Reconfigurable Interlocking Furniture

PENG SONG*, University of Science and Technology of China
CHI-WING FU*, The Chinese University of Hong Kong
YUEMING JIN, The Chinese University of Hong Kong
HONGFEI XU, University of Science and Technology of China
LIGANG LIU†, University of Science and Technology of China
PHENG-ANN HENG, The Chinese University of Hong Kong and Shenzhen Institutes of Advanced Technology, CAS
DANIEL COHEN-OR, Tel Aviv University

Fig. 1. Reconfigurable assemblies of different functionality produced by our method: Ladder, Stool, and Hand Truck (left to right).

Reconfigurable assemblies consist of a common set of parts that can be assembled into different forms for use in different situations. Designing these assemblies is a complex problem, since it requires a compatible decomposition of shapes with correspondence across forms, and a planning of well-matched joints to connect parts in each form. This paper presents computational methods as tools to assist the design and construction of reconfigurable assemblies, typically for furniture. There are three key contributions in this work. First, we present the compatible decomposition as a weakly-constrained dissection problem, and derive its solution based on a dynamic bipartite graph to construct parts across multiple forms; particularly, we optimize the parts reuse and preserve the geometric semantics. Second, we develop a joint connection graph to model the solution space of reconfigurable assemblies with part and joint compatibility across different forms. Third, we formulate the backward interlocking and multi-key interlocking models, with which we iteratively plan the joints consistently over multiple forms. We show the applicability of our approach by constructing reconfigurable furniture of various complexities, extend it with recursive connections to generate extensible and hierarchical structures, and fabricate a number of results using 3D printing, 2D laser cutting, and woodworking.

CCS Concepts: • **Computing methodologies** → *Shape modeling*; • **Applied computing** → *Computer-aided manufacturing*;

Additional Key Words and Phrases: reconfigurable, furniture assembly, computational design, mechanical interlocking, joints, co-analysis, dissection

ACM Reference Format:
Peng Song, Chi-Wing Fu, Yueming Jin, Hongfei Xu, Ligang Liu, Pheng-Ann Heng, and Daniel Cohen-Or. 2017. Reconfigurable Interlocking Furniture. *ACM Trans. Graph.* 36, 6, Article 174 (November 2017), 14 pages.
https://doi.org/10.1145/3130800.3130803

1 INTRODUCTION

Reconfigurability refers to the capability of an assembly structure to be re-assembled into different forms (configurations) for use in different situations; see Figs. 1 and 2. It is a cost-effective and desirable property that promotes the reuse of component parts and increases their utilization. In practice, this idea has been realized in many assembly structures, most notably in furniture design.

Designing and computing reconfigurable assemblies is a nontrivial problem. Recent works in inverse procedural modeling [Kalojanov et al. 2016; Liu et al. 2015; Zheng et al. 2013] only scratched the surface of the problem by exploring replaceable substructures

*Both authors contributed equally to the paper
†Corresponding author: lgliu@ustc.edu.cn (Ligang Liu)

0730-0301/2017/11-ART174 $15.00
https://doi.org/10.1145/3130800.3130803

Fig. 2. Example: modular furniture designed by T. Maitz.

in a given 3D model. While a recent work by Garg et al. [2016] presented interactive visualization and monitoring tools to aid the design of reconfigurable structures, these tools mostly concerned with collision detection in 3D space. At present, reconfigurable models are mainly developed by professionals in a case-by-case manner for rather limited classes of shape, typically furniture.

This paper presents computational methods that serve as tools to assist the design and construction of reconfigurable furniture from a given set of 3D designs, where users specify mainly the appearance of each form. To achieve this, there are three major challenges:

- First, for cost effectiveness, we should maximize the reuse of component parts across different forms. This requires a *compatible decomposition* of multiple input designs into parts that may be reused. This means that the decomposition should produce parts of exactly the same geometry for reuse in different designs, and yet, the parts can be reused in different orientations.
- Second, we have to ensure *compatible joint connections* among the parts in different designs, so that the same part, when reused in different designs, may connect to different neighbors through the same joint geometry fixed on the part.
- Lastly, for every furniture assembly, the component parts should connect steadily together, so that the assembly can function coherently for casual usage. Yet, we should be able to *repeatedly disassemble and re-assemble the structure*; hence, there should not be irreversible assembly actions such as gluing or nailing.

To meet the above challenges, we develop our methods based on two foundations. First, we present the problem of co-decomposing multiple shapes into compatible parts as a *weakly-constrained dissection problem*, which is a well-known recreational geometric problem [Frederickson 1997]. The basic dissection problem is to find a decomposition of two given 2D (equal area) shapes into a finite number of congruent polygons [Kranakis et al. 2000; Zhou and Wang 2012]. We consider our compatible decomposition as a weaker setting of the classical geometric dissection: (i) the co-decomposition does not necessarily require a one-to-one correspondence of the decomposed parts; and (ii) we allow small modifications (e.g., local deformation) on the input shapes to obtain exact part correspondence. To maximize part reuse and cost-efficiency of the assemblies, our co-decomposition aims at increasing the part commonality among the shapes while minimizing the total number of parts.

Second, we need to create compatible joints on a common set of parts, so that the parts can be reused to steadily build different assemblies. To this end, we explore steady structures based on *mechanical interlocking* [Fu et al. 2015; Song et al. 2012], whose goal is *to immobilize every single part* and *every subset of parts* in the assembly, except for one specific *key part*, which is the only movable part in the entire assembly. Such immobilization helps tighten the assembly structure, and yet still allows repeated disassembly and re-assembly without irreversible actions. The challenge here is that we have to carefully plan a network of joints to immobilize the non-key parts and subsets of parts in each assembly. Moreover, we have to achieve the immobilization for each of the assemblies, which share a common set of parts. The requirements of interlocking every single assembly while having compatible joint connections using a common set of parts make the joint planning extremely challenging. The problem complexity in computing the solution exceeds the capability of all previous works [Fu et al. 2015; Song et al. 2012].

To achieve the above goals, we present the following contributions:

- First, we formulate the compatible co-decomposition as *an optimization over dynamic bipartite graphs*, whose nodes represent parts and edges represent part correspondences across different designs (configurations). Using this formulation, we iteratively construct parts with correspondence while maximizing parts reuse, minimizing the total number of parts, and reducing necessary modifications on the input shapes.
- Second, we develop the *joint connection graph* to model the joint connections between parts over multiple designs. Based on this graph, we can model the search space of planning compatible joints in reconfigurable assemblies, and formulate various necessary constraints to efficiently find the solutions.
- Third, we formulate the *backward interlocking* and *multi-key interlocking* models for connecting parts in reconfigurable assemblies, and devise a *co-construction method* to iteratively plan compatible joints over multiple assemblies with interlocking.

Lastly, we extend our framework to support *recursive connection* of parts, and generate families of reconfigurable furniture that are hierarchical and extensible. We present results of reconfigurable furniture in various complexities in Section 7, and fabricate some of them using 3D printing, 2D laser-cutting, and woodworking.

