Towards Zero-Waste Furniture Design

Bongjin Koo[1,⋆] Jean Hergel[2,⋆] Sylvain Lefebvre[2] Niloy J. Mitra[1]

[1]University College London [2]INRIA Nancy

Abstract—In traditional design, shapes are first conceived, and then fabricated. While this decoupling simplifies the design process, it can result in unwanted material wastage, especially where off-cut pieces are hard to reuse. In absence of explicit feedback on material usage, the designer remains helpless to effectively adapt the design – even when design variabilities exist. We investigate *waste minimizing furniture design* wherein based on the current design, the user is presented with design variations that result in less wastage of materials. Technically, we dynamically analyze material space layout to determine *which* parts to change and *how*, while maintaining original design intent specified in the form of design constraints. We evaluate the approach on various design scenarios, and demonstrate effective material usage that is difficult, if not impossible, to achieve without computational support.

1 Introduction

Furniture design is an exercise in form-finding wherein the designer arrives at a final form by balancing aesthetics, object function, and cost. Typically, design variations are manually explored by a mixture of guesswork, prior experience, and domain knowledge. Without appropriate computational support, such an exploration is often tedious, time consuming, and can result in wasteful choices.

In furniture manufacturing, both for mass production and for customized designs, material wastage plays a deterrent role. This not only leads to increased production cost (typically 5-20% wastage due to off-cuts), but also hampers ongoing efforts towards green manufacturing. Thus, there has been a growing interest in *zero-waste furniture* in order to reduce material wastage. A notable example being Maynard's 'Zero-waste Table.' Computational tools for such waste-reducing furniture designing are largely lacking.

Typically, material considerations are appraised only *after* a shape has been designed. While this simplifies designing, it leads to unnecessary wastage: at design time, the user can at best guess to account for how the shape will be physically realized, and can easily fail to effectively adjust the design to improve material utilization.

In the recent years, algorithms have been developed to economically 3D print given designs. For example, approaches have been proposed to cleverly breakup a given shape into parts that better pack together in print volumes [1], [2], [3], adaptively hollow shape interiors to save print materials [4], [5], [6], [7], explore parameter space variations for manufacturable forms [8], or design connector geometry to remove the need for any secondary connectors [9]. Improving material utilization by explicitly enabling interactive design changes has been less studied.

In this work, we introduce the problem of *waste-minimizing furniture design*, and investigate it in the context of flatpack furniture design (cf., [10]) using laser cut wooden parts. Specifically, we study the interplay between furniture design exploration and cost-effective material usage. By directly coupling the two, we empower the users to make more informed design decisions.

For example, in Figure 1, the user starts with an initial concept indicating *design constraints* (e.g., symmetry, desired height, etc.). Our system analyzes *material wastage* by computing a dynamic 2D layout of the parts and proposes design modifications to reduce material wastage without violating specified design constraints (i.e., design intent). Note that such adaptations often require synchronous adjustment of multiple parts affected by both design and material layout considerations, which are difficult to mentally imagine. The

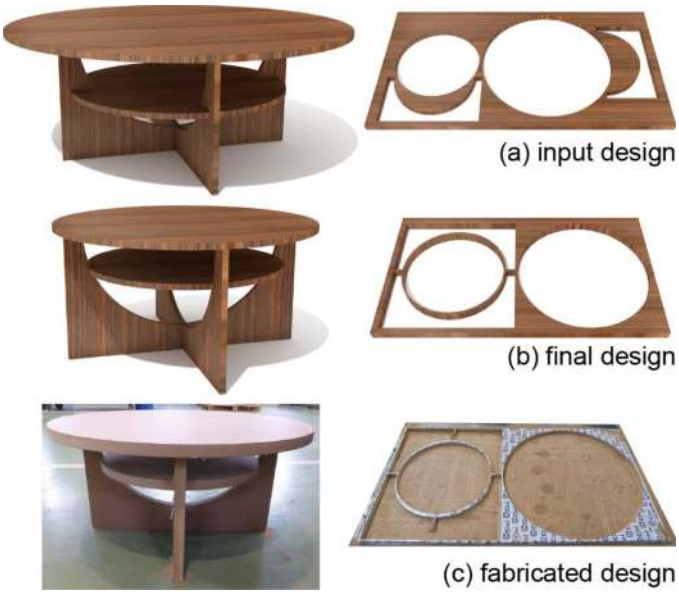


Fig. 1. We introduce waste-minimizing furniture design to dynamically analyze an input design (a) based on its 2D material wastage (shown in the right column) and design specifications to assist the user through (b) design suggestions to reduce material wastage. The final user design can directly be exported for laser cutting and be assembled (c). In this case, wastage was reduced from 22% to 11%.

⋆*Joint first authors.*

user can select any of the suggestions, either in its entirety or in part. She can further update the set of design constraints by locking parts of the current design, and continue the process. Thus, the user scopes out a design space via constraints, and our algorithm refines the design to reduce material wastage while restricting changes to the implicitly specified design space.

Technically, we achieve the above by using the current material layout to *dynamically* discover a set of relevant layout constraints. The algorithm has a discrete aspect involving *which* part to change based on the current 2D layout, and a continuous aspect involving *how* to adapt the part attributes based on the current material space layout without violating user-specified design constraints. Unfortunately, even for a fixed design, exploring the space of all possible packing is a combinatorial NP-hard problem. Instead, we locally analyze a set of candidate packings to determine which parts to modify and how to change them to reduce material wastage. We demonstrate that dynamic analysis of a set of current packings allows efficient and effective coupling of the 2D layouts and the constrained 3D designs. The user is then presented with different waste-reducing design variations to select from.

We evaluated the system to create a variety of simple and complex designs, and fabricated a selection of them. We also performed a user study with both designers and novices to evaluate the effectiveness of the system. The performance benefits were particularly obvious in case of complex designs involving different design constraints. In summary, we introduce the problem of material waste minimizing furniture design; and propose an algorithm that dynamically analyzes 2D material usage to suggest design modifications to improve material usage without violating user-specified constraints.

2 Related Work

Material considerations. Physical materials play an important role in 3D printing. Various approaches have been developed to economically and efficiently produce a designed object. For example, adaptively hollowing out interiors and adding struts to create durable yet cost-effective 3D printouts [4], cleverly hollowing the shape interiors in conjunction with shape deformation to ensure stability of the final shape [5], or perform FEM analysis to decide wall thickness and parameters to ensure model endurance under known or unknown forces [11], [12]. Techniques for designing scaffolds, both interior [6] and exterior [7], have been developed. Hu et al. [13] propose to optimize the shape of a 3D model to reduce support structures used during 3D printing. Alternatively, methods have been developed to decompose and pack 3D models for reducing assembly cost, support material, printing time or making big objects printable on small 3D printers [1]. Notably, Dapper [3] employs a decompose-and-pack approach for minimum assembly cost, support material and build time when using 3D printers. It breaks 3D objects into pyramidal primitives, then finds good packing configurations to achieve the goal.

In the context of laser cut fabrication, Hildebrand et al. [14] and Schwartzburg and Pauly [15] explore how to rationalize a given design for fabrication out of planar sheets. More broadly, Wang et al. [16] investigate planar material based fabrication for cloth industry, toy industry, etc. Further, material wastage has been investigated by testing various packing strategies from computational geometry community (cf., [17]) to efficiently layout the parts in the material space. More recently, Saakes et al. [18] proposed an interactive system to allow the user to interactively layout parts for more personalized usage. Such methods, however, do not explicitly *modify* the original designs in order to improve material usage.

Fabrication-aware design. Recently, the growing popularity of personalized fabrication has motivated researchers to develop algorithms to adapt existing shapes to make them better suited for physical construction. Examples include abstracting shapes as a collection of slices to be laser cut [14], [15], [19], [20], as foldable popups [21], developing toolkit to allow user to draft directly using a handheld laser pointer to control high-powered laser cutters [22], computationally designing gear trains to support part movement for converting animated characters to working physical automata [23], introducing necessary joint geometry to create non-assembly articulated objects [24], [25], supporting an example-driven fabrication paradigm [26], ergonomics-driven design [27], or designing with standardized components in a discrete optimization setting [28]. To simplify fabrication, Fu et al. [9] suggest a method to generate a globally-interlocking furniture assembly that enables easy disassembly/reassembly of furniture, *without* using glue, screws, etc. Such methods, however, are chiefly used to adapt existing shapes *after* they have been designed, rather than to guide the user to refine the designs.

Guided design. In the context of exploratory design, Xu et al. [29] proposed a fit-and-diverse framework to allow users to interactively guide model synthesis and exploration, while Talton et al. [30] exposed a parameterized shape space for model creation. These efforts, however, focus on aspects of digital content creation without fabrication and material considerations. Recently, Shugrina et al. [8] developed a system that allows novices to easily customize parametric models while maintaining 3D-printability of the models. In a work closely related to our motivation, Umetani et al. [31] use stability and durability of materials to propose design modifications, thus computationally guiding the users. With a similar motivation, we investigate the impact of material usage in the context of guided design. We are unaware of prior attempts investigating how material wastage can be dynamically analyzed to refine the designs.

Constraint-based modeling. In the CAD community, constrained-based modeling has long been demonstrated as a powerful parametric way to design shapes and interact with them. In the case of existing models, an inverse analyze-and-edit paradigm was proposed to first discover the constraints present in shapes, and then allow interactive editing [32], [33], [34]. Such approaches differ on how model parts are abstracted (e.g., feature curves, model parts, or abstracted segments as primitives) and how the inter-part constraints are conformed to. However, these methods have primarily focused on designing shapes for the virtual world where material and fabrication constraints are irrelevant, and hence ignored.

