithms in the origami domain [Demaine 2006; Demaine and Tachi 2017], which can even produce designs that automatically fold [Felton et al. 2013]. However, such methods start from a 3D shape, not from high-level specifications of functionality directly exploring the manufacturing space. Though some recent work performs optimization for functionality using 2D to 3D fabrication [Chen et al. 2020; Rus and Tolley 2018], it requires user assistance because it cannot automatically explore the mixed discrete and continuous spaces of 2D to 3D fabrication.

2.4 Shape Grammars

Closest to our work are computational design tools that use grammars to jointly search a mixed discrete and continuous domain. [Ertelt and Shea 2008] define a shape grammar for CNC; [Wu et al. 2019] define a carpentry design space as a shape grammar and explore it using a genetic algorithm; [Sass 2006] defines a shape grammar for house construction from plywood sheets and applies it to automatically subdivide the completed frame of the house into a manufacturable production of the grammar; [Schulz et al. 2017] develop a grammar for robot fabrication that dictates composition rules. Our work is similar to these approaches since it uses a grammar for sheet metal, but we propose unique solutions that can efficiently navigate the complexity of our domain.

3 METHODS

Figure 2 shows a system overview of our approach. Generative design inputs include an environment and a load to be placed in

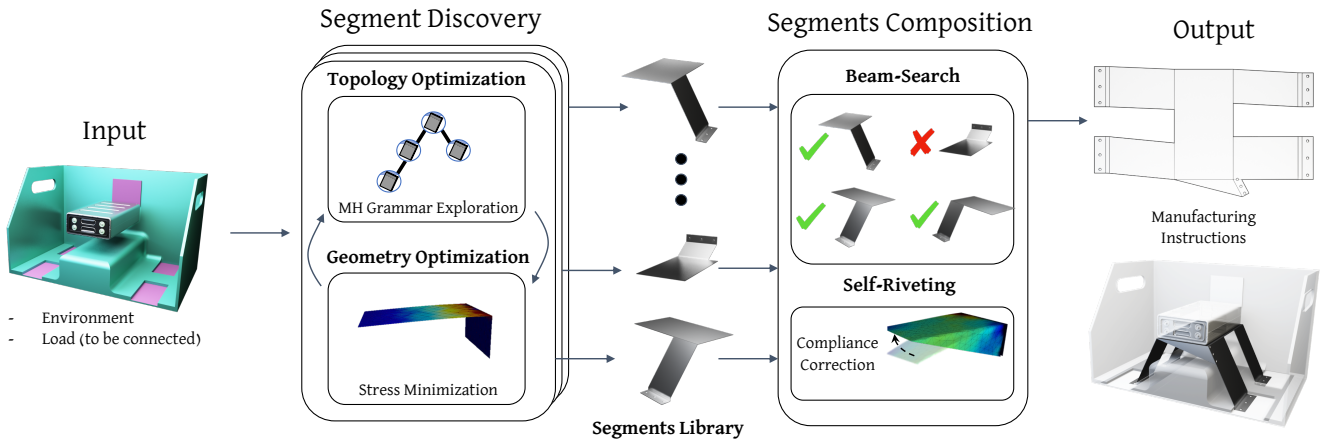
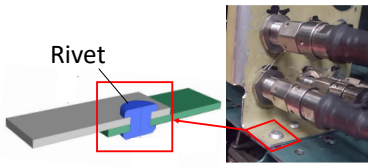


Fig. 2. System overview. The input to our system (left) includes the specifications of a subsystem (the input load) and the environment in which it should be mounted. A Metropolis-Hasting-based optimization traverses our shape grammar in alternating discrete/continuous steps (Segment Discovery), populating a library with physically valid, material-efficient potential segments connecting the load to regions of support. Compositions of parts from the library are then considered using a Beam-search approach (Segment Composition). The final result is the part that uses the least material while adhering to the physical and manufacturing constraints (right).

it; the output is a manufacturing plan for a functional sheet metal part connecting the load to the environment while adhering to spatial/environment, manufacturing, and mechanical performance constraints. The desired part should be manufactured from a single sheet, without welding. Therefore, manufacturing instructions consist of a flat pattern and a set of bend operations that should be performed in a given order.

The input environment is represented by a 3D model defining obstacles to avoid and rivet areas, i.e., a subset of the mesh surface that defines the places where sheet metal fixtures can be mounted. As shown in the inset figure, the process of riveting can only happen in designated areas and therefore must be specified by the engineer as part of the input. The load is represented by a 3D box in the environment and the external force applied on it.



3.1 Design Objectives

Engineers account for many different, and often conflicting, requirements when designing SM parts. Therefore, a fundamental step in developing an automatic design strategy involves identifying and analyzing these desiderata. This work formulates these requirements in four categories:

environmental constraints (C_E), manufacturing constraints (C_M), manufacturing objectives (O_M), and structural constraints (C_S).

- **Environment Constraints (C_E).**
  The sheet metal fixtures must be connected to pre-defined allowable rivet areas while avoiding collisions with restricted regions in the environment.
- **Manufacturing Objectives (O_M).** To reduce manufacturing costs, the part should minimize the amount of used material and the number of bends.
- **Manufacturing Constraints (C_M) .** We prohibit overlaps in the flat pattern and ensure that the part does not collide with itself during manufacturing and bending. We further limit the minimal/maximal tab size and bending angles according to the specifications of the bending and cutting tools.
- **Structural Constraints (C_S).** To ensure structural integrity, the stress and compliance values throughout the part must be kept below thresholds defined by the specifications. namely, we use the *margin of safety (MOS)* and *compliance* of the part as metrics. The required engineering definitions are given in the supplementary materials.

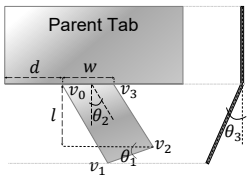
This approach formulates the problem as it is typically addressed by engineers: manufacturing cost is optimized, while other considerations are treated as strict requirements. Further, and importantly, decoupling the different considerations lets us define an efficient exploration strategy over such a tightly constrained space. We can define a design space such that solutions that satisfy C_M can be directly sampled; C_E can be checked on a partial design (when only a subset of the sheet metal part has been discovered), while C_S alone can be evaluated on the full design; and (O_M) can be measured additively. These inform our decisions for design space representation and exploration strategy, which we outline next.

3.2 Design Space

As is popular in the literature [Schlecht et al. 2007], [Ripperda and Brenner 2009], we represent our exploration space using a context-free grammar. The hierarchical derivations in a grammar tree correlate well to chains of bending operations on the SM parts. To reduce the search space as much as possible, we streamline popular grammars. Ours consists of only one type of node, called a *tab*,

which represents a region that remains flat in the final SM part, and that emanates from an edge of another tab at a specified angle. By explicitly defining the bending order, the grammar maps directly to manufacturing instructions, which enables immediate checkers for all manufacturability requirements (C_M).