2 RELATED WORK

Reusable Substructures. In shape synthesis and inverse procedural modeling, one common approach is to extract parts or substructures from given object(s) and then combine them like tiles to create new models. Kreavoy et al. [2007] developed a 3D modeling system for users to compose new models by interchanging parts extracted from a compatible segmentation of existing models. Following this modeling by parts idea, Jain et al. [2012] synthesized new models from a database by recombining parts from different shapes based on various geometric constraints, while Duncan et al. [2016] constructed interchangeable components by jointly deforming and segmenting multiple models with coherent appearance.

On the other hand, Zheng et al. [2013] identified compatible functional substructures in input shapes and rearranged parts to generate new models accordingly. Liu et al. [2015] explored replaceable substructures to synthesize plausible shape variations with local and global constraints. More recently, Kalojanov et al. [2016] proposed to automatically generate a small set of simple building blocks by

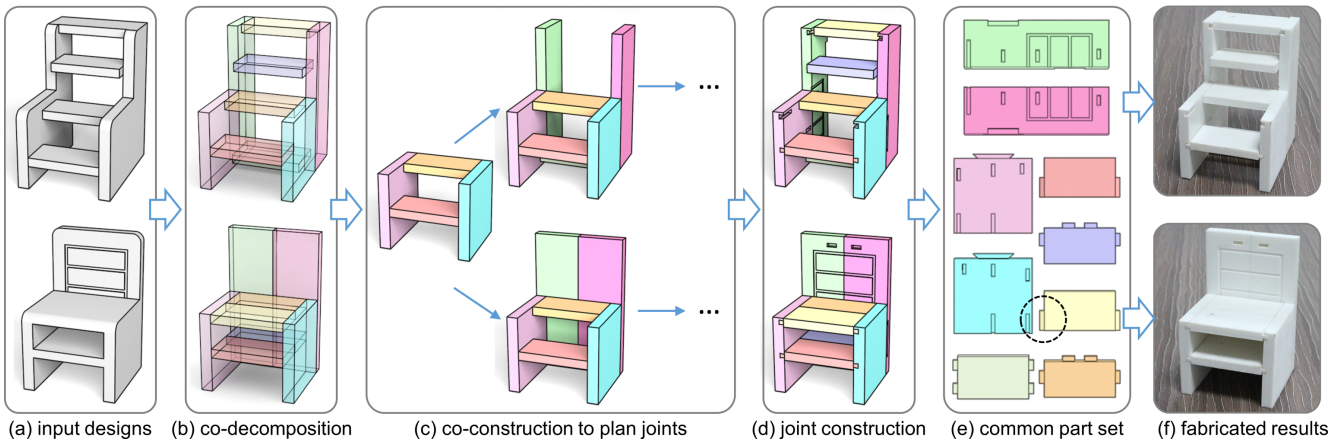


Fig. 3. Overview of our approach. Given two input designs (a), e.g., Step Ladder (top) and Chair (bottom), we first co-decompose them into compatible 3D parts with correspondences highlighted in the same color (b); see Section 4 for detail. We then analyze the parts connectivity and iteratively plan compatible joints over corresponded parts with mechanical interlocking (c); see Sections 5 and 6. Lastly, we create joint geometry on each part (d) to produce a common set of parts (e), fabricate the set, and employ it to build different assemblies (f).

simplifying a tiling grammar. Our work deals with a different decomposition problem, where we co-decompose multiple 3D shapes into parts with correspondence and small modifications, so that we can create a common set of parts for assembling the given shapes.

Co-analysis of Shape Collections. Simultaneously analyzing multiple objects of the same class enables us to acquire richer shape information about the class. Various methods related to this idea have been developed to co-segment a given set of shapes, e.g., [Hu et al. 2012; Huang et al. 2011; Sidi et al. 2011], or to refine the co-segmentation with user inputs, e.g., [Wang et al. 2012].

Besides co-segmentation, Yumer et al. [2012] performed a co-abstraction to achieve identity-preserving shape abstraction. van Kaick et al. [2013] co-analyzed a set of shapes hierarchically to discover relations between functionally-equivalent shape parts in the set. Zheng et al. [2014] discovered semantically-related parts across models by comparing part arrangements extracted from each model, while Averkiou et al. [2016] facilitated shape co-alignment by exploring rotational auto-correlations among the shapes.

Dissection of 2D/3D Shapes. Our compatible decomposition problem is related to the classical dissection problem. While many pioneering methods have been proposed for 2D shapes, e.g., triangle-square dissections [Cohn 1975], regular polygon dissections [Kranakis et al. 2000], and hinged dissections [Abbott et al. 2012], finding the optimal dissection of general shapes remains a difficult open problem. Later, Zhou and Wang [2012] developed an optimization method to create geometric dissection puzzles over a square or triangular lattice. Concurrent to our work, Duncan et al. [2017] studied approximate dissections between naturalistic 2D shapes, and presented a minimized deformation on the input shapes to achieve the dissection. Compared with the above works, we take multiple furniture-like 3D shapes as inputs, and aim to decompose these shapes into compatible parts with correspondence and joints while respecting the geometric semantics of the input shapes.

Transformable Objects. Recently, computer graphics community has a rising interest in fabricating mechanical assemblies that can be dynamically transformed into different forms. Bächer et al. [2012] and Calì et al. [2012] constructed ball-and-socket joints for making articulated models, while Coros et al. [2013], Ceylan et al. [2013], and Thomaszewski et al. [2014] constructed various dynamic mechanical assemblies with connectors such as gears and linkages.

Besides, Zhou et al. [2014] designed transformable objects that can be folded into a box, while Li et al. [2015] designed foldable furniture for space saving. Recently, Ureta et al. [2016] created working mechanisms from a set of disconnected parts by constructing mechanical joints in-between parts. Huang et al. [2016] produced transformable models with an associated motion from a source model to a target skeleton. Garg et al. [2016] developed an interactive design interface for reconfigurables with visualization and monitoring tools to help users explore parts collision and edit the geometry and motions of component parts.

In this work, we explore reconfigurable furniture built from a common set of parts, and focus on the geometric construction and assembly aspects, particularly on compatible parts and joints. This is a novel problem not explored in previous works.

Interlocking Assembly. Mechanical interlocking is an intriguing condition in 3D assemblies, where most component parts are immobilized by the geometric arrangement, thus enhancing the structural stability. This technique has been practiced in long-standing wooden architecture as well as in challenging games like the burr puzzle. In computer graphics research, some methods have been developed to construct assemblies with mechanical interlocking. Xin et al. [2011] created 3D interlocking puzzles by connecting multiple instances of a six-piece burr structure. Instead of repeating an existing structure, Song et al. [2012] generated interlocking assemblies by iteratively extracting parts from a voxelized 3D model with local interlocking. This method was later extended to handle smooth non-voxelized surface for 3D printing [Song et al. 2015].

Recently, Sun and Zheng [2015] created dynamically-interlocking assemblies by embedding the Rubik's Cube mechanism in given 3D shapes. Skouras et al. [2015] developed a computational design tool for 3D artworks assembled with interlocking paper elements, while Fu et al. [2015] formulated an interlocking scheme for frame-based structures such as furniture by planning and connecting local interlocking groups. Very recently, Song et al. [2016] developed CofiFab, which includes a formal model for interlocking 2D laser-cut parts into a convex polyhedron. Zhang and Balkcom [2016] presented a voxel-like interlocking cube set whose instances can be connected layer-by-layer into various voxelized shapes. Yao et al. [2017] constructed decorative joints for furniture design, some of which form interlocking geometries to enhance stability.