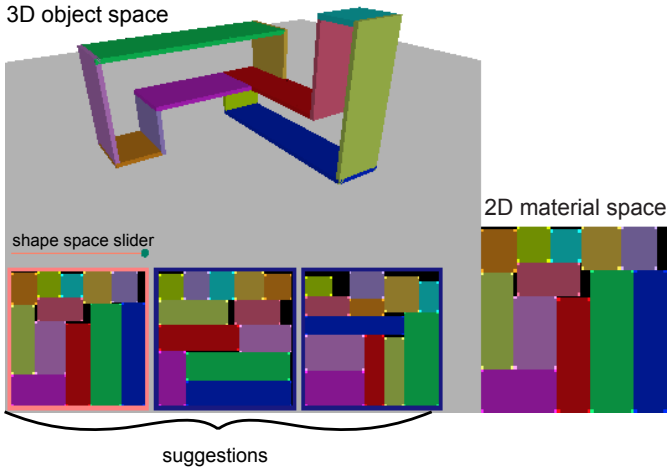


Fig. 2. System Interface. The user can freely design in the *3D object space* pane by adding/deleting planks, or directly editing plank dimensions. The user design is dynamically analyzed by laying it out in the *2D material space* and design adaptations are proposed to improve material usage (i.e., layout-based suggestions), or design effectiveness (i.e., outer/inner volume). The user can select a suggestion, and use the shape space slider to navigate along the proposed edit path. The user can accept a suggestion (or part of) and continue to edit. Once satisfied, the user can directly ask for cutting pattern to be generated along with necessary finger/cross joint specifications.

3 Design Workflow

Our system (see Figure 2) exposes design variations that minimize material wastage without violating original design intent. In this section, we present the proposed system as experienced by the user, and describe the main algorithmic details in the subsequent sections.

The user starts by choosing the desired material (i.e., thickness of wooden planks) and the number and dimensions of the master board(s). We support rectangular master boards — in practice, these can represent new boards or left over rectangular spaces in already used boards. The user starts by loading an initial part-based 3D object design, either created in a modeling system or as a parameteric model. The parts can be rectangular or have curves boundaries. The user also indicates a set of *design constraints*. In our implementation, we support: equal length (e.g., $l_i = l_j$), sum of lengths (e.g., $l_i + l_j + \cdots = l_k + \ldots$), fixed length (e.g., $l_i = c$), equal position, symmetric parts, ground touching, and coplanarity among indicated planks. The user can additionally specify that the object should fit an indicated volume (e.g., in between two walls) and the internal space in the form of inner volume indicating minimal shelf dimensions (see Figure 3).

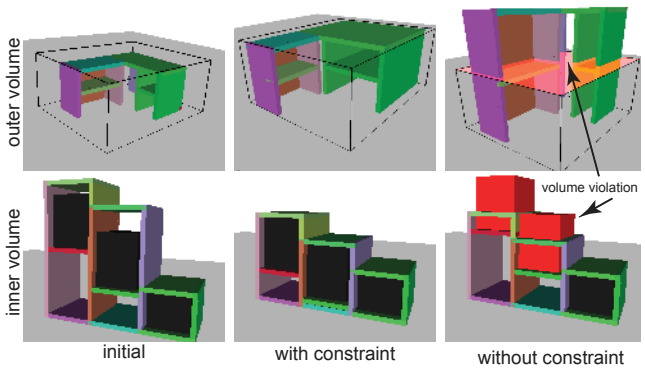


Fig. 3. We show effects of designing with (middle column) or without (right column) the respective constraints activated.

Based on the initial user-specified design, our system automatically adjusts the planks based on the static design constraints and dynamically generated constraints arising from material wastage. If after repacking the updated planks, all the constraints continue to be satisfied, the system passes the control back to the user. At any time the user can ask the system to analyze the current design and to propose multiple suggestions to reduce material wastage. This takes into account the design constraints, the material-space constraints, and design effectiveness (if applicable). We measure material wastage based on the unused fraction of the master board(s).

The top 3 generated suggestions are presented to the user as thumbnails. If the user mouse-overs any thumbnail, the system animates the proposed design modifications. The user can preview the object- and material-space views, and select her preferred design suggestion. Note that each thumbnail effectively represents a design exploration path pursued by the algorithm. We provide a slider to move along this path, which is particularly useful for making incremental updates to the design (see Figure 4 and Section 4).

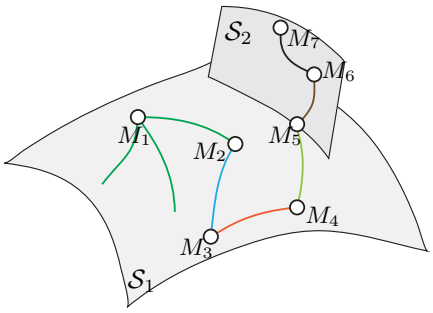


Fig. 4. Our algorithm discovers design variations in shape space. The user starts from a design M_1 along with indicated design constraints, and the algorithm seeks for wastage minimizing variations by interleaving between topologically different material layouts (indicated by changes in curved paths) or continuous changes to the layouts (indicated by same colored curves). For example, paths (M_i, M_j) denote continuous design changes, while points M_i denotes designs where new layouts are explored (i.e., branch points). The user can switch to another shape space by picking an updated set of design constraints (shape M_5 here). Note that by construction M_5 belongs to both shape spaces $\mathcal{S}_1$ and $\mathcal{S}_2$. See Algorithm 1.

The user either selects a suggested design variation, or picks part configurations from a suggested shape as additional design constraints (e.g., user can lock the proposed sizes of certain planks). Thus, effectively the user appends or updates the current set of specified design. Note that the new constraints are trivially satisfied by the current design, which is critical for subsequent design space exploration (e.g., M_5 is in both shape spaces $\mathcal{S}_1$ and $\mathcal{S}_2$).

Once satisfied with the design, the user can request for the cutting patterns. She can investigate the design, the material space usage and the cutting patterns, and send the patterns directly to the laser cutter (see supplementary video).

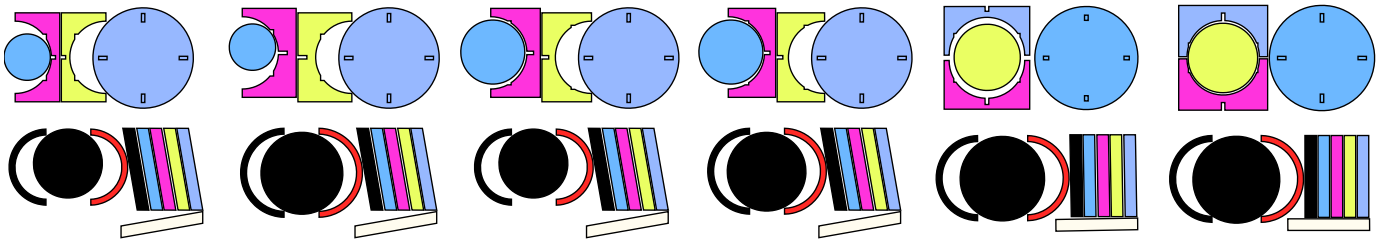


Fig. 5. Evolution of shape variation across a run of our algorithm on the coffee-table (top) and low-chair (bottom) models, improving upon a naive initial packing. In both cases our system discovers, through design changes, a configuration that can be packed more efficiently.

4 Overview

Our goal is to analyze aspects arising from material considerations, and investigate how design changes affect such considerations. Specifically, we ask how to adapt a furniture design so that it makes better utilization of material in the resultant design layout. Note that this is the inverse of the design rationalization problem, i.e., instead of taking a design as fixed and best fabricating it, we *adapt* the design so that the resultant rationalization makes better utilization of available material. First, we introduce some notations.

4.1 Parameterized designs

The design is considered as a function $D(\mathbf{X})$ that produces the geometry of a fixed number of parts, given a configuration vector $\mathbf{X}$. The parts can be assembled into a final furniture design.

We make no assumption as to how D is implemented – we demonstrate in Section 6 applications using both constrained based furniture design and parametric designs modeled by CSG. We however expect a continuous behavior from $D(\mathbf{X})$, i.e., small changes in $\mathbf{X}$ result in small changes in the part shapes. Parametric modelers generally offer such continuity to smoothly navigate the space shape.

During wastage optimization, our algorithm will change the value of $\mathbf{X}$ so as to explore whether changes in part shapes reduce wastage. Since we focus on laser cut furniture construction, we assume the parts to have the same thickness τ. The parts are thus represented as planar polygonal contours extruded orthogonally.

The geometry of a part p_i lies within an axis aligned bounding box, which we represent by a six dimensional vector encoding the box center $\mathbf{p}_i$ and the lengths of its three sides l_i^x, l_i^y, τ with the Z axis being aligned with part thickness by convention.

4.2 Material space

Since we focus on laser cut furniture, any 3D design given by a configuration vector $\mathbf{X}$ is realized as a layout (i.e., cutting plan) in the material space. Material space is characterized by the largest *master board* that the machine can possibly cut, a rectangle of size $W \times H$. In this space, each part i is associated with a position (u_i, v_i) and an orientation $o_i \in \{0, \pi/2, \pi, 3\pi/2\}$.

We use w_i, h_i as extent of a part bounding box in the material space along the x- and y-axis, respectively. The part box lengths in material space are given by the two plank dimensions other than thickness. For a plank i, of orientation o_i, we get one of the two cases:

$$\begin{array}{lcll} o_i = 0, o_i = \pi & \Rightarrow & w_i = l_i^x & h_i = l_i^y \\ o_i = \pi/2, o_i = 3\pi/2 & \Rightarrow & w_i = l_i^y & h_i = l_i^x \end{array}$$

The material space positions and orientations are variables in the layout optimization algorithm, alongside the design parameters $\mathbf{X}$ (see Section 5).

When wastage is not a concern and a design easily fits within material space, the variables (u_i, v_i, o_i) are independent of the design, i.e., they simply adapt to changes in part sizes. However, as we seek to minimize the wastage in the material space, the layout variables become tightly coupled with the design parameters. Our layout optimizer therefore jointly optimizes for material space variables and design parameters to minimize wastage (see Section 5). We next discuss what makes a desirable layout from the point of view of furniture fabrication.