We use the following six parameters to define tab geometry (see inset figure): d, w, l, θ_1, θ_2, θ_3, representing, respectively, the starting point on the parent edge, tab width, tab perpendicular length, and three rotation angles with respect to the parent tab, i.e., in-plane rotation angle of the tab end, in-plane rotation angle of the tab itself, and off-plane rotation angle. These six parameters, though minimalistic, can express a trapezoidal tab and hence, we argue, are sufficient.



Each tab (excluding the initial, or "base" tab) is connected in a tree structure to its parent through an edge and adds three additional edges to the part, which can serve as parents to other tabs. We do not enforce a single child per edge since some environments require more (see Figure 1).

Fig. 3 shows an example of a grammar tree and the associated part. Upon creation, tab parameter values are sampled from a predefined probability prior distribution (see Section 4). For simplicity, we initialize the root of the grammar to be a rectangular tab with 4 free edges that is attached to the bottom of the load, we call this tab the base tab. In general, there is no requirement for the base tab to be a rectangle, and our algorithm remains valid for any other polygonal shape.

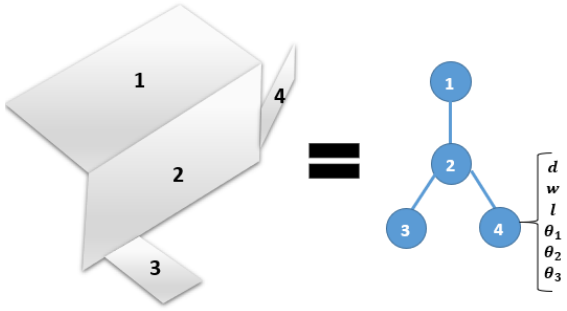


Fig. 3. Shape Grammar. Each grammar tree (right) corresponds to a sheet metal part (left). The tabs of the part correspond to nodes in the tree, with corresponding connectivity. Each node stores the tab geometry, represented by the grammar parameters.

3.3 Part Generation Algorithm

Due to the many design space constraints, a fundamental challenge with searching the space for an optimal design is finding feasible solutions on one hand, while fully searching the space on another.

We address this difficulty by applying the insight that sheet metal fixtures are composed of segments (sequences of tabs) that connect the load to the environment. In addition, we note that segments are monotonic with respect to strength, i.e., each added segment necessarily reinforces the fixture, rather than weakens it, in the same way adding a spring the a system of springs in parallel can only increase the total stiffness. This means that segments can be identified independently and composed afterwards. Hence we perform our exploration scheme in two phases: the segment discovery step, followed by the segment composition one. For additional clarity, we provide pseudo-code for the main components of the algorithm in the supplementary.

Segment Discovery. The goal of this step is to build a library of segments that expresses the solutions of interest, i.e., it should both explore the space of diverse segments and converge to segments that are likely to be useful for composition. We address this by alternating a *stochastic sampling method* that explores the space with a *local optimization technique* that pushes the samples toward high-performing solutions.

Our stochastic sampling strategy searches over the mixed discrete-continuous space defined by the grammar using a Metropolis-Hastings (MH) sampler. We find that a direct implementation of this strategy (as given in [Talton et al. 2011] does not yield samples that span the search space well; directly searching for segment grammar trees that connect the load to the rivet areas favors simple solutions, although, as we show in our results, more elaborate solutions with multiple branching connections are often required to satisfy the engineering requirements. We hence propose allowing new segments to emanate from already found ones, thus encouraging the desired branching behavior and better spanning our search over the entire space.

This accumulative approach, though, also turns out to be sub-optimal; it favors solutions that emanate from the earlier found segments, regardless of their quality. We hence propose a middle-ground strategy. We run multiple instances, or *chains* (MH search processes). Each chain collects segments while allowing new segments to grow out off previous ones found by same chain. This both encourages the construction of elaborate non-trivial segments and alleviates the bias introduced towards early found connections. To promote diversity even further, we allow the MH chains to share information about their segments with each other (The information shared is defined precisely in Eq. 1 and Eq. 2), encouraging the chains to omit regions that have already been connected too many times in other chains.

To account for the sparsity of the solution space, the sampling process considers only the combination of constraints and priors to ensure solutions that satisfy segment-level constraints (accounting for C_E and C_M, safe for a final collision detection step after composition). We then optimize over the continuous parameters of the sampled segments to find solutions that are likely to contribute to global part-level considerations once composed (O_M and C_S). We achieve this by modeling the segment as a non-linear spring system and running a physically inspired *spring retraction process* that (1) uses heuristics motivated by domain knowledge to minimize material usage while improving structural integrity and rigidity, and (2) adheres to manufacturability constraints.

Segment Composition. Given the Segment Library, the Segment Composition phase seeks a subset of the library that optimally fulfills the desiderata, i.e., it is valid (e.g., does not intersect with itself and adheres to structural integrity constraints) and uses minimal material. To do this, we employ a beam-search inspired search over

the Segment Library (Section 5.1), giving the user a single parameter (number of iterations) to trade-off quality with run-time in the large $O(2^N)$ part pool search space. Finally, we exploit a unique feature of sheet metal and add self-riveting (Section 5.2), both intra- and inter-segment (see Fig. 6). This step dramatically improves the ability to adhere to compliance constraints (C_S) with less material and fewer bend operations (O_M).

4 SEGMENT DISCOVERY

During segment discovery, our sampling scheme finds segments that can potentially be composed later into a single high quality part. The found segments should individually adhere to all constraints (Section 3.1), span the space of possible segments well, and perform optimally. To address this, we propose alternating a stochastic sampling approach that encourages diverse solutions with a continuous optimization that seeks local maximum performance. The sampling step sends out thin (meaning, having the minimal manufacturable w, as set by the user) and long 'feeler' segments in an attempt to make connections to rivet areas, resembling the role of feelers in nature, the feelers grow in the sampled directions until they collide with the environment or with volume bounds. From these, new feelers continue growing until a connection is found. To increase exploration diversification, we use multiple beams that communicate with each other. . For each found connection, the continuous geometry optimization (Section 4.2) seeks maximal strength and minimal material usage. We employ spring energy to contract the overly long feeler while avoiding collision with the environment. Finally, the widths are determined for the segment tabs, and redundant planar tabs are merged.

To avoid overloading the library with similar segments produced by different MH beams, and to eliminate bias towards easier-to-reach rivet areas, an optimized segment is only added to the library given a certain selection probability. That probability is determined according to the following criteria.

Segment similarity score. We score a candidate segment according to its distance from all segments already in the library. We define the distance between two segments as the distances between their root and end edges, which connects them to the rivet area.

Regularization. Additionally, we add a regularization term to prefer segments for which the root (i.e, first) and connecting (i.e, last) edges are more colinear and that have fewer tabs. We find that these segments make more visual sense. If the segment being evaluated for optimization has a score that is similar to an existing segment in the library but a lower regularization score, we replace that segment with the segment under evaluation.

After the optimized segment is added to the library, new feelers can emerge from the segment's free edges as well as the base tab, allowing tree-like hierarchies to form. This alternating MH exploration and geometry optimization runs M parallel beams, with N iterations for each beam. In our experiments (Section 6), we set N to 500 and M to 10.