More than a single assembly, we simultaneously consider multiple assemblies built by a common set of parts, and develop the joint connection graph to model the solution space of joints in reconfigurable assemblies. Moreover, we formulate the backward interlocking model and multi-key interlocking model to support iterative planning of joints consistently for multiple assemblies.

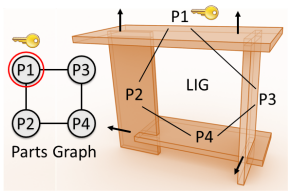
3 OVERVIEW

Our input is a set of 3D furniture designs, each represented as a triangular mesh without parts and joints. Hence, the input provides only the overall appearance of the designs. To facilitate the co-decomposition and parts reuse, the input designs should have fairly similar volume and substructure; see Fig. 3(a). To prepare them, we manually search over a large amount of existing designs (mainly photos in the Internet), look for those with similar substructures, and recreate the designs with compatible sizes using conventional 3D modeling tools. This manual preprocessing took around 15-30 minutes. Since the chance to encounter similar designs is not very high, we can employ shape retrieval methods to speed up the search, and create a new design by modifying an existing one.

Goal. Our goal in this work is summarized below:

> *To co-decompose the input designs into parts with correspondence and to co-construct compatible joints on the parts, so that we can produce a common set of parts and reuse it to assemble each design with mechanical interlocking.*

To achieve mechanical interlocking, we extend [Fu et al. 2015] to plan joints with *local interlocking groups* (LIG). Representing an assembly as a *parts-graph*, where nodes denote parts and edges denote joints between parts, *an LIG is a connected subgraph in the parts-graph, such that only a specific part (key) is mobile in the LIG*; see the inset figure for a simple example, where we plan the joints to restrict the movement of parts in the LIG, so that only part P_1 is mobile in the LIG. Therefore, if the key of an LIG (child) is a non-key part of another LIG (parent), the parent LIG can immobilize the child LIG. By successively forming LIGs in a parts-graph with carefully-planned joints, we can create dependency over all the LIGs, thus mechanically interlocking all the parts in the entire assembly. This is the central idea in [Fu et al. 2015], which we will employ and generalize in this work; see Section 6.



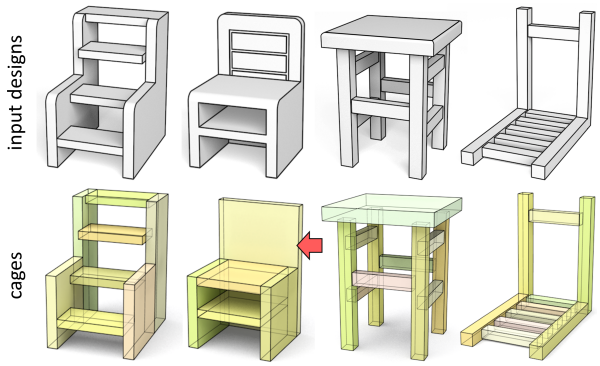


Fig. 4. Preprocessing #1: approximate each input design (top) by a set of simple cages (bottom), i.e., axis-aligned boxes.

Preprocessing. Given the input 3D designs, our method starts with two preprocessing steps on each design. In the first step, we fit a set of cages to approximate each design; see Fig. 4, where a part will later be constructed for each cage. In detail, we use an axis-aligned bounding box [Cohen et al. 1995] to model each cage, and fit cages with three criteria: i) minimize the number of cages in the approximation; ii) maximize the volume of each cage as a tight local bounding box; and iii) adjacent cages should remain contact without overlaps or gaps. These three criteria encourage simpler assemblies and better approximation of the design, and ensure that the parts can be fabricated and assembled. In our implementation, we arrange cages for each given design by fitting axis-aligned planes to each input design and then iteratively construct maximal boxes bounded by the extracted planes; see again Fig. 4.

In the second step, we identify geometric semantics in each design to ensure the resulting assemblies can be fabricated, assembled, and functioning as in the associated design [Koo et al. 2014; Rong et al. 2016]. Hence, we identify *contacts* between adjacent cages, groups of *symmetric* cages, and co-planar faces in different cages for *alignment*; see Fig. 5 for the illustrations. Note that the co-decomposition process (see Section 4) should maintain these constraints.

Main pipeline. After the preprocessing, we proceed to the following three major components in the main pipeline (see Fig. 3):

i) co-decomposition of multiple designs. We formulate the co-decomposition problem using *dynamic bipartite graphs* to model the part correspondence across designs. Here, we aim to optimize

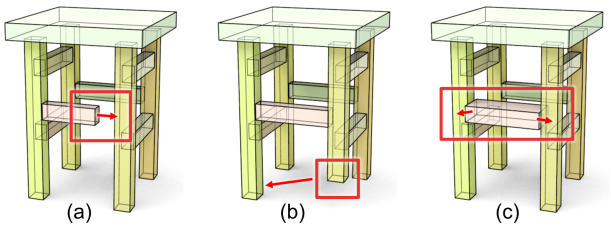


Fig. 5. Preprocessing #2: Geometric semantics: (a) contact, not preserved if a cage loses its contact with a neighbor; (b) symmetry, not preserved if a cage is no longer in sync with associated cages in the same symmetric group; and (c) alignment, not preserved if a cage is excessively resized and loses alignment with the associated cage(s).

the cages with necessary local modifications, such that we can maximize the number of correspondences for better parts reuse, while minimizing the total number of cages and the amount of local modifications. Hence, the problem involves a huge search space, and we solve it by iteratively constructing cage correspondences and locally modifying cages; see Section 4. Fig. 3(b) shows a result, where corresponded cages have the same color. Note also that for the cages of CHAIR shown in Fig. 4, the co-decomposition has modified a number of its cages for exact correspondences. See the large planar cage on the back of CHAIR; it has been divided into two.

ii) interlocking with compatible joints. After the co-decomposition, we create one part for each set of corresponded cages. To this end, we need to create compatible joints on the parts, and ensure that the joints lead to LIGs that mechanically interlock each assembly. Hence, we formulate the *joint connection graph* to encode parts connectivity over the designs and model the solution space of planning compatible joints; see Section 5. Furthermore, we develop the *backward interlocking model* and *multi-key interlocking model*, and employ these models to further devise a *co-construction method* to iteratively plan joints over multiple designs; see Fig. 3(c) and Section 6.

iii) fabrication and assembly. Lastly, we create joint geometry on each part, fabricate the parts (see Fig. 3(d&e)), and reuse the common set of parts to assemble the designs; see Fig. 3(f).

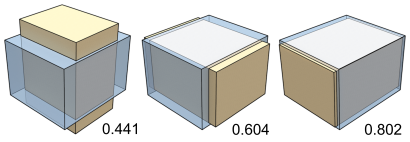
4 CO-DECOMPOSING MULTIPLE DESIGNS

After the preprocessing steps, we obtain a set of cages for each design; see again Fig. 4. Next, we need to form correspondences between cages, i.e., cages of exact dimensions but in different designs, so that we later can construct a part that can be reused at the corresponded cage locations in different designs.