4.3 Properties of a good design layout

Our goal is to achieve a full utilization of rectangular spaces, so that the user can use boards of exactly the right size and minimize wastage. The machine dimensions determine the maximum extent of a single board.

We measure wastage as the fraction of the space *not* utilized by the design in its material space bounding rectangle. Ideally, we want to achieve full utilization, i.e., zero wastage. An ideal packing is one that tightly packs all the parts to perfectly fill up one or more rectangular master boards (like a puzzle). Our system helps the user achieve this by automatically exploring changes improving material space usage (see Figure 6).

Fig. 6. Examples of stages of layout refinement, from bad to mediocre to good. A good layout is characterized by less area of material wasted (shown in green).

5 Design Layout Optimization

The wastage of a layout depends essentially on two factors. First, the quality of the packing that can be achieved given a fixed set of design parts. Second is the set of parts itself, which can be changed through the design parameters $\mathbf{X}$. Our design optimization simultaneously improves both as illustrated in Figure 5.

Algorithm 1: MinWastage

Input: Design function D, starting design parameters $\mathbf{X}_s$
Output: Set of best layouts found $\mathcal{L}$

1 $O_s \leftarrow$ identity ordering ; // 1,2,3,...
2 $\mathcal{X} \leftarrow \{(\mathbf{X}_s, O_s)\}$;
3 **for** G *iterations* **do**
4 **foreach** $(\mathbf{X}, O) \in \mathcal{X}$ **do**
5 $\mathcal{O} \leftarrow$ ExploreOrderings($\mathbf{X}, O$);
6 **foreach** $O \in \mathcal{O}$ **do**
7 $\mathcal{X} \leftarrow \mathcal{X} \cup \{$(ImproveDesign($\mathbf{X}, O$),$O$)$\}$;
8 $\mathcal{X} \leftarrow$ KeepBests(K,$\mathcal{X}$);
9 $\mathcal{L} \leftarrow \emptyset$;
10 **foreach** $(\mathbf{X}, O) \in \mathcal{X}$ **do**
11 $\mathcal{L} \leftarrow \mathcal{L} \cup$ Docking($D(\mathbf{X})$,O);
12 **return** ($\mathcal{L}$);

We pack the parts using a deterministic docking algorithm that always produces the same result for a same ordering of the design parts. Therefore, a first optimization variable is the order in which the parts are sent to the docking algorithm. The second optimization variable is the vector of design parameters $\mathbf{X}$. These two variables have different natures: finding an ordering is a combinatorial problem, while the design parameters can be continuously explored.

We proceed in two main steps, first determining a set of good orderings that then serve as starting points for continuously evolving the design, reducing wastage. The overall approach is described in Algorithm 1. The subroutine ImproveDesign is described in Section 5.1 while ExploreOrderings is described in Section 5.2. The process restarts for a number of iterations (we use $G = 3$) to jump out of local minima reached by the continuous design exploration. This results in the shape space exploration illustrated in Figure 4. The process returns the K best found layouts and designs and presents them to the user in thumbnails. She can then select her favorite design, and if desired update the constraints and restart the exploration from this point by simply calling MinWastage again.

Bitmaps. During optimization we regularly call the parameterized design function $D(\mathbf{X})$ to obtain a new set of parts after changing parameters. The layout optimization represents parts internally as bitmaps: each part contour is rasterized at a resolution τ, typically 0.5 mm per pixel. This enables fast manipulation of the parts within the layout. Each part thus becomes a bitmap having either 1 (inside) or 0 (outside) in each pixel. The size of the bitmap matches the part extents in material space w_i and h_i. Every time the design is refreshed a new set of bitmaps is computed for the parts. The master board is similarly discretized into a regular grid of resolution τ. Note that this step of using bitmap-based packing is reminiscent of packing for efficient texture atlas generation [35].

5.1 Design optimization for wastage minimization

The design optimization improves the design parameters $\mathbf{X}$ to minimize wastage in the layout, keeping the docking ordering fixed (see subroutine ImproveDesign in Algorithm 1). The pseudo-code for this step is given in Algorithm 2. Our objective is to suggest design changes that reduce wastage, progressively improving the initial layout. The algorithm performs a guided local search by changing the parts through the design parameters to reduce wastage.

Algorithm 2: ImproveDesign

Input: Starting design parameters $\mathbf{X}$ and ordering O
Output: Modified design parameters $\mathbf{X}_b$ with reduced wastage

1 $L \leftarrow$ Docking(D($\mathbf{X}$),O);
2 $\mathbf{X}_b \leftarrow \mathbf{X}, L_b \leftarrow L$;
3 $\mathbf{X}_c \leftarrow \mathbf{X}, L_c \leftarrow L$;
4 **for** N *iterations* **do**
5 $\mathbf{X}_b, L_b \leftarrow$GrowParts($\mathbf{X}_b, L_b$,$\mathbf{X}_c, L_c, O$);
6 $\mathbf{X}_c \leftarrow$ShrinkParts($\mathbf{X}_b, L_b$);
7 $L_c \leftarrow$ Slide(L_b,$D(\mathbf{X}_c)$);
  // Check for improvement over current.
8 **if** $W(L_c) < W(L_b)$ **then**
9 $\mathbf{X}_b = \mathbf{X}_c, L_b = L_c$;
10 **return** ($\mathbf{X}_b$);

Prior to considering which parts to modify, we have to answer two questions: First, how to drive the design parameters $\mathbf{X}$ to change only a given part (Section 5.1.1). This is achieved by relying on the gradients of the part size with respect to $\mathbf{X}$. Second, we have to decide on how to evolve the layout when parts are changed (Section 5.1.2). We rely on a sliding algorithm that avoids jumps in the layout configuration, thus producing only small changes in the wastage function when small changes are applied to the part sizes.

Overall strategy. Our approach changes the size of parts iteratively with two different steps in each iteration: *grow* (line 5) and *shrink* (line 6). These steps progressively modify the design and keep track of the design of smallest wastage encountered so far.

The grow step (Section 5.1.3) attempts to enlarge the parts so as to reduce wastage. Each part is considered and its size is increased for as long as the growth further reduces wastage. When no further improvement can be obtained, we create further opportunities by shrinking a set of parts (Section 5.1.4). However, randomly shrinking parts would be inefficient, as most parts would grow back immediately to their original sizes. Other parts are tightly coupled to many others in the design D, and shrinking these would impact the entire design. Therefore, we analyze the layout to determine which parts have a higher probability to result in wastage reduction.

5.1.1 *Changing part sizes*

During design space exploration the algorithm attempts to vary the part bounding box sizes w_i and h_i individually. These dimensions vary as a function of design parameters $\mathbf{X}$. Note in particular that with parametric models the change in bounding box size of a part may result from arbitrary operations such as shearing or CSG (see Figure 9b,c). In the remainder we use $\mathbf{s}(\mathbf{X})$ to designate the vector of all part sizes assembled such that $s_{2i} = w_i$ and $s_{2i+1} = h_i$.

Let us denote λ the change of size desired on s_i. Our objective is to compute a design change Δ such that $s_i(\mathbf{X}+\Delta) = s_i(\mathbf{X}) + \lambda$. We denote the vector of changes as $\Lambda = \mathbf{s}(\mathbf{X}+\Delta) - \mathbf{s}(\mathbf{X})$. In this process only the size s_i should change with others remain unchanged whenever possible, that is $\Lambda_{s_j, j\neq i} = 0$ and $\Lambda_{s_i} = \lambda$.

However, parts are not independent in the design and therefore there is no trivial link between $\mathbf{X}$ and $s_i(\mathbf{X})$. We therefore analyze the relationship through the gradients $\frac{\partial s_i(\mathbf{X})}{\partial x_j}$. These are computed by local finite differencing (depending on the design analytical expressions may be available). Each non-null gradient indicates that parameter x_j influences s_i. Multiple parameters may influence s_i and parameters typically also influence other variables: there exists $k \neq i$ such that $\frac{\partial s_k(\mathbf{X})}{\partial x_j} \neq 0$.

To compute Δ we formulate the following problem. Let us consider the components of $\Delta = (\delta_0, ..., \delta_{|\mathbf{X}|-1})$. The change in part sizes due to Δ can be approximated in the first order through the gradients as $\mathbf{\Lambda} = \sum_i \delta_i \cdot \frac{\partial \mathbf{s}(\mathbf{X})}{\partial x_i}$. We solve for Δ such that $\Lambda_{s_i} = \lambda$ and $\Lambda_{s_j, j\neq i} = 0$.

If there are less parameters than part sizes, the problem is over-constrained and solved in the least-square sense, minimizing $||\mathbf{\Lambda} - (0, ..., \lambda, ..., 0)||^2$. If there are more parameters than part sizes, the problem is under-constrained and solved in the least-norm sense, minimizing $||\Delta||$. We rely on a QR decomposition of the system matrix to solve for both cases, accounting for possible rank deficiencies due to overlapping parameters in $\mathbf{X}$.

We implement this process as a subroutine CHANGEPARTSIZE($\mathbf{X}$,s_i,λ), with $\mathbf{X}$ the current design parameters, s_i the part size to change and λ the change to apply. It returns the new design parameters $\mathbf{X}+\Delta$. A second subroutine CHANGEPARTSIZES($\mathbf{X}$,Λ) allows to change the size of multiple parts at once. These routines automatically take into account constraints by returning only valid designs: if a constraint is violated during the part size change, the change is discarded ($\Delta = 0$).

5.1.2 Updating layouts by sliding

As the shapes and sizes of the parts change the layout has to be updated. One option would be to restart the docking process after each change. However, for a small change the docking process can produce large discontinuities in the wastage function. This makes a local search difficult. Instead, we propose a sliding operation that attempts to continuously update the position of the parts after each change. Note that performing such an update while optimizing for a given objective (i.e., wastage) is a very challenging combinatorial problem, as each part can move in four directions (left, right, top, or bottom) and multiple cascading overlaps have to be resolved. We propose a heuristic approach that works well for small changes in the part shapes.