4.1 Reversible Jump Metropolis-Hasting (MCMC) Algorithm

Metropolis-Hasting (MH) is an MCMC algorithm used to obtain random samples that converge to a required distribution. Since our search space difficult to cover, such an approach is required. Furthermore it was shown that MCMC lends itself well to grammars [Ripperda and Brenner 2009; Schlecht et al. 2007]. We employ *Reversible Jump MCMC* [Green 1995] since it lets us vary the dimensionality of samples along the progression of the search, as proposed for grammars by Talton et al. [Talton et al. 2011]. To promote efficient and varied exploration, we adapt this algorithm to our problem setting.

The MH framework we employ [Talton et al. 2011] suggests a sampling approach that evolves tree structures based on grammar rules. Given a current grammar tree configuration, the MH sampler performs one of the following operations, with some selection probability:

- **Diffuse move:** a local move, where a feeler (i.e, not part of some optimized segment in the Segment Library) tab is chosen and its parameters are re-sampled. Note that this might induce a change in geometry to more than just the tab itself if the feeler is not a leaf in the grammar tree.
- **Jump move:** a global move, where a some free edge of a feeler tab is chosen and its sub-tree is deleted, and a new sub-tree is re-sampled. to maintain validity, we check collision with the environment, the load and the part's already optimized segments, and remove all tabs in the subtree which are not valid. Note that this is the main reason we opt to not sample, but rather "cast", the tab length, as random lengths would create a high number of removed (i.e, rejected) tabs in the subtree due to the physical constraints.

The sampling process is governed by a likelihood function f, which indicates the probability that a current configuration will be chosen. This function should promote both the *amount* and *variety* of connections. To promote connections with the rivet areas in the exploration phase the policies defined are biased towards getting closer to the rivet areas. As a proxy for finding connections, we measure the distance from the currently added tab to each rivet area. To promote variety, we penalize connecting to rivet areas that are popular across all the MH chains we ran simultaneously. We propose the following likelihood function:

$$f = \sum_{i}^{\#RA} \sum_{j}^{edge(T)} e^{-\alpha \cdot dist(edge_j, RA_i)} \cdot \left(1 - \frac{c(RA_i)}{(1 + \sum_{RA_i} c(RA_i))}\right)^{\beta} \tag{1}$$

where RA_i is the i^{th} rivet area, $edge_j$ is an edge of the unoptimized (feeler) tab, $dist$ is the shortest distance between each rivet area and $edge_j$, and c is a function that counts the number of existing connections to RA_i across all beams. α and β are hyper-parameters, which we set to 1 and 3, respectively. Eq. 1 gives the unnormalized probability of a production of the grammar (i.e, a part) We use this unnormalized probability as an input to the MH sampler. The first term in the likelihood function rewards parts that have edges close to a rivet area, and the second term scales the reward down by rivet area "popularity". The jump move requires an additional likelihood

function for the selection of the tab to remove. We use a function similar to f:

$$pe(T) = \sum_{RA_i} e^{-\alpha \cdot dist(pe(T),RA_i) \cdot se(T)} \cdot (1 - \frac{c(RA_i)}{(1 + \sum_{RA_i} c(RA_i))})^{\beta} \tag{2}$$

where $pe(T)$ is the edge connecting T to its parent and $se = 1 - \frac{\#selected(pe(T))}{\#steps+1}$, where $\#steps$ is the ordinal number of the current exploration step. Eq. 2 gives the unnormalized probability of selecting an edge in the existing part which is then normalized over all the free edges. We add se to the first term as a scaling factor that prefers fewer selected edges.

When generating a new tab, all its values are randomly assigned except for its width w, which is minimal (since it is in feeler form), and its length l. For the latter, we extend the feeler until it collides with the environment or reaches a maximal length. This design choice reduces the dimensionality of the search space, and does not adversely effect the results, as the geometry optimization minimizes l back.

4.2 Geometry Optimization

Our exploration procedure discovers connections from optimized tabs to rivet areas. These connections are usually impractical for manufacturing in terms of length and width. During optimization, the segment's total length decreases, which decreases total material usage. We now make a domain knowledge motivated distinction between the length and width of a segment. In SM fixture design, it is good practice to maximize the connections to the rivet area, due to the fact that material in the width dimension can be reduced using internal cutouts in the tab without affecting the segment validity (i.e it still connects to the rivet area). This means that in practice, A wider segment usually means a stronger part without increased material costs. Of course, the segment must still respect spatial, environment, and other constraints (Section 3.1) during optimization. To summarize, the geometry optimization is split to steps - length minimization and width maximization. for length minimization, direct gradient optimization approaches over tab parameters are unfitting due to factors such as collision avoidance with the environment, which is typically a complicated mesh. We have also found that Derivative-free optimizations (such as those offered by the NLopt library) are not sufficiently robust or efficient for our case.

To overcome this, we propose a spring-based process.Modeling the segment as a spring network, with the rest length being the minimal allowed tab length, l_{min}, and simulating its contraction under the spring forces. Note that modeling the segment as a regular spring network minimizes the sum of squared tab lengths rather than the overall segment length. Therefore, we make a simple correction to the spring constant and define it as $k_{base}/(l_i - l_{min})$for each plane i in the segment. From Eq. 3, we get that the energy minimized under this formulation becomes the sum of the lengths (and not *squared lengths*). This approach is simple to implement using existing physics frameworks (e.g., Nvidia's Physx) and naturally handles intersections while maintaining a valid state in every time step. In this way, our spring model becomes essentially a force-density model, as defined in [Schek 1974].

$$\begin{aligned} E_{segment} &= \frac{1}{2} \sum_i k \cdot (l_i - l_{min})^2 \\ &= \frac{1}{2} \sum_i \frac{k_{base}}{(l_i - l_{min})} \cdot (l_i - l_{min})^2 \\ &= \frac{1}{2} K_{base} \sum_i (l_i - l_{min}) \end{aligned} \tag{3}$$

The width maximization step is simpler and performed by increasing w and decreasing d for each segment until an intersection is reached or the segment is fully expanded to its parent edge width, as depicted in Figure 4.

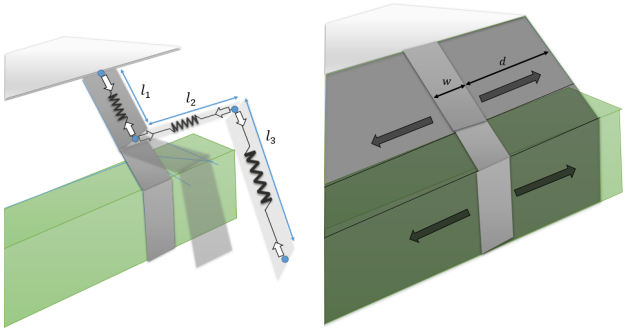


Fig. 4. Geometry Optimization. Left: Step (1). The segment is modeled as a constant force spring network and simulated until convergence. Right: Step (2). The segment's d and w are decreased and increased, respectively, until intersection with the environment or root edge width is reached.