Cage Similarity. To do so, we first need to measure the dimension similarity between cages. Since cages can be corresponded in different orientations, we align their centroids in 3D and compute the similarity between cages, say B_1 and B_2, as

$$\mathbb{S}(B_1, B_2) \;=\; \max_i \frac{\text{vol}(\, B_1 \cap R_i(B_2)\,)}{\text{vol}(\, B_1 \cup R_i(B_2)\,)}\,, \tag{1}$$

where vol$(\cdot)$ is the volume operator; and $\{R_i\}$ are the six possible axis-to-axis orientations due to symmetry. Note that $\mathbb{S}$ ranges $(0, 1]$, and $\mathbb{S}$=1 indicates a perfect (exact) match.



Modification Operations. To form a correspondence between two cages, the two cages must have exactly the same dimensions. However, this may not be the case for the initial cages produced from the preprocessing steps. Hence, we may need to modify some of the cages to achieve the exact matching. Yet, we should minimize the amount of modification to keep the functionality and appearance of the designs. Hence, we consider the following two operations to modify cages, while respecting the geometric semantics constraints:

i) *Resize.* Given two cages, we can resize them to have the same dimensions to form a correspondence. In the implementation, we restrict the relative volume change of each cage to be no larger than α, which is empirically set as 20% to tradeoff between forming the correspondence and preserving the appearance of the input designs.

ii) *Split.* For large cages, e.g., the one marked by the red arrow in Fig. 4, if we resize it to match some other cages, it will require a large volume change, which is undesirable. Hence, we can split a large cage into two smaller cages, e.g., the green and pink cages shown at the bottom of Fig. 3(b), and form a correspondence for each of the smaller cages, possibly with a moderate resizing.

Problem Formulation. Given N input designs, each approximated by a set of cages $\{\mathbf{S}_i\}$, where $i = 1, ..., N$ with $N \geq 2$, we formulate co-decomposition as an edge construction problem on dynamic bipartite graphs. For each pair of designs, say with $\mathbf{S}_i$ and $\mathbf{S}_j$ $(i \neq j)$, we create a dynamic bipartite graph, where each node denotes a cage in $\mathbf{S}_i \cup \mathbf{S}_j$, and each edge denotes a correspondence between a cage in $\mathbf{S}_i$ and a cage in $\mathbf{S}_j$, such that their similarity $\mathbb{S}$ is one.

To guide the iterative method (see the next paragraph) to modify cages, form correspondences, and construct edges in the bipartite graphs, we formulate the following objective function:

$$\max \sum_{i,j \;s.t.\; i\neq j} \Big[\; \omega_1 C_1(\mathbf{S}_i, \mathbf{S}_j) \;-\; \omega_2 C_2(\mathbf{S}_i, \mathbf{S}_j) \;\Big] \;+\; \sum_i \omega_3 C_3(\mathbf{S}_i)\,, \tag{2}$$

$$\begin{aligned} \text{with} \quad C_1(\mathbf{S}_i, \mathbf{S}_j) &= |\mathbf{E}_{ij}|\,, \\ C_2(\mathbf{S}_i, \mathbf{S}_j) &= |\mathbf{S}_i| \;+\; |\mathbf{S}_j| \;-\; |\mathbf{E}_{ij}|\,, \\ \text{and} \quad C_3(\mathbf{S}_i) &= \sum_k \frac{\text{vol}(B_k^i \cap \bar{B}_k^i)}{\text{vol}(B_k^i \cup \bar{B}_k^i)}\,, \end{aligned}$$

where $\mathbf{E}_{ij}$ is the edge set of the bipartite graph created for $\mathbf{S}_i$ and $\mathbf{S}_j$, so $|\mathbf{S}_i|+|\mathbf{S}_j|-|\mathbf{E}_{ij}|$ means the number of actual (common) parts to be constructed and fabricated; B_k^i is the k-th cage in $\mathbf{S}_i$; $\bar{B}_k^i$ is the initial B_k^i; and ω_1, ω_2, and ω_3 are weights, which are empirically set as 1, 0.5, and 0.3, respectively, in all the experiments. In summary, the first term aims to maximize the parts reuse, the second term aims for a smaller common set of parts, while the third term aims to reduce the amount of resizing on the cages.

Iterative Method. The way we construct edges in bipartite graphs differs from the well-known problems "*Maximum Bipartite Matching*" and "*Minimum Weight Perfect Matching*" [Lovász and Plummer 2009], since we maximize the number of correspondences while considering other factors. Moreover, our bipartite graph is dynamic, where a node may split, and we aim for a small graph that leads to few parts and simple assemblies. These unique features make the problem very challenging, where existing bipartite matching techniques [Lovász and Plummer 2009] are insufficient to solve. Hence, we approach the problem using an iterative method with the following steps to maximize the objective function:

Step 1: Align the designs. For a given pair of designs, we first score each pair of cages between them using Eq. 1, and find the candidate pairs whose score $> \sigma$ (typically in $[0.8, 0.95]$); see the dashed lines in Fig. 6(a). Then, we randomly pick a pair with higher probability for those with larger similarity, and form a correspondence between the pair, possibly after a moderate resizing. Next, since there are four possible axis-to-axis rotations to align two cages (which are rectangular boxes) in 3D space, we translate the two designs to align the centroids of the two cages, try all four possible rotations

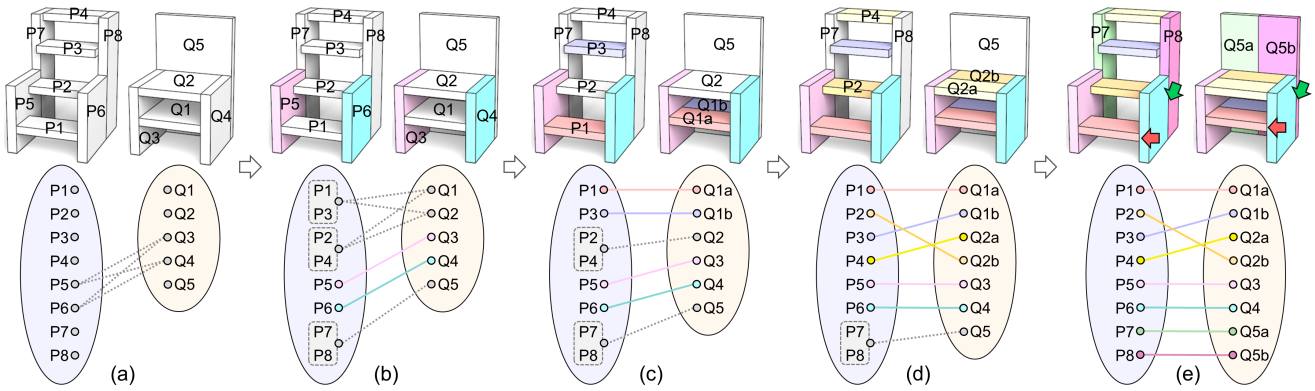


Fig. 6. Iterative co-decomposing cages in Step Ladder and Chair. In the associated dynamic bipartite graph shown at the bottom, dashed edges denote candidate correspondences while colored edges denote established correspondences. (a) Input cages in the two designs; (b) two correspondences established; (c-e) we iteratively split cages Q_1, Q_2, and Q_5, and obtain the co-decomposition result shown in (e).