The algorithm is based on the following principle. After changing the part shapes, we reintroduce them in an empty layout in order of docking. However, each time a part is reintroduced it may now have empty space to its left/bottom or it may overlap with previously placed parts. Both cases can be resolved by a single horizontal or vertical move. However, a single move is generally not desirable as empty space may remain along the other direction. We therefore perform a limited sequence of horizontal/vertical moves. At each iteration we select between vertical or horizontal by favoring moves that result in the smallest layout bounding box. In case of a tie, we favor moves to the left/bottom versus displacements to the top/right (see Figure 7).

Algorithm 3: SLIDE

Input: current layout $C = (u_0, v_0, ...)$ and set of changed parts $parts$
Output: updated layout L

1 $L \leftarrow \emptyset$
2 **foreach** *part* $p_i \in parts$ *in docking order* **do**
3 **for** N *iterations* **do**
4 $\Delta_x \leftarrow -smallestLeftFreeInterval(L, p_i)$;
5 **if** $\Delta_x = \emptyset$ **then**
6 $\Delta_x \leftarrow smallestRightDecollision(L, p_i)$;
7 $pos_x \leftarrow (u_i + \Delta_x, v_i)$;
8 $\Delta_y \leftarrow -smallestBottomFreeInterval(L, p_i)$;
9 **if** $\Delta_y = \emptyset$ **then**
10 $\Delta_y \leftarrow smallestTopDecollision(L, p_i)$;
11 $pos_y \leftarrow (u_i, v_i + \Delta_y)$;
12 **if** $pos_x = \emptyset$ *and* $pos_y = \emptyset$ **then**
 ; // cannot fit masterboard
13 return $\emptyset$; // $W(\emptyset) = 1$
14 **if** $pos_x = pos$ *and* $pos_y = pos$ **then**
15 break;
16 **if** $A(box(L \lhd_{pos_x} p_i) < A(box(L \lhd_{pos_y} p_i))$ **then**
17 $(u_i, v_i) \leftarrow pos_x$
18 **else if** $A(box(L \lhd_{pos_x} p_i) > A(box(L \lhd_{pos_y} p_i)$ **then**
19 $(u_i, v_i) \leftarrow pos_y$
20 **else**
21 **if** $\Delta_x < \Delta_y$ *and* $|\Delta_x| > 0$ **then**
22 $(u_i, v_i) \leftarrow pos_x$
23 **else**
24 $(u_i, v_i) \leftarrow pos_y$
25 $L \leftarrow L \lhd_{(u_i, v_i)} p_i$
26 **return** (L);

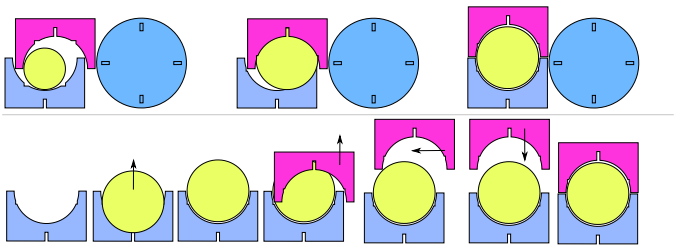


Fig. 7. Sliding a layout after a change of part sizes. **Top:** From left to right, initial layout, same after change revealing overlaps, layout after sliding. **Bottom:** Moves performed on the three first parts during sliding.

The pseudo-code is given in Algorithm 3. In the algorithm we denote by L the layout and denote by $L \lhd_{pos} p_i$ the layout obtained when adding part p_i at position pos in the master board grid of L. Let $A(.)$ measure the area, and $box(L)$ denote the bounding rectangle of the layout. The algorithm iterates over all parts in docking order (line 2). It then performs a fixed number of sliding operations on each part (line 3) – we use $N = 4$ in our implementation. Lines 4-7 compute a horizontal move, favoring moves to the left that collapse newly created empty spaces. Lines 8-11 similarly compute a vertical move. Lines 16-24 decide whether to select a horizontal (pos_x) or vertical (pos_y) move.

The process may fail if parts can no longer fit in the masterboard. This can happen either because there is not enough remaining area, or because sliding cascades in large moves that prevent further insertion of parts. In such cases we return an empty layout, which by convention has a wastage of 1 (worst possible), line 13.

Algorithm 4: GROWPARTS

Input: Best design parameters $\mathbf{X}_b$ and layout L_b so far, current design parameters $\mathbf{X}_c$ and current layout L_c being explored, ordering O.
Output: New best design and packing.

```
1  improvement ← true;
2  while improvement do
3      improvement ← false;
4      foreach part size s_i in random order do
5          W_e ← 1 ; // max wastage
6          X_e ← X_c, L_e ← L_c ;
           // Grow a first time and then continue as
              long as it improves.
7          while true do
8              X_e ← ChangePartSize(X_e,s_i,1) ; // +1 pix.
9              L_e ← Slide(L_e,D(X_e));
10             if W(L_e) > W_e then
11                 L_e ← Docking(D(X_e),O);
12             if W(L_e) < W_e then
13                 W_e = W(L_e);
14             else
15                 break;
           // Check for improvement over current.
16         if W_e < W(L_c) then
17             X_c = X_e, L_c = L_e;
18             improvement ← true;
   // Check for improvement over global best.
19 if W(L_c) < W(L_b) then
20     X_b = X_c, L_b = L_c;
21 return (X_b, L_b);
```

5.1.3 Grow step

The grow step is described in Algorithm 4. The algorithm iterates over all parts in random order (line 4) and progressively increases the size of a part in a loop (line 7). Note that the first iteration of the loop determines the starting wastage for growing this part (lines 5 and 12-13). The process continues until the growth results in an increased wastage (line 15).

After each change of parameters the design parts are recomputed (line 9, $D(\mathbf{X_e})$) and sliding is called to adapt the current layout to the change. If wastage decreases, we continue the process (line 13). If not, we first attempt to dock the parts again (line 11). This can help continue the growth in cases were sliding fails to resolve overlaps by continuous changes. If wastage still not decrease, we stop the growth of this part size (line 15).

5.1.4 Shrink step

The goal of the shrink step is to create further opportunities for design changes when no parts can further grow. The typical situation is that a subset of parts are forming *locking chains* between respectively the left/right and top/bottom borders. The parts belonging to such chains prevent any further growth. We therefore detect locking chains and select the parts to shrink among these. This often results in a change of aspect ratio of the masterboard, and new opportunities for other parts to grow.

Algorithm 5: SHRINKPARTS

Input: Best design parameters $\mathbf{X}_b$ and layout L_b so far.
Output: Shrunk design parameters.

```
1  X_s ← X;
2  S ← SelectPartSizesToShrink(L_b);
3  Λ ← (0, ..., 0);
4  foreach s_i ∈ S do
5      Λ_i ← −1 ; // -1 pixel
6  X_s ← ChangePartSizes(X_s,Λ) ; // -1 pixel
7  return (X_s);
```

The overall approach is described in Algorithm 5. It first determines which parts to shrink by calling SELECTPARTSTOSHRINK and then computes a change of parameters using the approach described in Section 5.1.1.

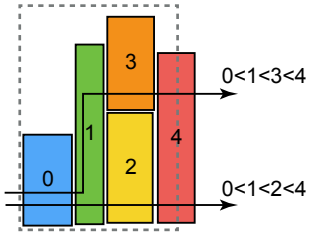
Algorithm 6: SELECTPARTSIZESTOSHRINK

Input: A layout L.
Output: Set of part sizes to shrink.

```
1  K ← ∅;
2  foreach axis a ∈ {X, Y} do
3      C ←GatherContactsAlongAxis(a) ;
4      K ← K ∪ FormContactChains(C) ;
5  S ← ∅;
6  while K ≠ ∅ do
7      s_i ← DrawPartSizeWithProbability(K);
8      S ← S ∪ {s_i};
9      K ← K \ KilledChains(K,s_i);
10 return S;
```

The core component is the SELECTPARTSIZESTOSHRINK subroutine (see Algorithm 6), which gathers all contacts between parts in the layout – this is done efficiently in the discretized layout grid. We first draw the part images into the grid and then check pairs of neighbors belonging to different parts. This produces the set of left/right and bottom/left contacts between part sizes (the involved part size is deduced from the part orientation and the considered axis). The contacts are oriented from right to left (respectively top to bottom). We similarly detect which parts touch the borders (see GATHERCONTACTSALONGAXIS).

Having obtained the contacts, we start from the left (respectively top) border and form locking chains as illustrated in the inset. Starting from the border, we produce the set of chains iteratively. Each chain c is a sequence $(left, s_{first}, ..., s_{last})$. At each iteration the chain spawns new chains for each contact pair (s_{last}, s_{next}) obtained by augmenting c as $(left, s_{first}, ..., s_{last}, s_{next})$. Potential cycles are easily detected as repetition of a same part in the chain and are ignored (see FORMCONTACTCHAINS subroutine).



Next, we randomly select part sizes to shrink until all locking chains are removed. The selection probability of each part is designed to avoid too large a jump in the

design space. To achieve this we consider two factors. First, we compute the number of occurrences of each part in the locking chains, $occ(p_i)$. A part with many occurrences is a good candidate as shrinking it will resolve multiple locking chains at once. Second, we seek to avoid shrinking part sizes that are tightly coupled with others in the design D. We compute the dependence of a part size by counting the number of non-zero entries in the $\mathbf{\Lambda}$ vector computed internally by CHANGEPARTSIZE($\mathbf{X_e}$,s_i,-1).

We select part sizes as follows. First, we select a number of occurrences o with probability $P(o) = \frac{\sum_{p_i, occ(pi)=o} occ(o)}{\sum_{p_i} occ(p_i)}$. Then, among the parts such that $occ(p_i) = o$ we select a part size s_i with probability $P(s_i|occ(s_i) = o) = 1 - \frac{dep(s_i)}{\sum_{p_j, occ(p_j)=o} dep(p_i)}$. This process is implemented by the DRAWPARTSIZEWITHPROBABILITY subroutine.