5 SEGMENT COMPOSITION

Although each chain can produce a complete SM part, the complexity of the heavily constrained, discrete-continuous optimization space pushes it regularly toward local minima (see ablation in 6.4.2). To overcome this, we collect all discovered segments from the different chains into the Segment Library. When the exploration procedure ends, we can compose a subset of the Segment Library's nodes to create a part. This subset is found in a global, yet inexpensive search process over the Segment Library (Section 5.1). Then, if necessary, the recomposed part can be reinforced with self-riveting (Section 5.2) to increases compliance.

5.1 Searching the Segment Library

To avoid local minima in the SM-elaborated parameter space, we compose the output sheet metal part from the segments generated in the Segment Discovery phase. We search for a subset of segments in the Segment Library that minimize material use and comply with the constraints specified in Section 3.1. To satisfy structural constraints, we must run costly physical simulations for each candidate subset. Hence, an exhaustive search (involving $2^{O(N)}$ FEM calls) is not feasible, even for a small Segment Library. Instead, we adapt a *beam-search* inspired algorithm, originally proposed for computational linguistics, that suggests a middle ground between greedy and exhaustive search. Beam-search lets the user to control the beam

width, a single parameter trading off quality with run-time. The search algorithm is detailed in the supplementary materials.

Fig. 2 shows a typical recomposition case, while Fig. 5 provides more complex and elaborate examples.

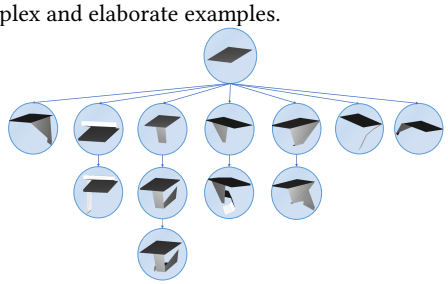


Fig. 5. Example Segment Library. The library can be visualized as a tree, where each node represents an optimized segment and the root is the base tab. Note that some segments are built on previous ones, and nodes originate from different MH beams. Selecting a subset of tree nodes defines a potential part.

5.2 Self-Riveting

It is possible to increase a part's rigidity by using a unique feature of sheet metal design, which we call *self-riveting*. This involves bending a part to be flush to itself so overlaps can be riveted together. Joining an edge with high compliance to one with low compliance increases overall part rigidity.

We employ self-riveting during beam-search when a candidate part's values have a sufficient margin of safety (MOS) but its compliance is too high. We calculate a visibility graph between every two free edges in the candidate part. Two edges are defined as visible to one another if their 4 endpoints are on the same plane and their parent tabs are not.

According to the visibility graph, we add self-riveting to connect low-compliance (rigid) edges to a high-compliance edge, Fig. 6 shows this process. We refer the reader to the supplementary materials for a more detailed explanation of this step.

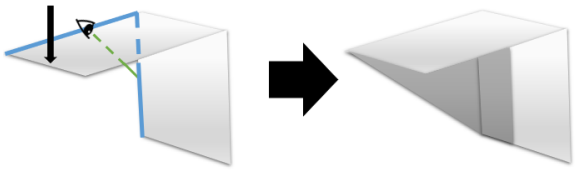


Fig. 6. Self-Riveting. For each free edge, visibility is calculated to all other free edges. The system attempts to connect high- and low-compliance edges to reduce overall part compliance. This stage occurs after geometry optimization if the MOS of the part is admissible but its compliance is not.

6 RESULTS

6.1 Sheet Metal Part Generation

We test our pipeline on several environments of varying complexity. We define our SM material to be Aluminum 2024T3 with a thickness of 0.063" (1.6 mm), which is widely used for aerospace applications. For all runs, we set the number of concurrent chains to 10 and the number of exploration iterations to 1000, after which the Segment Library is searched for an optimal composition of segments to make a valid part. Supplementary materials present a full list of parameters for each experiment.

We summarize results in Fig. 7, showing how our method can generate manufacturable sheet metal designs that satisfy the user-specified structural requirements for models with varying levels of complexity. As shown in the figure, our method can effectively overcome challenges such as obstacles (g, h) and rivet planes with sharp angles (c). Each of these results took 10-20 minutes to generate.

Fig. 8 shows a more comprehensive run of our method, with compliance, MOS, and amount of material given as a function of iteration number. Note that as the number of exploration iterations increases, we achieve convergence of the results. This is due to our Segment Library, which tracks potential segments found during exploration and prevents our pipeline from getting stuck at local minima.

6.2 Expert Comparison

To provide a reference for the performance of the system we compare our results to those achieved by a domain expert. We gave an SM design engineer (an avionics bay with 10 positioned input loads and marked allowable rivet areas, as shown in Fig 10. The engineer was asked to design SM parts for installing each input load using the allowed rivet areas. We then used our pipeline on the same scenario. Results show that we significantly reduced weight while finishing in much less time than the expert. The table to the right of Fig 10 summarizes cumulative findings. We provide a thorough explanation of the comparison procedure as well as a per input load comparison in the supplementary materials.

6.3 Fabrication

To demonstrate that our system generates designs usable for SM production, we fabricated 4 results that were computed with our tool. Fabrication material is Aluminum 2024, at 0.0063". Minor post-processing was performed for manufacturing, such as adding relief holes at bend corners (to avoid causing the material to crack during the bend); this post process is well defined given a part geometry and is supported by standard CAD systems. Fig. 12 shows examples.

6.4 Ablation Studies

We present three ablation studies. Each demonstrates the necessity of a specific system component

6.4.1 Segment Optimization and Edge Selection Policies. Section 4 presented our policies for selecting an edge to explore and a segment to optimize. To demonstrate the necessity of these policies, we compare exploration results using two naive policies: greedy and uniform. For the edge selection, the greedy policy always picks the edge that is closest to a rivet area, and the uniform policy picks uniformly from all the part's free edges. For the Segment Optimization, the greedy policy always optimizes a segment when one is available for optimization, and the uniform policy optimizes a segment uniformly at random from all those available for optimization. We create a maze-like environment for this test, shown in Fig. 13(a). This environment has 5 rivet areas arranged in a sequence away