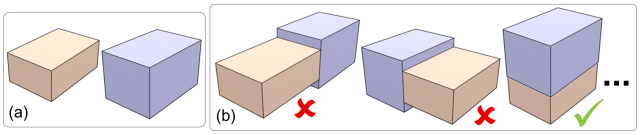


Fig. 7. Given two cages (a), we try all 3×3×2 different ways of merging the two cages by finding compatible faces of similar dimensions (b).

about the aligned centroids, and find the rotation that minimizes the sum of minimum centroid-to-centroid (L_2) distances from each cage in a design to all cages in the other design. Then, we can obtain a transformation that aligns the two cages and the two designs. After that, we identify nearby cage pairs between the two designs (within 10% of the maximum dimension of the two designs), and increase the scores of these pairs by β (set as 50% in all experiments). Since these cages have consistent relative locations from the first corresponded cage pair, forming correspondences for these pairs can facilitate joint and interlocking construction later.

Step 2: Locate cliques among cage pairs. In case of more than two input designs, we look for cliques. For each candidate cage pair P_a-Q_b, if they have a common partner (say R) in their correspondences with cages in some other designs, we increase the score of P_a-Q_b by γ% (set as 5 in all experiments) times the sum of scores of P_a-R and Q_b-R. Then, we sort all candidate cage pairs by their scores.

Step 3: Form correspondences. Next, we randomly pick a candidate pair with preference on those with high scores, and attempt to form a correspondence between them by resizing the cages with minimum volume changes. In case we resize a cage in a symmetry group or in a clique, other cages in the group should also be resized accordingly, e.g., P_5 and P_6 in Fig. 6(b). Moreover, we discard candidate pairs whose "resize" violate the maximum allowed volume change, and we may slightly translate adjacent cages to avoid cage collision and to enforce the contact and alignment constraints in a design. We iterate step 3 until we cannot form more correspondences, during which the corresponded cages should not be picked again. Note that when a correspondence is formed, $|\mathbf{E}_{ij}|$ increases by one, so the value of the objective function (Eq. 2) should improve by "$\omega_1 + \omega_2 - \omega_3 \Delta C_3$," where ΔC_3 is the sum of relative volume change in the two associated cages based on C_3's formulation in Eq. 2.

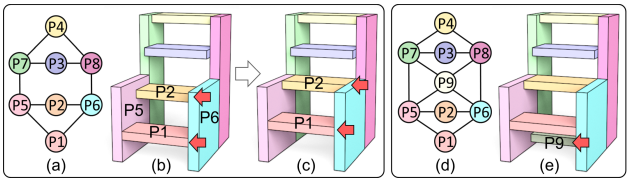


Fig. 8. Post-processing the co-decomposed cages in Step Ladder: (b&c) show how we adjust the location of cages P_1 and P_2 (red arrows) to merge and reuse the joint connections with cages P_5 and P_6; see the associated parts-graph in (a); and (e) shows the extra part P9 (red arrow) that facilitates LIG and interlocking constructions; see the associated parts-graph in (d).

Step 4: Split to form correspondences. Other than resizing, we can form cage-to-cage correspondences by splitting cages. To do so, we take an inverse approach by finding cages (in each design) that can be merged into a larger cage (in other designs) with compatible contact faces (see Fig. 7) by using a 2D version of Eq. 1 to measure the face-to-face similarity. Then, for each merged cage, we use Eq. 1 to check if it is similar to any cage (a split candidate) in the other designs, e.g., the grey frame $[P_1, P_3]$ in the bipartite graph shown in Fig. 6(b). Similar to steps 1 and 3, we sort the split candidates by their similarity scores, iteratively pick a split candidate at random, and then follow step 3 to create correspondences between each split cage and its partner; see Fig. 6(b&c) for an example, where Q_1 is split into Q_{1a} and Q_{1b}, which correspond to P_1 and P_3, respectively.

Note. After steps 1 to 4, the input designs are decomposed into cages with correspondences; see Fig. 6(e). However, due to the complexity in constructing joints with interlocking, not all decomposition results can lead to compatible joints and mechanical interlocking for every design. Hence, we perform steps 1 to 4 multiple times with randomness, and pick the best K decomposition results that maximize the objective function (Eq. 2); typically, we set K as 10.

Post-processing. For the purpose of simplifying and reusing joints and facilitating interlocking construction, we have the following two post-processing steps to modify the co-decomposed cages:

Simplify connections. On each common part to be created (for each group of corresponded cages), we generally need one joint to

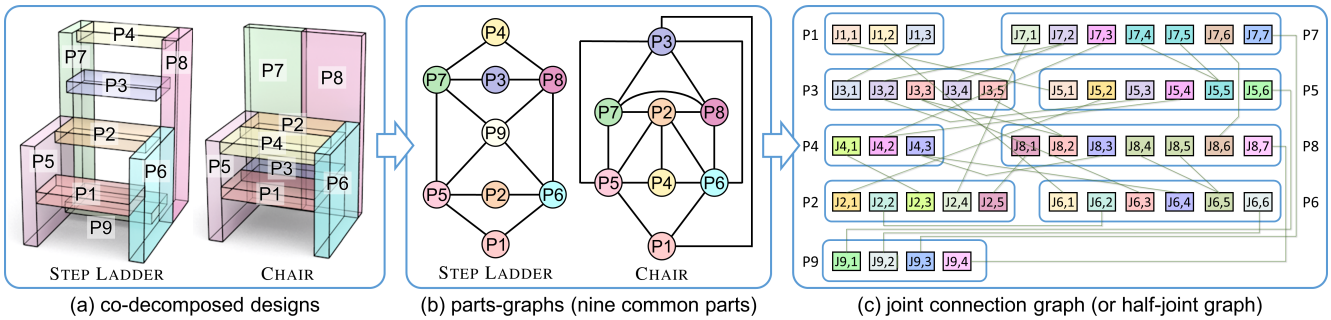


Fig. 9. For each co-decomposed design (a), we create a parts-graph (b) to model the joint connections between its parts; note that P_1, ..., P_9 above refer to common part IDs, which may differ from the part IDs in individual designs before the co-decomposition. Next, we summarize the half-joint connections over the designs to create the half-joint graph (c) to model the joint connections (search space) in the reconfigurable assemblies.

connect it with each of its neighbors for each design. As an example, the common part to be created for cages P_6 and Q_4 (cyan) in Fig. 6(e) should have eight connections: three for P6-P1, P6-P2, and P6-P8 in STEP LADDER plus five for Q4-Q1a, Q4-Q1b, Q4-Q2a, Q4-Q2b, and Q4-Q5b in CHAIR. Observing that some of these connections are at the same, or nearly the same, locations on the common part (see the green and red arrows, respectively, in Fig. 6(e) for examples), we can reduce the number of actual connections (or joint geometry) to be constructed on the common part by merging joint connections at the same location. For joint connections at nearby locations, we can translate the associated cage slightly; see the arrows in Fig. 8(b&c) for examples. In this way, only five connections remain on the common part to be constructed for P_6 and Q_4, i.e., Q4-Q1a (≡P6-P1), Q4-Q1b, Q4-Q2a, Q4-Q2b (≡P6-P2), and Q4-Q5b (≡P6-P8).