After each part size selection we update the set of locking chain by removing all chains where the part size appears.

5.2 Exploring orderings

The subroutine EXPLOREORDERINGS in Algorithm 1 performs a stochastic search of orderings resulting in low wastage layouts. The process starts from a random order and iteratively considers possible improvements by swapping two parts. At each iteration, we perform a swap and recompute a layout using the docking algorithm. If wastage is reduced the swap is accepted, otherwise it is rejected. We apply the process for a number of iterations and keep the best ordering found as the starting point. We use $|D(\mathbf{X})|^2$ iterations, where $|D(\mathbf{X})|$ is the number of parts. For each ordering, we use a fast docking algorithm to compute a layout with low wastage.

In addition to the random orderings, we also consider two standard heuristic orders: parts ordered by decreasing area, and parts ordered by decreasing maximum extent.

Docking algorithm. The docking algorithm places each part in order by 'dropping' the next part on the current layout either from the right, or from the top. It locally searches for the best placement of each part, according to a criterion that minimizes wastage. The result is a layout including all parts.

Given the layout so far our algorithm searches for the best orientation and best position for the next part. We denote by L_{i-1} the layout obtained for the $i-1$ first parts, and by $L_i \leftarrow L_{i-1} \triangleleft_{pos} p_i$ the layout obtained by adding the next part at position pos. The docking position pos is computed from a drop location (s, x, o), with $s \in \{top, right\}$, x a position along the corresponding axis and $o \in \{0, \pi/2, \pi, 3\pi/2\}$ an orientation (more orientations could be used).

The pseudo code for the docking algorithm is given in Algorithm 7. The drop locations are ranked according to a docking criterion that we denote $D(L_{i-1}, p_i, pos)$, explained next. The docking positions are computed from the drop locations by the `ComputeDockingPosition` subroutine. It is efficiently implemented by maintaining the right/top height-fields of the current layout as illustrated in Figure 8. Whenever evaluating a drop location we use the height-fields to quickly compute the docking positions that bring the part in close contact with the current layout.

Algorithm 7: DOCKING

Input: Set of parts P, order O, master board dimensions $W \times H$
Output: A layout L

```
foreach part p_i ∈ P following order in O do
    best ← ∅ ;
    bestscore ← 1 ;
    foreach drop location (s, x, o) do
        pos ← ComputeDockingPosition(p_i, (s, x, o)) ;
        score ← D(L_{i-1}, p_i, pos) ;
        if score < bestscore then
            best ← pos ;
            bestscore ← score ;
    L_i ← L_{i-1} ◁_pos p_i ;
return L_n;
```

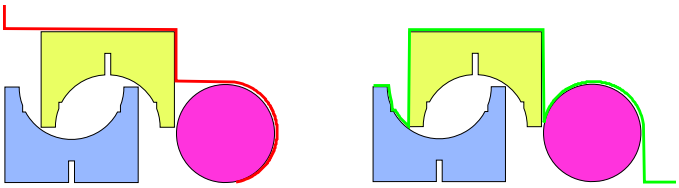


Fig. 8. Height-fields of the layout used to position the next part. **Left:** Height-field for dropping parts from the right (red curve). **Right:** Height-field for dropping parts from above (green curve). These height-fields are maintained every time a new part is added to the layout, and used for fast computation of the docking positions. Similar height-fields are pre-computed for the left/bottom of the parts.

Docking criterion. The docking criterion considers wastage as the primary objective, where wastage is defined by the ratio of occupied area divided by the bounding rectangle area of the layout. We denote $W(L_i)$ the wastage of a layout including up to part i. It is obtained as $W(L_i) = \frac{\sum_{k=0}^{i} A(p_k)}{A(box(L_i))}$ where A measures area and $box(L)$ is the bounding rectangle of the layout.

However, as the algorithm heuristically docks parts in sequence it cannot foresee that some spaces will be definitely enclosed. In particular, for newly inserted *concave* parts there are often multiple orientations of the part resulting in the same wastage: if the concavity remains empty there is no preferred choice. However, some choices are indeed better than others. If the concavity faces an already placed object, then further docking *within* the concavity will never be possible. This is illustrated in Figure 10, left.

We therefore propose a second criterion that discourages such bad choices. The idea is to estimate the space that will be definitely enclosed when a part is added to the current layout. This is done efficiently by considering the enclosed space between the height-field of the current layout and the height-field of the added part, along both horizontal and vertical directions.

Let $H^r(L)$ (respectively H^t) be the right (respectively top) height-field of layout L and $A(H^r(L))$ the area *below* it. The enclosed area is then defined as:

$$E(L_{i-1}, p_i, pos) = \sum_{s \in \{r,t\}} \max\left(0, A(H^s(L_{i-1} \triangleleft_{pos} p_i)) - A(H^s(L_{i-1})) - A(p_i)\right)$$

with $A(p_i)$ the area of part p_i. Note the max that clamps negative values: this is due to cases where the part nests in a concavity below the height-field of the other direction.

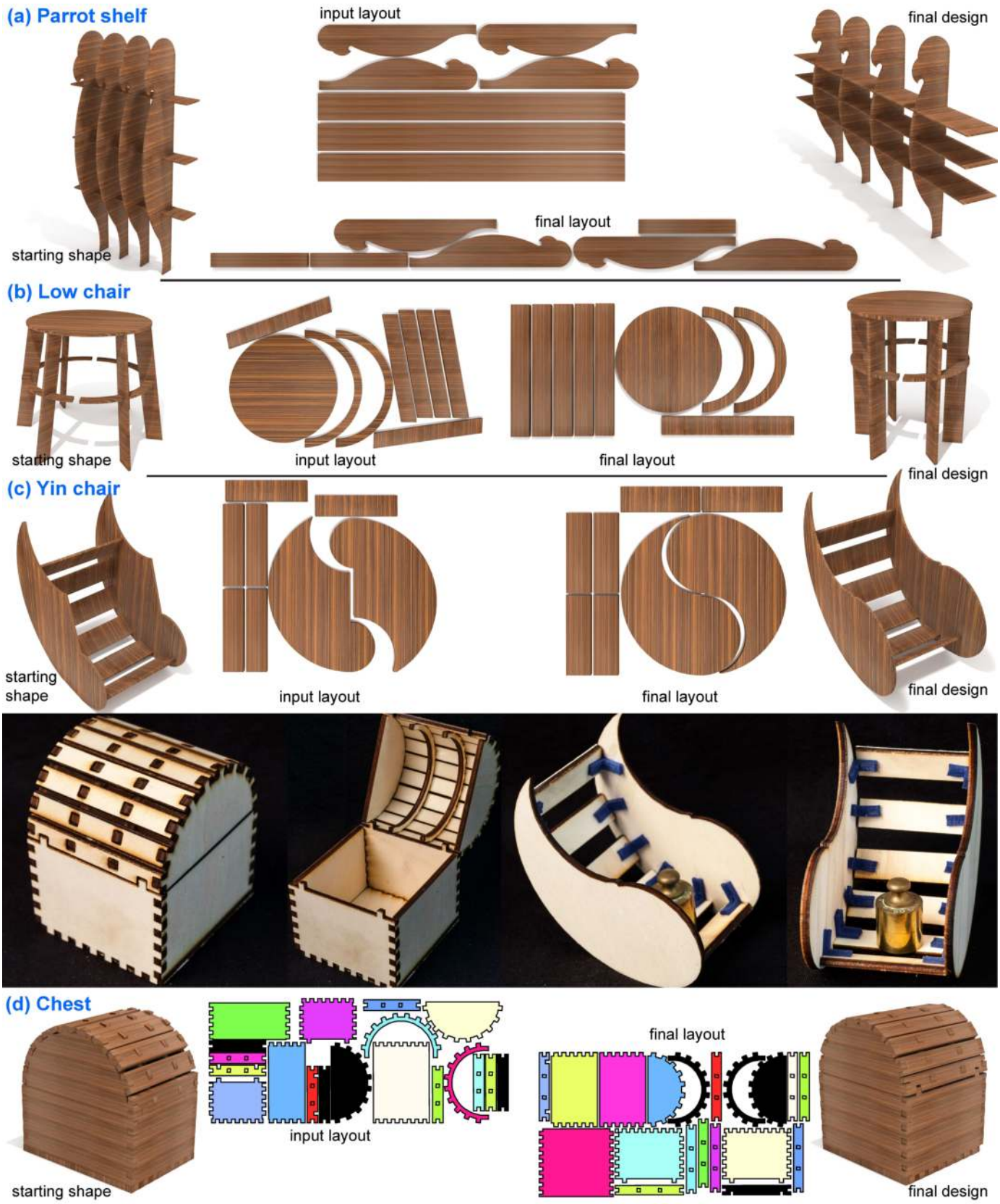


Fig. 9. Designs created using our system. Each design is shown with initial shape, starting layout, optimized layout, and final design.

The enclosed space is used as a tie-breaker when docking positions produce the same wastage values; therefore $D(L_{i-1}, p_i, pos)$ returns the vector $(W(L_{i-1} \lhd_{pos} p_i), E(L_{i-1}, p_i, pos))$. The effect of the enclosed area criterion is shown in Figure 10.

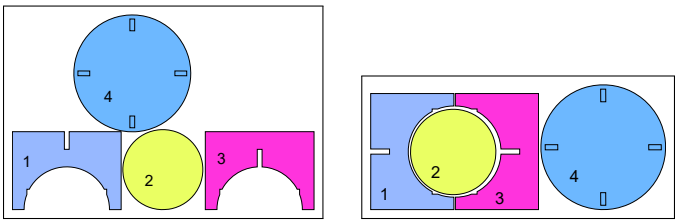


Fig. 10. Layouts obtained with the same docking order. **Left:** Without taking enclosed area into account, part #1 is placed with the concavity against the bottom packing border. This prevents part #2 to nest within and cascades into a series of poor placements. **Right:** Taking into account enclosed areas results in a placement of the part #1 that allows nesting of the part #2 and produces a layout with lower wastage.