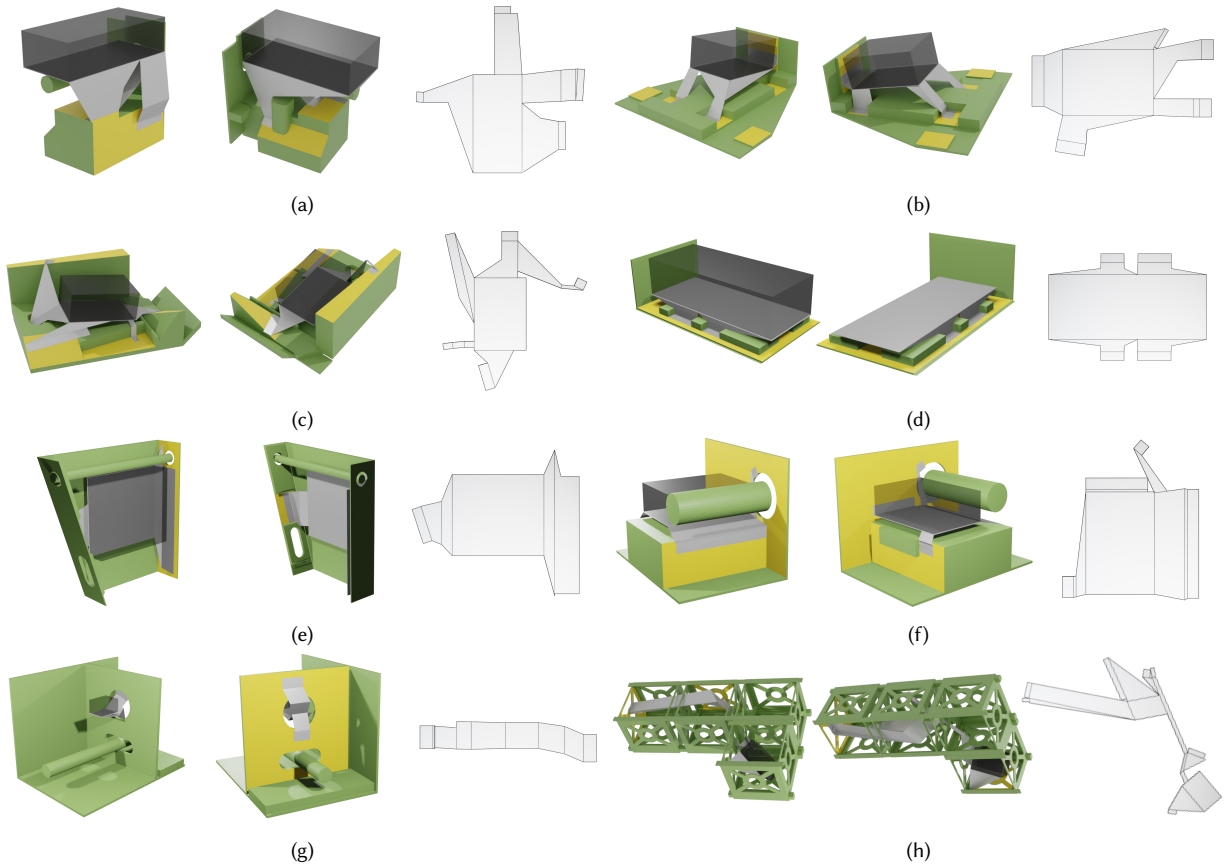


Fig. 7. System results for a variety of environments and constraints, with the flat pattern shown to the right. For the given scenarios, our pipeline finds valid sheet metal part designs that use the minimal amount of material from the connections discovered during the exploration procedure, especially for complex environments with many obstacles such as in (h), and environments with that have very angular surfaces, such as in (c), which are difficult to design for with traditional CAD tools.

from the input load, where the near rivet planes are easier to reach during exploration. For each policy configuration (segment greedy, segment uniform, edge greedy, edge uniform, ours), we run our pipeline's exploration procedure for 500 iterations and compare the resulting Segment Library size for rivet areas 1, 3 and 5. Furthermore, to study the quality and variety of the resulting segments, we use t-SNE analysis for clustering; The features used for t-SNE clustering are the midpoint positions of the root and connecting edge of the segment, the cumulative value of l for the segment and the cumulative absolute values for θ_2 and θ_3. These features define a segment geometrically, and therefore will cluster geometrically similar segments.13(b) shows the number of connections per rivet area. We observe that for the greedy and uniform segment optimization policies, we quickly reach the maximum number of connections per rivet plane (capped at 10 for this test) but get repetitive or poor quality connections, as shown in the t-sne analysis in Fig. 13(c-d). For greedy or uniform edge selection policies, we get ineffective exploration results, with no connections made in the more challenging rivet area 3. Our policies are tuned to achieve effective exploration without bloating the Segment Library with redundant or low-quality segments, which hinder the run-time and output quality of the Segment Composition phase.

6.4.2 Segment Composition. As we note in Section 5, each chain can produce a complete part, but its tendency toward local minima make it insufficient. We demonstrate this for a given scenario. We run the exploration phase for 500 iterations, with 10 parallel chains. Fig. 14 shows the parts generated for each chain, and the part generated using composition. We observe that the chains tend to exhibit self-differentiation behavior, where each chain focuses on a different "area" of the environment. This is due to our segment optimization policy, which we designed to avoid bloating the Segment Library with identical segments from different chains. In general, depending on the environment's complexity, not all chains might contribute to the library. For the example shown, 9 chains were sufficient, as one chain did not make any connections. This localized behavior of individual chains motivates the Segment Composition phase, which helps search over all segments to compose an overall better part.

6.4.3 Self-Riveting. In Section 5.2, we counter-intuitively claim that adding self-riveting reduces material use. To demonstrate this, Fig. 9

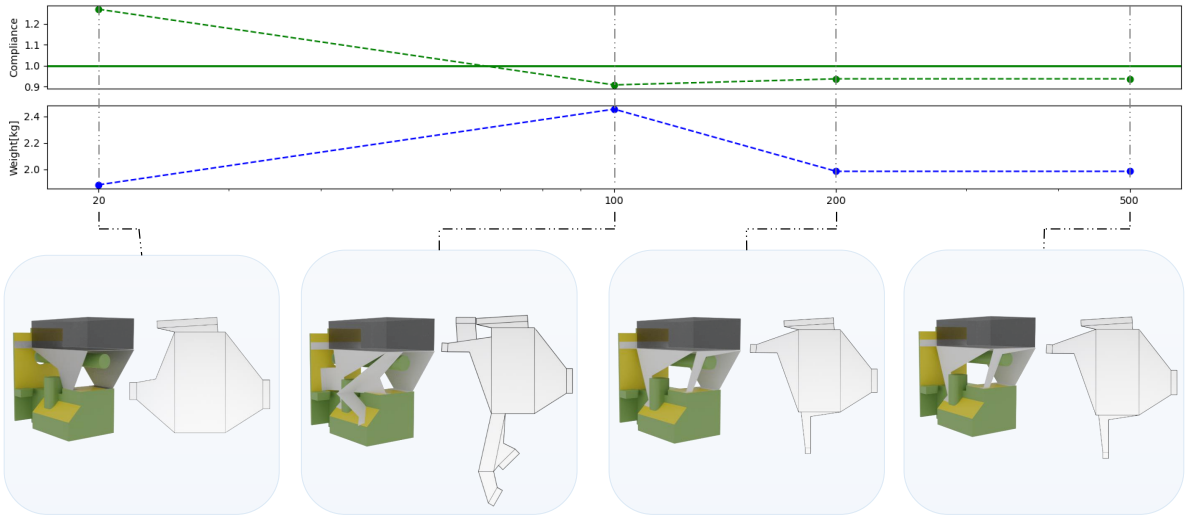


Fig. 8. A comprehensive run of our pipeline for a given scenario. Due to our exploration procedure, which commits a potential connection to the Segment Library, our pipeline avoids local minima and generally achieves convergence after a certain number of exploration iterations. The graphs above show the part weight and compliance as a function of exploration iterations (the solid green line is the maximum allowed compliance).