Facilitate interlocking. To interlock the assembly of each design, we follow [Fu et al. 2015] to form 3- and 4-part local interlocking groups as described in Section 3. However, if a joint connection (i.e., an edge in the parts-graph) is not included in any 3- and 4-part cycle, the two associated parts may not connect well. Hence, our tool can automatically identify such connections (e.g., P5-P7 and P6-P8 in STEP LADDER; see Fig. 8(a)), and give suggestions for manual introduction of a new node (e.g., see P9 in Fig. 8(d&e)) to form extra 3- or 4-part cycles that include the joint connections.

5 MODELING RECONFIGURABLE ASSEMBLIES

Half joints. Obviously, parts-graphs (see Fig. 9(b)) can only model the joint connections in individual designs; they are insufficient to model the joint connections across multiple designs in reconfigurable assemblies. Hence, we consider *half joint*, which is the portion of a joint on an individual part. Taking the half joint circled on the yellow part in Fig. 3(e) as an example, it can connect to the green part in STEP LADDER or the light pink part in CHAIR; see Fig. 3(d). Thanks to the post-processing step, which simplifies the joint connections, the total number of connection locations (i.e., half joints) is minimized on each common part. As a notation, we denote $J_{i,j}$ as the j-th half joint on the i-th common part P_i.

By summarizing the half joint connections between common parts over all parts-graphs (see Fig. 9(b)), we construct the *joint connection graph* (or *half-joint graph*) to model the joint connections in reconfigurable assemblies (see Fig. 9(c)). In this graph, each node is a half joint and each edge is a half-joint connection, so a node may connect to multiple other nodes. For assembly stability, we employ only the joint geometries that constrain parts to separate in a single direction, e.g., halved joints, mortise-and-tenon, and dovetails. Hence, each half joint associates with a free variable (named $d_{i,j}$ for $J_{i,j}$), i.e., the free-axial direction (defined in P_i's object space) to remove P_i from its partner at the half joint. For simplicity, we consider only the six principal axial directions in this work.

Formulating the Solution. The search space of reconfigurable interlocking assemblies has two portions. One is the set of free variables $\{d_{i,j}\}$ associated with the half joints. The other is the set of transformations (denoted as $T_{k,i}$) on rotating each common part (P_i) for use in each (k-th) design. Hence, P_i's removal direction at $J_{i,j}$ in the k-th design is $T_{k,i} \cdot d_{i,j}$, and if half joints $J_{a,x}$ and $J_{b,y}$ connect parts P_a and P_b in the k-th design, we should have

$$T_{k,a} \cdot d_{a,x} = -T_{k,b} \cdot d_{b,y} \,, \tag{3}$$

since P_a and P_b should move in opposite directions from each other in the k-th design. After the co-decomposition, we only have a rough orientation of each common part in a design. Taking P_1 in STEP LADDER (left) in Fig. 9(a) as an example, if we rotate its cage 180 degrees about one of the three principal axes, we still can fit it with its neighbors (P_5 and P_6), since it does not have joint geometry yet. Thus, there are generally four choices of $T_{k,i}$ for posing each common part in a design. Therefore, we can formulate the problem of designing reconfigurable interlocking assemblies as a problem of finding $\{d_{i,j}\}$ and $\{T_{k,i}\}$ that fulfill the following five constraints:

- Joint compatibility: a half joint may connect to more than one half joints; for example, $J_{3,2}$ may connect to $J_{7,2}$ or $J_{5,3}$; see Fig. 9(c). Hence, its geometry should be compatible for both connections. Thanks to the half-joint graph, we can form the joint compatibility constraint using Eq. 3 for each edge in the graph.
- Joint geometry: for each half-joint connection, we employ the joint analysis technique in [Fu et al. 2015] to find the possible joint geometries (halved joints, mortise-and-tenon, and dovetails) that can be deployed at the connection. Hence, for each half joint on each common part, we can find a set of $d_{i,j}$ for each half-joint connection that it involves, and then determine the

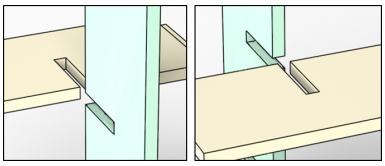


Fig. 10. This connection allows only two joint geometry candidates.

common $\{d_{i,j}\}$ as its candidates. Note that we may not allow all six axial directions as candidates; it depends on the local geometric arrangement between the associated parts. In the example shown in Fig. 10, there are only two possible choices of $d_{i,j}$, since we can only employ halved joints to connect the two parts.

- Interlocking: next, we need to choose appropriate $\{d_{i,j}\}$ to form local interlocking groups and plan a network of joints to interlock the assembly of each design; see Section 6.3, step (iii).
- Functional stability: we may set a constraint to forbid a specific part to move in a certain direction; for example, $d_{i,j}$ of Ladder's horizontal parts should not go downward or inward; see Fig. 1.
- Orientation association: in general, there are four choices for each $T_{k,i}$. However, not all choices are effective. Taking common part P_5 in Fig. 9(a) as an example, if we use P_5 in different orientations in the two designs, say by rotating it 180^o about the vertical axis, we need four half joints to connect it in Step Ladder (left) and five in Chair (right). Since these four and five half joints locate on opposite sides of P_5, we cannot combine them; hence, we will need to create an excessive number of half joints on P_5. Therefore, if we impose a constraint on the associated choices of $T_{k,i}$ for P_5 in the two designs, e.g, same orientation for P_5 in both designs, we only need six half joints on P_5; see again Fig. 9(a). In practice, this is a soft rather than hard constraint compared to the above four constraints, and we initialize all $T_{k,i}$'s during the post-processing step when simplifying the joint connections.

The Search Space. If we choose a certain $d_{i,j}$ for $J_{i,j}$, half joints connected *directly and indirectly* to $J_{i,j}$ in the half-joint graph will be constrained to have certain $d_{i,j}$ subject to the chosen $T_{k,i}$'s due to the joint compatibility constraint. So, the actual number of free choices for $\{d_{i,j}\}$ depends on the number of disjoint subgraphs (denoted as g) in the half-joint graph. For examples, there are 19 and 7 subgraphs in Figs. 9(c) and 11(c), respectively; we use the same color for nodes of the same subgraph. Note that if we change $T_{k,i}$, we have to update the portion of the half-joint graph related to P_i by rebuilding and reconnecting the nodes associated to P_i.

If m is the average number of candidate directions for $d_{i,j}$ given by the joint geometry constraint ($m = 4$ on average), l is the average number of choices for $T_{k,i}$, and n is the number of common parts, we roughly have $m^g l^n$ combined choices in total. This is a huge space, in which we seek solutions ($\{d_{i,j}\}$ and $\{T_{k,i}\}$) that achieve interlocking rather than deadlocking[1] for each design. Hence, we develop novel interlocking models and the co-construction method to efficiently explore the search space by iteratively planning joints on the common parts simultaneously over multiple designs.