6 Results and Discussion

We used our system for various design explorations. As the complexity of the designs grows beyond 4-6 planks, the utility of the system quickly becomes apparent. Note that the design constraints (see Figure 3), by coupling different object parts, make the optimization challenging by preventing independent adaptation of part sizes. By offloading material usage considerations to the system, the user can focus on the design. Note that even when changes to the design are visually subtle, material utilization often increases significantly.

Design examples. We used our system to design and fabricate a range of examples comprising rectangular and/or curved parts. We fabricated fullscale and miniature models of designed furniture. Models were made from MDF of 3 mm thickness and MDF of 30 mm thickness. The designs are easy to manufacture in batches since after design layout optimization they typically fit master boards completely: there is no need to attempt to reuse leftover pieces of wood, and switching boards requires little clean up.

Fig. 11. Various material-driven design and fabrication examples. In each row, we show initial design (with material space layout inset), optimized design result (with material space layout inset), along with final cutout assembled model. Note that the design changes are often subtle, but still leads to significant improvement in material usage.

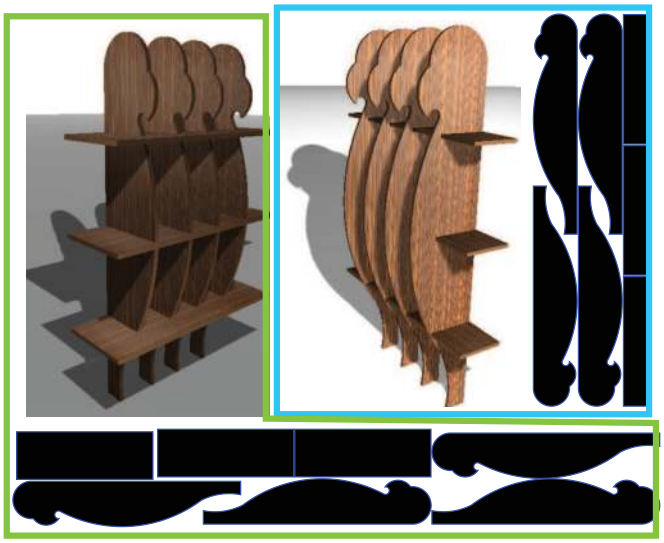


Fig. 12. Two different design suggestions (green has wastage ratio 0.14, blue has wastage ratio 0.15) for the parrot-shelf. Original design with another design suggestion is shown in Figure 9.

We directly output the cutting plan for the laser cutter (or CNC machine) from the design layout, adding connectors for planks sharing an edge, if needed. These are conveniently detected since planks exactly overlap on edges in the 3D design. The connectors are either *finger joints*, which are both strong after gluing and easy to assemble; *cross connectors* for interleaved planks, or dowel-jointed for thicker materials (20 mm and 30 mm thickness).

Figures 9, 11, and 13 show various results. Table 1 gives an overview of the complexity of each model, and the gains obtained by our system. The system performs at interactive rates on a laptop taking from a few seconds to 3-4 minutes for the larger examples. Note that speed depends on how many parallel exploration threads are pursued.

Figure 15 shows additional examples inspired by existing furniture, optimized for different masterboard sizes. In all cases the system significantly reduces wastage. This

TABLE 1
Statistics for cut design showing the number of planks, number of constraints, material wastage ratio before and after the design suggestions/optimization.

	#planks	#constraints	ratio before	ratio after
Figure 1	4	21	0.22	0.11
Figure 9a	7	33	0.34	0.08
Figure 9b	9	NA	0.34	0.20
Figure 9c	8	NA	0.24	0.17
Figure 9d	16	NA	0.21	0.14
Figure 11a	6	22	0.15	0.04
Figure 11b	11	41	0.15	0.03
Figure 11c	8	13	0.26	0.03
Figure 11d	16	29	0.11	0.02
Figure 13a	12	42	0.4	0.03
Figure 13b	15	66	0.19	0
Figure 13c	8	41	0.13	0
Figure 16	11	57	0.11	0.04
Figure 15a	16	83	0.09	0
Figure 15b	16	83	0.08	0
Figure 15c	11	60	0.12	0.01
Figure 15d	11	60	0.06	0.02

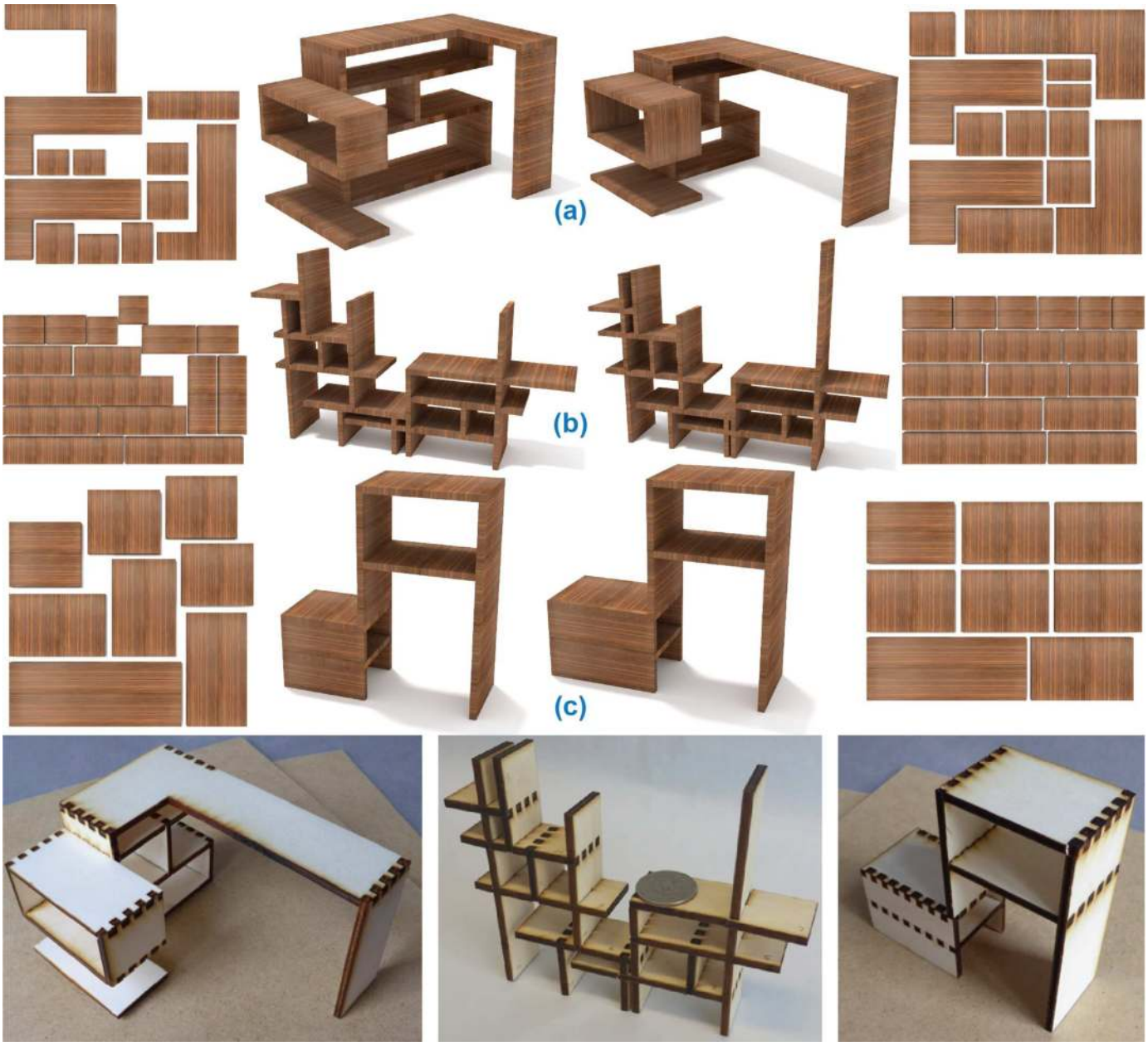


Fig. 13. Flat pack furniture optimized using our system. For each example, we show the initial layout and design, the final layout and design, and the fabricated prototype. The final wastage ratios for the three examples are $0.03, 0, 0$, respectively. (Please note that for visualization, slight gaps are added between parts in the final layout figure.)

also reveals how different runs can find different designs, in particular with the drawer example where the first run produces undesirable changes rejected by the user.

Figures 1 and 9 show results for objects with curved parts. Figure 5 shows some intermediate shapes as the design evolves for the coffee-table (Figure 1) and the low-chair (Figures 9-top) examples. Figure 12 shows alternate designs discovered by the algorithm for the Parrot shelf. While they have slightly higher wastage they offer interesting variations that the user might prefer.

Figures 1 was fabricated using a CNC machine. The optimized design achieved less than 0.1 material wastage, although one can achieve zero wastage by deciding to pick a rectangular top – a decision that can be made after layout optimization as this opportunity is revealed. An allowable range was specified for the height and the bases were marked as symmetric as input design constraints. In the case of the parrot-shelf (Figure 9a), the user indicated minimum and maximum range for the horizontal shelves along with desired range for the shelf heights.