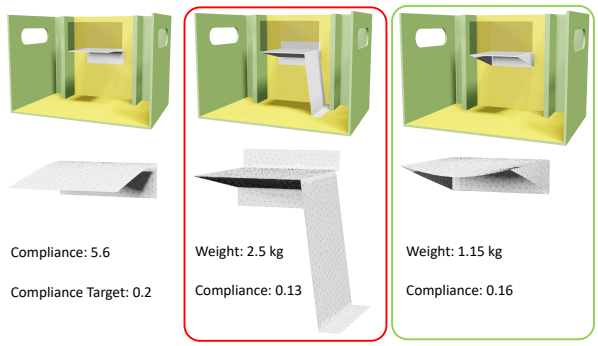
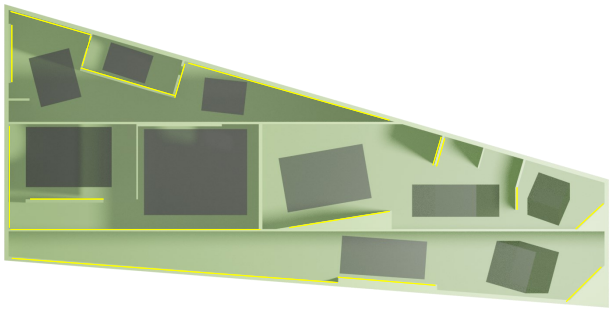


Fig. 9. Self-riveting ablation: (left) the input load is close to a wall but far away from the floor. Naively connecting to the wall produces a springboard-like structure, which naturally has high compliance. (middle) without self riveting, the it is necessary to make a costly connection to the floor in order to lower the compliance to a valid amount. (right) with self riveting, the part can be reinforced to itself, drastically lowering the compliance with a comparatively small addition of weight.



	Time $[hrs]$	Material $[m^2]$
Expert	12.45	5.169
Ours	3.65	4.467
% change	**-70.6**	**-13.5**

Fig. 10. **Top**: The avionics bay environment with 10 input loads. **Bottom**: Cumulative results for all input loads; note the large time difference between the domain expert and our pipeline.

introduces a light load close to a wall. The immediate connection to the wall can satisfy compliance conditions only with self-riveting; otherwise, expensive connections to the floor are needed instead.

7 CONCLUSION

In this work we present, to the best of our knowledge, the first fully automatic tool for the generation of functional, load bearing sheet metal parts from high level specifications. This design problem has a challenging combined discrete and continuous nature, and has remained relatively unexplored. Our framework takes advantage of the insight that the problem can be divided in to exploration and composition phases. We note that this structure characterizes a variety of other connection finding problems, such as designing efficient 3d printed supports and plastic sprues, on which our framework can be applied on with minimal, domain specific changes. Therfore, we believe our framework can be used to make feasible a class of previously underexplored design problems.

7.1 Limitations and Future Work

As previously discussed, our method terminates once it discovers a solution that complies with the user specifications. While the beam-search over the Segment Library aids in avoiding local minima, there

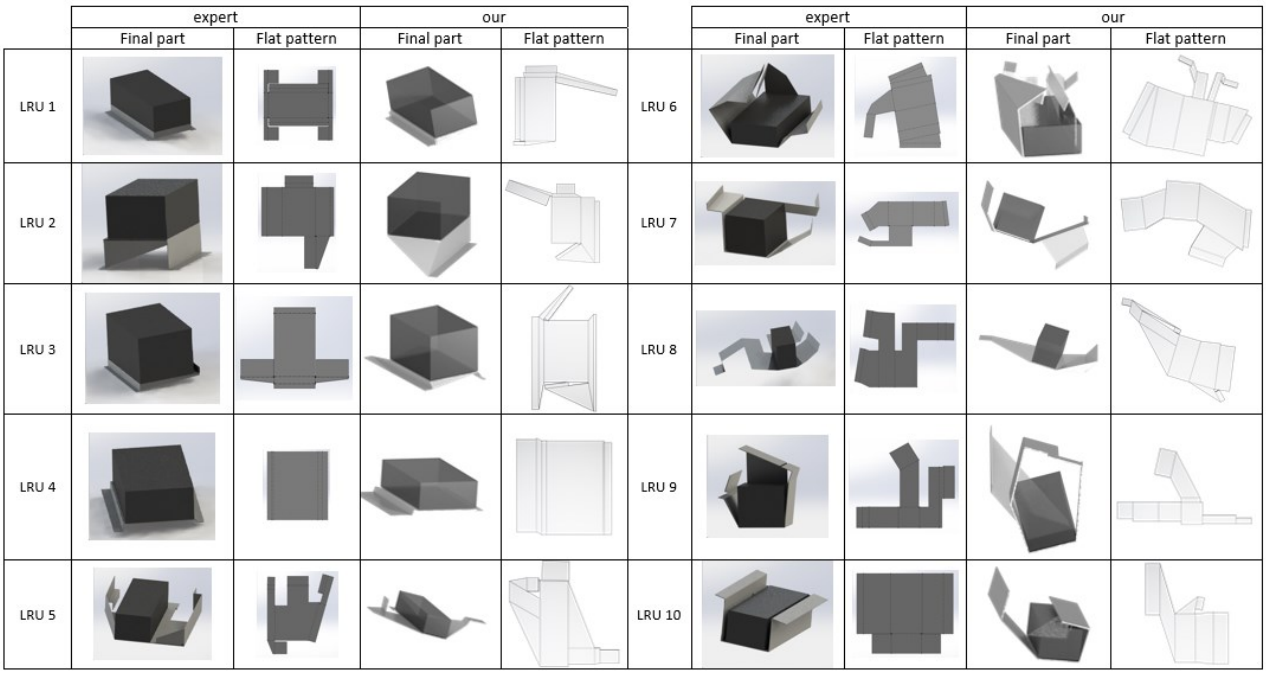


Fig. 11. The final parts created by the domain expert and our system, and the resulting flat patterns. The CAD tool used by the expert does not allow for easy handling of non-right bend angles, leading to non-optimal segments. Our system creates more elaborate," flat patterns due to our geometry optimization, allowing for lighter valid solutions.

are no guarantees that our solution will be a global optimum, like most optimization methods over combinatorial spaces. Additionally, due to the challenge of recruiting participants with equivalent expertise, we provide a reference comparison against a single expert. Although this comparison serves as a reference of our system's capabilities, claiming that the system outperforms human sheet metal engineers in general would require conducting the comparison on a large amount of experts. In practice, though, we have found that our method successfully produces viable results consistently. A computational bottleneck for our method is the depth of the grammar tree and maximum number of children we allow during discovery. In the future, it would be interesting to conduct a thorough scalability evaluation that extends our method to enable deeper graphs, which could be useful outside the typical industrial settings. One potential application for this would be artistic design.