[1]Deadlocking refers to the situation that a structure cannot be assembled from its component parts due to physical parts blockage during the assembly process.

6 CO-CONSTRUCTING COMPATIBLE JOINTS

6.1 Backward Interlocking Model

Existing works, e.g., [Fu et al. 2015; Song et al. 2012], create interlocking assemblies by iteratively constructing local interlocking groups with dependency; see Section 3 for description on LIG. Dependency means that when we successively construct each LIG (except the first), we purposely plan its joints, such that its local key is immobilized by some previous LIG(s). Hence, every part in the second LIG is immobilized by the first LIG, every part in the third LIG is immobilized by the first and/or the second LIG, etc., except the first LIG, whose local key becomes the only mobile part (global key) in the entire assembly. We call this the *forward interlocking* model, since the LIG construction starts from the global key.

Forward interlocking assumes a single global key. It is not suitable for reconfigurable interlocking assemblies. Since different designs have different structures, it is unlikely that we can always start with a *common* global key and a *common* first LIG that can immobilize all subsequent LIGs in every single design. The planning of joints is extremely restrictive given the joint compatibility constraint.

Interestingly, we found that it is possible to reverse the LIG construction order when creating interlocking assemblies. In this work, we present the *backward interlocking* model, by which we construct LIGs in reversed dependency order. It means that when we construct each LIG (except the first), we purposely plan its local joints to immobilize the local key(s) of some previous LIG(s); in other words, a new LIG is not immobilized by any previous LIG. Hence, backward interlocking ends the LIG construction at the global key.

6.2 Multi-key Interlocking Model

Single-key interlocking is hard to achieve for reconfigurable interlocking assemblies, since the assembly of each design should be locked by only one specific key with joint compatibility across multiple designs. To relax the constraint for constructing reconfigurable interlocking assemblies, we present the *multi-key interlocking* model, which generalizes the conventional single-key model:

> *In a multi-key interlocking assembly, only a few specific keys are mobile; all parts and all subsets of parts are immobilized unless we first take out some of these keys.*

Multi-key interlocking and backward interlocking together facilitate LIG construction for reconfigurable assemblies. When we iteratively construct LIGs, backward interlocking allows us to first explore (cyclic) substructures common to all designs, without requiring the local key of the first LIG to lock all subsequent LIGs. When there are no more substructures common to all designs, we can explore substructures in a subset of designs and then individual designs by branching out the exploration (see Figs. 3(c) and 11(d)) and planning joints with *multiple global keys* for each individual design.

6.3 Co-Construction Method

Our co-construction method plans joints in reconfigurable interlocking assemblies based on the backward interlocking and multi-key interlocking models. Altogether, it has the following steps:

i) Find common substructures. Thanks to the post-processing step that facilitates interlocking, we have identified (and possibly

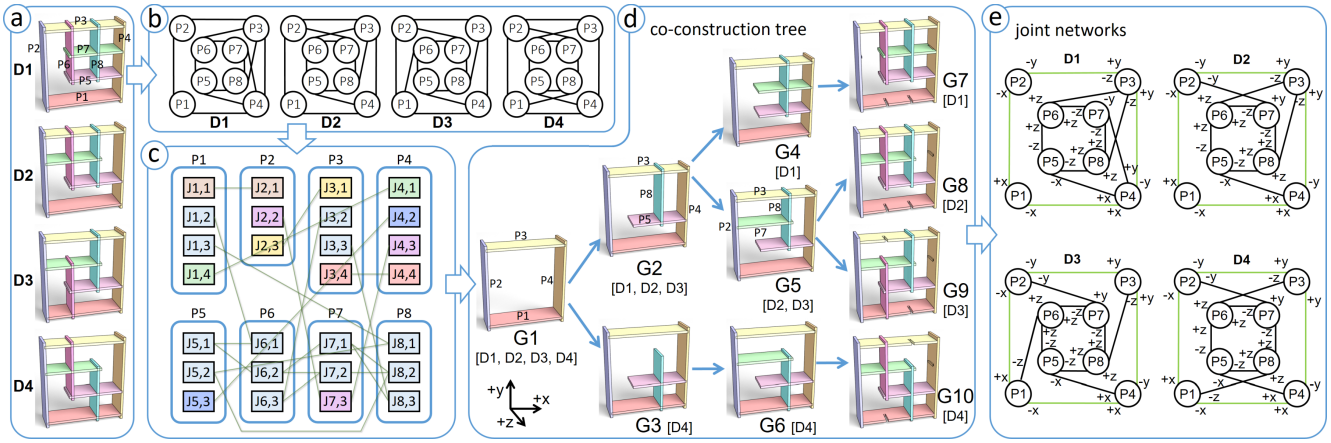


Fig. 11. The co-construction method. Given a set of co-decomposed designs (a), it first creates parts graphs (b) and a half-joint graph (c) to model the joint connections, and identifies cyclic substructures common to multiple designs. Then, it aggregates common substructures into a co-construction tree (d), and iteratively plans joints (e) on the substructures over multiple designs simultaneously. Note that in (e), the direction labels ($\pm x$, $\pm y$, and $\pm z$) next to the nodes (parts) along each edge (joint) indicate the free principal axial directions (the assigned $d_{i,j}$) that the associated joint imposes on the parts.

introduced) cyclic substructures in each decomposed design. Owing that cyclic substructures of three or four parts are candidates for forming LIGs, we start by finding cyclic substructures common in all or multiple designs, e.g., see Fig. 11(a,b&c): $[P_1, P_2, P_3, P_4]$ is common in all designs, while $[P_3, P_4, P_5, P_8]$ is common in D_1, D_2 and D_3. Moreover, we have to examine whether the half-joint connections in substructures across different designs are consistent. For example, $[P_5, P_6, P_7, P_8]$ appears to be a common substructure in all the four parts-graphs in Fig. 11(b), but it is actually not, since the half-joint connections involve in the four designs are in fact different.

ii) Create a co-construction tree. Next, we aggregate common substructures into a *co-construction tree*, where each node is associated with a cyclic substructure to be constructed into an LIG. To build the tree, we first form internal nodes that correspond to substructures in all or multiple designs, and then form nodes for the substructures in individual designs; see Fig. 11(d). Moreover, we aim to maximize the grouping of common substructures in the tree to reduce the number of LIGs needed to be constructed over the assemblies. In our current implementation, we greedily aggregate first the substructures common to more designs, but this can also be done with an optimization. Lastly, we may not always have a single root node for all the designs; if this is the case, we create multiple co-construction trees.

iii) Enumerate various constraints. Next, we visit each node and edge in the co-construction tree to enumerate the interlocking constraints: i) each node (substructure) should form an LIG with a local key; and ii) along each edge, successive LIGs are preferred to have dependency in backward interlocking style, where parent LIG depends on child LIG. Moreover, we take into account the associated choices of $T_{k,i}$ (for the orientation association constraint) and choices of $d_{i,j}$ (for the joint geometry and functional stability constraints), and then identify disjoint subgraphs in the half-joint graph to find groups of $d_{i,j}$ (for the joint compatibility constraint).

iv) Joint planning. Lastly, we traverse the co-construction tree from the root in breadth-first order to plan joints. For the root node, we plan the local joints in its associated substructure, so that the substructure becomes an LIG, e.g., G_1 in Fig. 11(d). Then, to fulfill the joint compatibility constraint, for each half joint in the LIG, we examine the subgraph it belongs to in the half-joint graph, and assign appropriate $d_{i,j}$'s to the half joints of the other nodes in the subgraph according to Eq. (3). For each successive node in the co-construction tree, we first find the free $d_{i,j}$'s remained in the associated substructure, i.e., $d_{i,j}$'s that are still unassigned after we used Eq. (3) to assign $d_{i,j}$'s for the previous nodes. Then, we try to plan joints to make the substructure into an LIG by assigning appropriate $d_{i,j}$'s to the free variables.