As described, parameteric designs are easily supported and optimized for in our framework. Figures 9b-d show three such examples. In each case, additional constraints were provided to keep the objects within a given volume. The parts of the objects are all tightly coupled making these challenging examples to optimize for. Figure 14 illustrates optimization of two copies of a parametric design. In one case (top row) the user decided to constrain the angles of packed parts to 0 or π – this is for instance useful to align with a pattern printed on the masterboard, or to align with

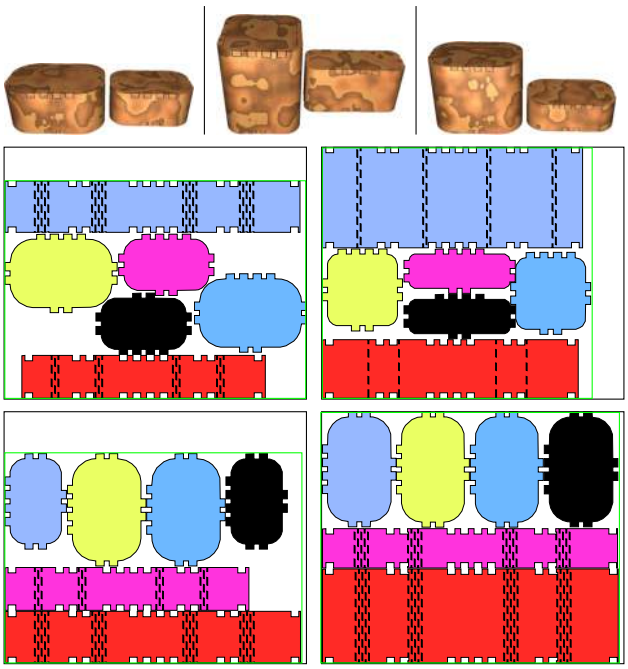


Fig. 14. **Top**, from left to right: initial design, design optimized with 0-180 degree constraint (middle row), design optimized freely (bottom row). This case is challenging due to the inter-dependencies between dimensions, and the presence of the dents. **Middle**: Optimized design constraining the angle to 0-180 degree rotations only (note how the four top dents on the side parts are either up or down). Wastage goes from 26.4% down to 12.1%. **Bottom**: Optimization using all angles. Wastage goes from 22.2% down to 11.8%. (Wastage is computed using the tightest bounding box around the parts, show in green. The black outline is the maximum extent of the masterboard.)

wood fibers. Figure 18 shows how usage increases with iterations during optimization, for both the 0 and π case and when angles are free. For each case the Figure reports three runs; while not all runs perform the same they all reduce wastage significantly.

Figure 11a shows a L-shaped work table. The user specified a target height for the design and a maximum work volume. Note that the legs of the table were also constrained to not change more than $1/4$ of original dimensions to prevent unwanted design changes. Figure 11b shows a coupled shelf and table design where height of shelves and tabletop were similarly constrained. Figure 11c shows a stylized chair, where both the chair seat height and chair width were constrained not to change beyond a margin. Figure 11d shows multiple designs covering 2 master boards. The second master board is used as an overflow when docking can no longer fit a part in the first. The layouts are slid independently.

Figure 16 shows a design study involving a chair with two chairs and a platform from a single master board. In this example, wastage was reduced from 11% to 4%.

Comparison. We now evaluate the relative importance of the key algorithm steps. Figure 17a shows the importance of the docking criteria introduced in Section 5.2. We ran 500 random runs of our proposed packing algorithm with ('ours') and without ('baseline') the docking criteria on the coffee-table example. We sort the runs based on resultant usage (no shape optimization is performed here) and plot the two conditions. The docking criteria consistently resulted in

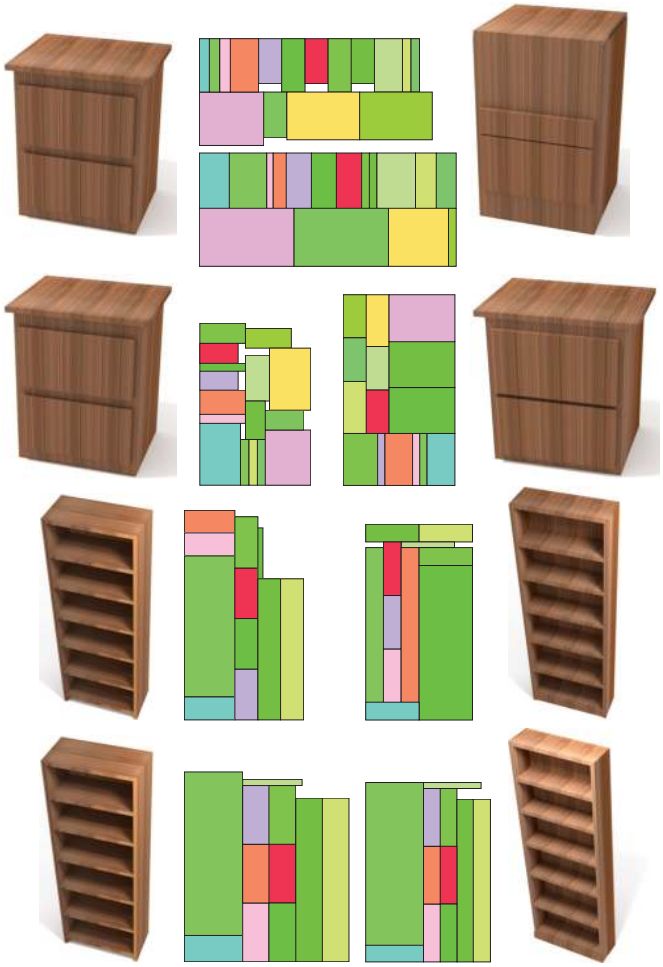


Fig. 15. Ikea inspired designs. **Two top rows**: Drawer model. A first optimization reaches zero-waste (down from 9.3% wastage). However the solution changes the height of the drawers. The second result (below) also reaches zero-waste (down from 8.2% wastage) with a design closer to the original. **Two bottom rows**: A same shelf optimized with two different master boards. In the first (top) case wastage goes down from 12.1% to 0.9%, in the second case it goes down from 5.7% to 1.9%.

Fig. 16. Designing a chair with two chairs and a platform from a single master board. Top row shows the original design (11% wastage), while the bottom row shows the final design (4% wastage).

10-15% less wastage.

Figure 17b shows usage improvement over one explo-

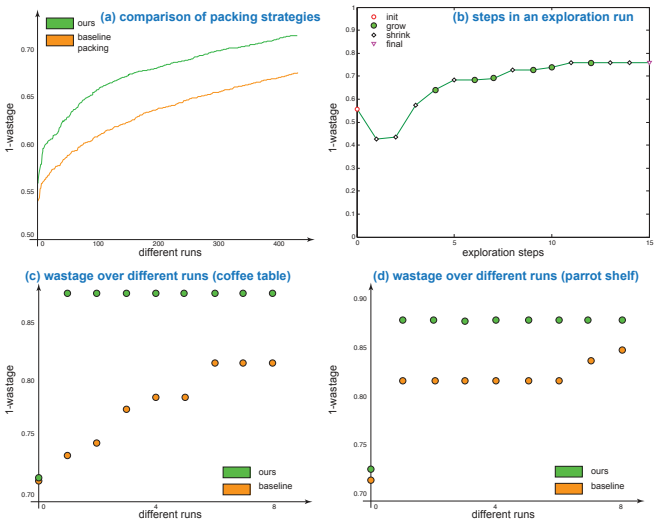


Fig. 17. Comparison of our algorithm against baseline alternatives. Higher is better. Please refer to the text for details.

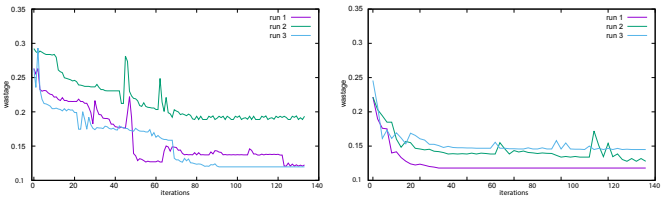


Fig. 18. Decrease in wastage over time during optimization, using the two flex-box models of Figure 14. All runs use the same master board and initial design parameters. **Left:** Three runs with angles constrained to 0-180 degrees. The run with the worst performance still reduces wastage by 10.4% compared to the initial packing. **Right:** Three runs with all angles. Average wastage reduction is 10%.

ration run on the coffee-table sequence. The legend explains which step (grow, shrink, etc.) is being performed. While this is the result from a single thread, many similar threads are simultaneously explored. The few best results are then presented to the user as suggestions. Figure 17c-d compare the importance of analyzing the material space layout to decide which plank to change and how. As baseline, we selected planks at random and perform either a grow or shrink sequence with equal probability. Note that our method consistently outperforms the alternative approach.

Design sessions. We asked second year art students (6 subjects) from a design college to try our system. Figures 11b-d show a selection of their designs. These particular students had performed a very similar task as part of their first year assignment – 'design furniture of your choice making best use of the provided piece of MDF board.' Hence, they were very aware of the implicit link between design and material usage. Previously, they had used commercial 3D modeling tool (Rhinoceros, Solidworks, Sketchup Pro) for designing and mainly Illustrator for manually laying out the designs. They recalled the frustration of having to switch between the different 2D-3D design representations. First, the students sketched design concepts before using our system. Then, they used the exploration interface on their designs to reduce wastage. Note that visually the initial sketch and final design can look similar, despite the increase in material utilization, which is desirable in terms of preserving the original design.

Overall, the feedback was positive. They appreciated being able to easily move between 2D↔3D, and not having to explicitly worry about material utilization. They appreciated the suggestions, instead of previous attempts using trial-and-error iterations between various softwares to reduce material wastage.

Limitations. Currently, the algorithm can only make topological changes only for parameteric models. This will be an interesting future direction to pursue for constrained models. Our docking approach cannot nest parts into holes of other parts, a more advanced algorithm would be required. A more material-induced restriction arises when the starting layout does not leave much space to optimize over. This effectively means that the degree of freedom for the design is low. Adding more planks does reduce this problem (by providing additional freedom). However, beyond 25-30 planks, the exploration of the shape space becomes slow as there are too many paths to explore. One option is to limit exploration to only a subset of planks at a time, but then again, very desirable design configurations may be missed.