Our work also leaves room for future research on expanding the capabilities of our design system. Expanding the grammar to include other specialized sheet metal operations – such as joggles, curls, or deep drawing – would enable a larger design space that is relevant to more applications and fields. Another interesting direction could consider different ways of exploring conflicting objectives with multi-objective search methods. This would help designers explore trade-offs between manufacturing cost and physical behavior instead of treating the latter as a set of hard constraints.

We further note that our method can be adapted to other connection-finding design problems, such as plastic sprues and 3d printing supports.

Finally, to cope with elaborate and complex scenarios, the system could benefit from an enriched understanding of its environment. Global semantic understanding could be incorporated, e.g., as better, more meaningful priors given to the MCMC searching algorithm (instead of the current uniform ones). A suitable way to deduce such semantic awareness includes, of course, deep neural networks: designs produced by our method could be a starting point for the generation of an SM part dataset for deep learning. Such a dataset could open avenues for even faster convergence and far more elaborate scenarios.

REFERENCES

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

Desai Chen, Pitchaya Sitthi-amorn, Justin T. Lan, and W. Matusik. 2013. Computing and Fabricating Multiplanar Models. *Computer Graphics Forum* 32 (2013).

Wei-Hsi Chen, Shivangi Mishra, Yuchong Gao, Young-Joo Lee, Daniel Koditschek, Shu Yang, and Cynthia Sung. 2020. A Programmably Compliant Origami Mechanism for Dynamically Dexterous Robots. *IEEE Robotics and Automation Letters* PP (01 2020). https://doi.org/10.1109/LRA.2020.2970637

Erik Demaine. 2006. Origami, Linkages, and Polyhedra: Folding with Algorithms. 1. https://doi.org/10.1007/11841036_1

Erik Demaine and Tomohiro Tachi. 2017. Origamizer: A Practical Algorithm for Folding Any Polyhedron. https://doi.org/10.4230/LIPIcs.SoCG.2017.34

Tao Du, Kui Wu, Andrew Spielberg, Wojciech Matusik, Bo Zhu, and Eftychios Sifakis. 2020. Functional optimization of fluidic devices with differentiable stokes flow. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.

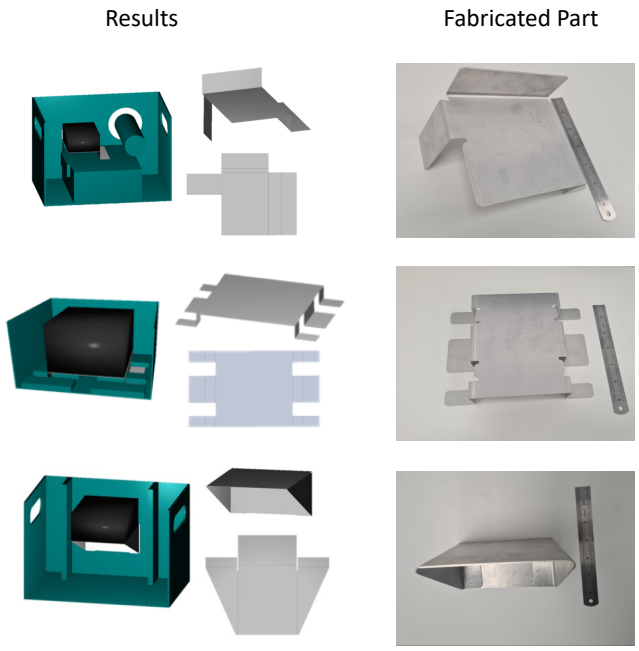


Fig. 12. Fabricated results.

Christoph Ertelt and Kristina Shea. 2008. Generative Design and CNC Fabrication Using Shape Grammars. *Proceedings of the ASME Design Engineering Technical Conference* 289 (01 2008). https://doi.org/10.1115/DETC2008-49856

Samuel Felton, Michael Tolley, C.D. Onal, Daniela Rus, and Robert Wood. 2013. Robot self-assembly by folding: A printed inchworm robot. *Proceedings - IEEE International Conference on Robotics and Automation*, 277–282. https://doi.org/10.1109/ICRA.2013.6630588

Peter J Green. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* 82, 4 (1995), 711–732.

Ruslan Guseinov, Connor McMahan, Jesús Pérez, Chiara Daraio, and Bernd Bickel. 2020. Programming temporal morphing of self-actuated shells. *Nature Communications* 11 (01 2020), 237. https://doi.org/10.1038/s41467-019-14015-2

Jeffrey Herrmann and D.R. Delalio. 2001. Algorithms for sheet metal nesting. *Robotics and Automation, IEEE Transactions on* 17 (05 2001), 183 – 190. https://doi.org/10.1109/70.928563

Kristian Hildebrand, Bernd Bickel, and Marc Alexa. 2012. crdbrd: Shape Fabrication by Sliding Planar Slices. *Computer Graphics Forum* 31 (05 2012), 583–592. https://doi.org/10.1111/j.1467-8659.2012.03037.x

Carl-Johan Jonsson, Roland Stolt, and Fredrik Elgh. 2020. Stamping Tools for Sheet Metal Forming: Current State and Future Research Directions. https://doi.org/10.3233/ATDE200087

Jikai Liu, Andrew T Gaynor, Shikui Chen, Zhan Kang, Krishnan Suresh, Akihiro Takezawa, Lei Li, Junji Kato, Jinyuan Tang, Charlie CL Wang, et al. 2018. Current and future trends in topology optimization for additive manufacturing. *Structural and Multidisciplinary Optimization* 57, 6 (2018), 2457–2483.

N. Morris, A. Butscher, and Francesco iorio. 2020. A subtractive manufacturing constraint for level set topology optimization. *Structural and Multidisciplinary Optimization* 61 (2020), 1573–1588.

Vishal Naranje. 2010. AI applications to metal stamping die design– A Review.

Jay Patel and Matthew Campbell. 2008. An Approach to Automate Concept Generation of Sheet Metal Parts Based on Manufacturing Operations. *Proceedings of the ASME Design Engineering Technical Conference* 1 (01 2008). https://doi.org/10.1115/DETC2008-49648

Jay Patel and Matthew I. Campbell. 2010. An Approach to Automate and Optimize Concept Generation of Sheet Metal Parts by Topological and Parametric Decoupling. *Journal of Mechanical Design* 132, 5 (2010), 051001. https://doi.org/10.1115/1.4001409

Grand View Research. 2019. Sheet Metal Market Size, Share & Trends Analysis Report By Material (Steel, Aluminum), By End-Use (Automotive & Transportation, Building & Construction), By Region, And Segment Forecasts, 2019 - 2025. *Report ID* (2019), 110–1. https://www.grandviewresearch.com/industry-analysis/sheet-metal-market/segmentation

Nora Ripperda and Claus Brenner. 2009. Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments.

Daniela Rus and Michael Tolley. 2018. Design, fabrication and control of origami robots. *Nature Reviews Materials* 3 (05 2018), 1. https://doi.org/10.1038/s41578-018-0009-8

Sachin Salunkhe, Soham Teraiya, Hussein Mohamed, and Shailendra Kumar. 2019. *Smart System for Feature Recognition of Sheet Metal Parts: A Review: Volume 2*. 535–549. https://doi.org/10.1007/978-981-13-2718-6_52

Larry Sass. 2006. A Wood Frame Grammar: A Generative System for Digital Fabrication. *International Journal of Architectural Computing* 4, 1 (2006), 51–67. https://doi.org/10.1260/147807706777008920 arXiv:https://doi.org/10.1260/147807706777008920

H.-J. Schek. 1974. The force density method for form finding and computation of general networks. *Computer Methods in Applied Mechanics and Engineering* 3, 1 (1974), 115–134. https://doi.org/10.1016/0045-7825(74)90045-0

Joseph Schlecht, Kobus Barnard, Ekaterina Spriggs, and Barry Pryor. 2007. Inferring Grammar-based Structure Models from 3D Microscopy Data. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 1–8. https://doi.org/10.1109/CVPR.2007.383031

Adriana Schulz, Cynthia Sung, Andrew Spielberg, Wei Zhao, Robin Cheng, Eitan Grinspun, Daniela Rus, and Wojciech Matusik. 2017. Interactive robogami: An end-to-end system for design of robots with ground locomotion. *The International Journal of Robotics Research* 36, 10 (2017), 1131–1147. https://doi.org/10.1177/0278364917723465 arXiv:https://doi.org/10.1177/0278364917723465

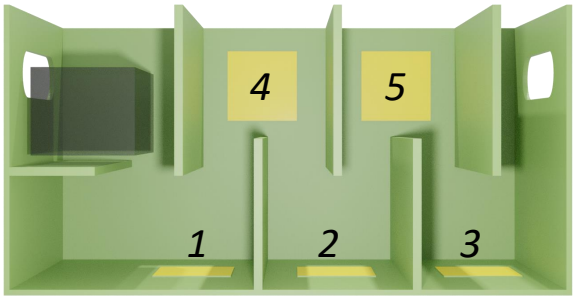
Aditya Soman and Matthew Campbell. 2002. A Grammar-Based Approach to Sheet Metal Design. *Proceedings of the ASME Design Engineering Technical Conference* 2 (01 2002). https://doi.org/10.1115/DETC2002/DAC-34087

Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. 2011. Metropolis Procedural Modeling. *ACM Trans. Graph.* 30, 2, Article 11 (April 2011), 14 pages. https://doi.org/10.1145/1944846.1944851

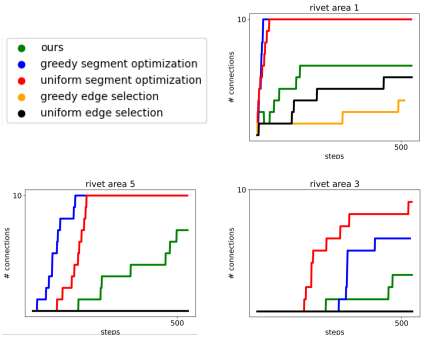
Chenming Wu, Haisen Zhao, Chandrakana Nandi, Jeffrey I. Lipton, Zachary Tatlock, and Adriana Schulz. 2019. Carpentry Compiler. *ACM Trans. Graph.* 38, 6, Article 195 (Nov. 2019), 14 pages. https://doi.org/10.1145/3355089.3356518

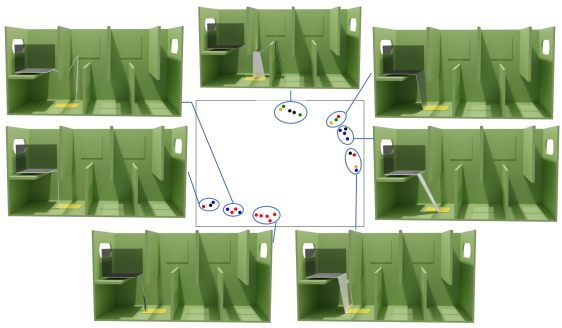
Yuan Yuan, Hua Xu, and Bo Wang. 2014. An Improved NSGA-III Procedure for Evolutionary Many-Objective Optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (Vancouver, BC, Canada) *(GECCO '14)*. Association for Computing Machinery, New York, NY, USA, 661–668. https://doi.org/10.1145/2576768.2598342

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

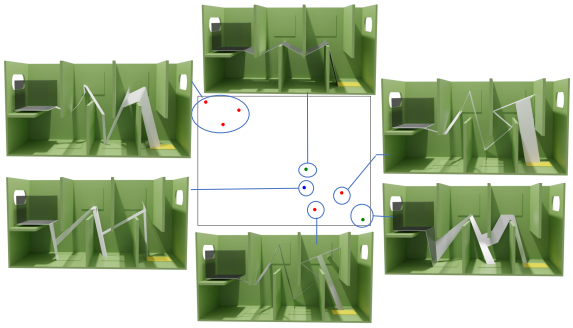


(a) Maze Environment



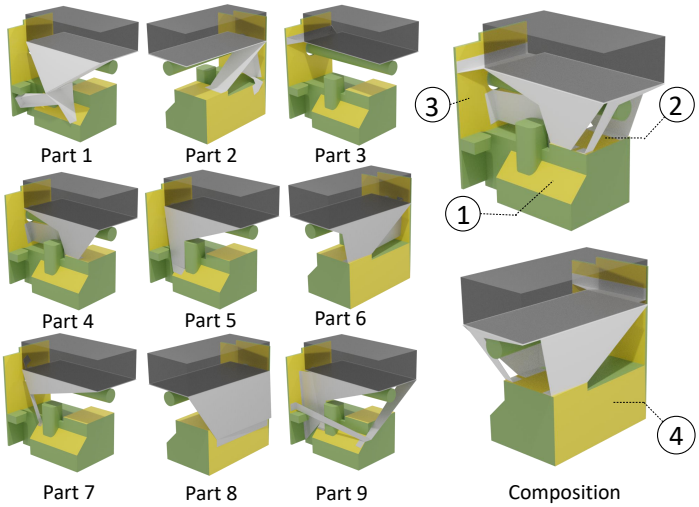


(b) t-sne analysis for rivet area 1



(c) t-sne analysis for rivet area 3

Fig. 13. Exploration results after 500 iterations for different policies. Our edge and segment selection policies balance the number of discovered segments with segment variety. Note that both greedy and uniform edge selection policies do not reach rivet area 3 at all. The greedy and uniform segment selection policies find more segments, but most are identical to existing segments or of low quality, resulting in an unnecessarily bloated Segment library.



Part/Rivet Area	1	2	3	4	MOS	Compliance
1					-0.28	243
2					-0.74	549
3					-0.7	551
4					-0.36	126
5					-0.69	604
6					0.21	103
7					-0.92	8408
8					1.07	149
9					0.5	19.4
composition					**15.3**	**0.841**

Fig. 14. Segment Composition ablation. The compliance target given was 0.85. Each chain creates has a valid part locally, but none cover all the possible rivet areas. The composition is shown on their right. The table on the right shows that none of the parts achieve the required compliance target. The composition step allows us to build a part from each segments' chains, allowing us to build a part which connects to all available rivet areas, allowing for better compliance.