7 Conclusions and Future Work

We investigated how design constraints and material usage can be linked together towards form finding. Our system dynamically discovers and adapts to constraints arising due to current material usage, and computationally generates design variations to reduce material wastage. By dynamically analyzing 2D material space layouts, we determine which and how to modify object parts, while using design constraints to determine how the proposed changes can be realized. This interplay results in a tight coupling between 3D design and 2D material usage and reveals information that usually remains largely invisible to the designers, and hence difficult to account for. We used our system to generate a variety of shapes and demonstrated different margins of wastage reduction.

Currently, we do not consider the stability of the produced furniture nor the durability of the joints. This could be integrated as dynamic constraints following previous work on structural reinforcement [4] and shape balancing [5]. We would like to generalize the framework to handle other types of materials e.g., fabric, plastic, etc. that can be easily cut and more interestingly bend to have freeform shapes for manufacturing garments, toys, etc. Note that the packing problem will still be in 2D for such developable pieces. This can help produce interesting freeform shapes, while still making efficient use of materials. In terms of allowed part deformations, operations like stretching and sheering can also be considered along with allowing discrete changes in the form of splitting and merging of parts. Finally, in the packing stage, we would like to evaluate the effect of different strategies for selecting and swapping parts.

References

[1] L. Luo, I. Baran, S. Rusinkiewicz, and W. Matusik, "Chopper: Partitioning models into 3D-printable parts," *SIGGRAPH Asia*, vol. 31, no. 6, 2012.

[2] J. Vanek, J. A. G. Galicia, B. Benes, R. Měch, N. Carr, O. Stava, and G. S. Miller, "Packmerger: A 3d print volume optimizer," *CGF Eurographics*, vol. 33, no. 6, 2014.

[3] X. Chen, H. Zhang, J. Lin, R. Hu, L. Lu, Q. Huang, B. Benes, D. Cohen-Or, and B. Chen, "Dapper: Decompose-and-pack for 3d printing," *SIGGRAPH Asia*, vol. 34, no. 6, pp. 213:1–213:12, 2015.

[4] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch, "Stress relief: Improving structural strength of 3d printable objects," *ACM SIGGRAPH*, vol. 31, no. 4, pp. 48:1–48:11, 2012.

[5] R. Prévost, E. Whiting, S. Lefebvre, and O. Sorkine-Hornung, "Make it stand: Balancing shapes for 3d fabrication," *ACM SIGGRAPH*, vol. 32, no. 4, pp. 81:1–81:10, 2013.

[6] W. Wang, T. Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, and X. Liu, "Cost-effective printing of 3d objects with skin-frame structures," *SIGGRAPH Asia*, vol. 32, no. 6, pp. 177:1–177:10, 2013.

[7] J. Dumas, J. Hergel, and S. Lefebvre, "Bridging the gap: Automated steady scaffoldings for 3d printing," *ACM SIGGRAPH*, vol. 33, no. 4, pp. 98:1–98:10, 2014.

[8] M. Shugrina, A. Shamir, and W. Matusik, "Fab forms: Customizable objects for fabrication with validity and geometry caching," *ACM SIGGRAPH*, vol. 34, no. 4, pp. 100:1–100:12, 2015.

[9] C.-W. Fu, P. Song, X. Yan, L. W. Yang, P. K. Jayaraman, and D. Cohen-Or, "Computational interlocking furniture assembly," *ACM SIGGRAPH*, vol. 34, no. 4, pp. 91:1–91:11, 2015.

[10] G. Brennan, J. Brown, M. Docherty, and B. Tullett, *FlatPack/PlaskaPaczka*, K. Cockburn, Ed. The Caseroom Press, 2006.

[11] Q. Zhou, J. Panetta, and D. Zorin, "Worst-case structural analysis," *ACM SIGGRAPH*, vol. 32, no. 4, pp. 137:1–137:12, 2013.

[12] L. Lu, A. Sharf, H. Zhao, Y. Wei, Q. Fan, X. Chen, Y. Savoye, C. Tu, D. Cohen-Or, and B. Chen, "Build-to-last: Strength to weight 3d printed objects," *ACM SIGGRAPH*, vol. 33, no. 4, pp. 97:1–97:10, 2014.

[13] K. Hu, S. Jin, and C. C. Wang, "Support slimming for single material based additive manufacturing," *Computer-Aided Design*, vol. 65, pp. 1 – 10, 2015.

[14] K. Hildebrand, B. Bickel, and M. Alexa, "crdbrd : Shape fabrication by sliding planar slices," *CGF Eurographics*, vol. 31, no. 2, 2012.

[15] Y. Schwartzburg and M. Pauly, "Fabrication-aware design with intersecting planar pieces," *CGF Eurographics*, vol. 32, no. 2, pp. 317–326, 2013.

[16] W. CC, Y. Zhang, and H. Sheung, "From designing products to fabricating them from planar materials," *IEEE Comput Graph Applications*, 2010.

[17] J. Jylänki, "A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing," *Retrieved from http://clb.demon.fi/files/RectangleBinPack.pdf*, 2010.

[18] D. Saakes, T. Cambazard, J. Mitani, and T. Igarashi, "Paccam: Material capture and interactive 2d packing for efficient material usage on cnc cutting machines," in *Proc. UIST*, 2013, pp. 441–446.

[19] J. McCrae, K. Singh, and N. J. Mitra, "Slices: A shape-proxy based on planar sections," *SIGGRAPH Asia*, vol. 30, no. 6, 2011.

[20] P. Cignoni, N. Pietroni, L. Malomo, and R. Scopigno, "Field-aligned mesh joinery," *ACM TOG*, vol. 33, no. 1, January 2014.

[21] X.-Y. Li, C.-H. Shen, S.-S. Huang, T. Ju, and S.-M. Hu, "Popup: automatic paper architectures from 3d models," *ACM SIGGRAPH*, vol. 29, no. 4, pp. 111:1–9, 2010.

[22] S. Mueller, P. Lopes, and P. Baudisch, "Interactive construction: Interactive fabrication of functional mechanical devices," in *Proc. UIST*, 2012, pp. 599–606.

[23] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R. W. Sumner, W. Matusik, and B. Bickel, "Computational design of mechanical characters," *ACM SIGGRAPH*, vol. 32, no. 4, pp. 83:1–83:12, 2013.

[24] M. Bächer, B. Bickel, D. L. James, and H. Pfister, "Fabricating articulated characters from skinned meshes," *ACM SIGGRAPH*, vol. 31, no. 4, 2012.

[25] J. Calì, D. Calian, C. Amati, R. Kleinberger, A. Steed, J. Kautz, and T. Weyrich, "3d-printing of non-assembly, articulated models," vol. 31, no. 6, pp. 130:1–130:8, 2012.

[26] A. Schulz, A. Shamir, D. I. W. Levin, P. Sitthi-amorn, and W. Matusik, "Design and fabrication by example," *ACM SIGGRAPH*, vol. 33, no. 4, pp. 62:1–62:11, 2014.

[27] Y. Zheng, H. Liu, J. Dorsey, and N. J. Mitra, "Ergonomics-inspired reshaping and exploration of collections of models," *IEEE TVCG*, 2015.

[28] T.-H. Kwok and C. C. Wang, "Shape optimization for human-centric products with standardized components," *Computer-Aided Design*, vol. 52, pp. 40–50, 2014.

[29] K. Xu, H. Zhang, D. Cohen-Or, and B. Chen, "Fit and diverse: Set evolution for inspiring 3d shape galleries," *ACM SIGGRAPH*, vol. 31, no. 4, pp. 57:1–10, 2012.

[30] J. O. Talton, D. Gibson, L. Yang, P. Hanrahan, and V. Koltun, "Exploratory modeling with collaborative design spaces," in *SIGGRAPH Asia*, 2009, pp. 167:1–167:10.

[31] N. Umetani, T. Igarashi, and N. J. Mitra, "Guided exploration of physically valid shapes for furniture design," *ACM SIGGRAPH*, vol. 31, no. 4, pp. 86:1–86:11, 2012.

[32] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or, "iwires: An analyze-and-edit approach to shape manipulation," *ACM SIGGRAPH*, vol. 28, no. 3, pp. #33, 1–10, 2009.

[33] W. Xu, J. Wang, K. Yin, K. Zhou, M. van de Panne, F. Chen, and B. Guo, "Joint-aware manipulation of deformable models," *ACM SIGGRAPH*, vol. 28, no. 3, pp. 35:1–35:9, Jul. 2009.

[34] Y. Zheng, H. Fu, D. Cohen-Or, O. K.-C. Au, and C.-L. Tai, "Component-wise controllers for structure-preserving shape manipulation," in *CGF Eurographics*, vol. 30, no. 2, 2011.

[35] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM TOG*, vol. 21, no. 3, pp. 362–371, 2002.

Bongjin Koo is a PhD student in Computer Science supervised by Prof. Niloy J. Mitra at the University College London. His current research focuses on addressing problems in computational fabrication. Before starting a PhD, Bongjin obtained a Masters degree in computer graphics, vision, and imaging at the University College London.

Jean Hergel is a PhD student specializing in computer graphics at the Inria Nancy Grand Est under the supervision of Sylvain Lefebvre. He received his Masters in computer science on Machine Learning, Recognition and Reasoning in 2013 from the Université de Lorraine. His research interests are customized fabrication, optimization, and computer graphics.

Sylvain Lefebvre is a researcher at Inria since 2006. His main research focus is to simplify content creation, in particular to produce highly detailed patterns, structures and shapes. In 2010, he was honored with the EUROGRAPHICS Young Researcher Award. Since 2012, he is the principal investigator of the ERC ShapeForge project, which focuses on the modeling of complex, detailed functional shapes for additive manufacturing.

Niloy J. Mitra is a Professor of Computer Science at University College London. His research focuses on algorithmic issues in shape analysis and geometry processing. He received the ACM SIGGRAPH Significant New Researcher award in 2013 and the BCS Roger Needham award in 2015. Since 2013, he is the the principal investigator of the ERC SmartGeometry project.