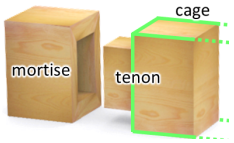
We iterate the above process until we traverse all the nodes and successfully form an LIG for every single node in the co-construction tree. In this way, we can obtain a feasible solution, i.e., a network of joints, for each design; see Fig. 11(e). Lastly, for each half joint, we lookup the associated joint geometry based on the assigned $d_{i,j}$, and adopt the joint geometry onto the associated common part by constructive solid geometry operations. As a result, we can produce a common set of parts for the reconfigurable assemblies; see Fig. 3(e).

In practice, due to the joint compatibility and the interlocking constraints, we often need backtracking when traversing the tree in case we cannot form an LIG for the substructure being visited. Moreover, we *may* also need to break the orientation association constraint to relax the joint compatibility constraint to facilitate the LIG constructions by having more half joints on the associated common parts. However, it is worth to note that when planning the joints, our method has made full use of the various constraints in an organized manner to optimize our efficiency in exploring the search space. If we simply use randomized or exhaustive methods to plan the joints without considering LIG and joint compatibility, the procedure is typically intractable, even for simple assemblies; see the baseline comparison in Section 7.

Fig. 12. Results. Reconfigurable interlocking furniture generated by our method (from left to right and then top to bottom): Bookshelf, Table, and two Chair; Bed, Cot, and Desk; Armoire (extensible and hierarchical); Bookshelf 1-4; Hand Truck, Stool, and Ladder; Shoe Rack and Laundry Box; Chair and Step Ladder; and Office Box 1-3.

7 IMPLEMENTATION AND RESULTS

Implementation. Due to the complexity in the co-decomposition and co-construction processes, especially for fulfilling the joint compatibility and interlocking constraints while maximizing the part reuse, each process actually generates a couple of results with randomness; see Section 4. Moreover, we may discard undesirable results for the following cases. First, when adopting joint geometry on a common part, certain portion of the joint geometry may bulge out of the cage associated with the part, e.g., the tenon portion in the inset figure. Since a common part could be reused with different poses in different designs and a half joint on it may not be in use in every design, an unused tenon that points downward could distract the part's contact on the ground, e.g., a chair leg. If this issue happens, we first try to resolve it by swapping the role of mortise and tenon in the associated joint geometry; otherwise, we have to discard the result. In addition, we may discard a result, if its assembly structure is not stable, or it reveals excessive unused half joints on the outer and front sides, which hurt the aesthetics of a design. Currently, these undesirable cases are detected manually.



After adopting the joint geometry on the common parts, we attempt to further take the local features in the input designs to the associated common parts, if the features are compatible (by manual inspection) in all associated designs. Taking Chair in Fig. 3(a) as an example, we may take the relief pattern on its back plane, since having this feature on the green and pink common parts is compatible for all the designs. However, we cannot take the two round corners on top of Chair, otherwise they will appear at the bottom back sides of the green and pink parts in Step Ladder, so the ladder may fall when in use. Another example is the wheel set on Hand truck; see Fig. 1. These wheels are compatible in Ladder and Stool, and they locate in regions without half-joint geometry for all the designs; such regions are identified automatically.

Table 1. Statistics of results produced by our method; the labels in 3rd to 9th columns refer to the number of parts, number of joints, number of keys, total number of common parts, total number of half joints, time for co-decomposition (T-co-D), and time for co-construction (T-co-C).

Fig.	Forms (designs)	#P	#J	#K	#C	#H	T-co-D. (sec.)	T-co-C. (sec.)
1, 4	Ladder	9	14	2	11	50	24.8	35.6
	Stool	11	16	3				
	Hand Truck	10	14	2				
3, 4, 6, 8, 9	Step Ladder	9	14	2	9	46	0.4	0.7
	Chair	8	17	2				
11, 17	Bookshelf 1	8	12	2	8	27	2.7	1.1
	Bookshelf 2	8	12	2				
	Bookshelf 3	8	12	2				
	Bookshelf 4	8	12	2				
13a	Bed	8	13	1	11	44	0.6	0.5
	Cot	5	8	1				
	Desk	6	8	1				
13b	Bookshelf	15	22	1	16	70	275.0	493.8
	Table	6	9	1				
	Chair	5	8	1				
	Chair	5	8	1				
16,18	Office Box 1	9	19	1	11	64	38.0	272.2
	Office Box 2	10	21	1				
	Office Box 3	10	20	1				
19	Shoe Rack	9	16	2	12	60	0.5	2.3
	Laundry Box	9	16	2				

Results and Performance. We employed our method to create reconfigurable interlocking furniture of various functionalities; see Fig. 12. Table 1 presents their statistics. Note that some parts in the common set of parts may not be in use for all the designs. Taking

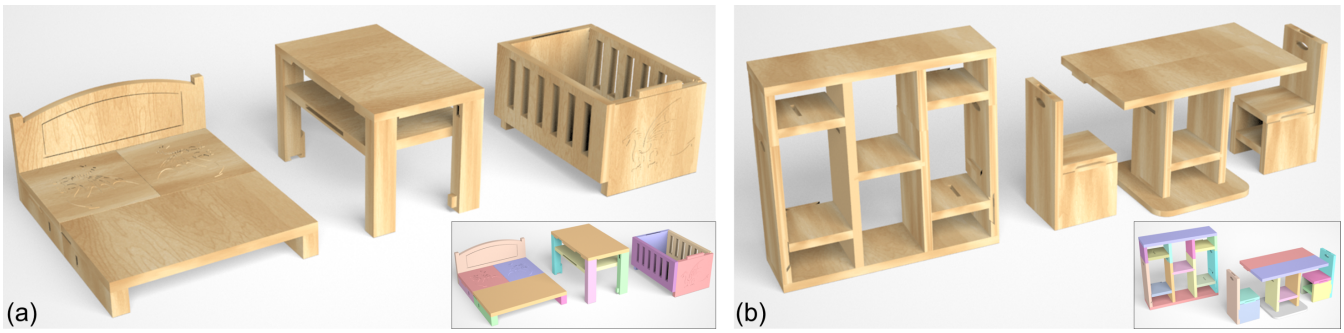


Fig. 13. Reconfigurable Bed-Cot-Desk (a) and Bookshelf-Table-Chairs (b); the colors on parts reveal the part correspondence.

Table 2. Performance of the two baseline methods (B1 and B2).

	method	# total trials	# non-compatiable	# non-interlocking	# non-assemblable	# valid solutions	time (min.)
Bookshelves	B1	10^7	10^7	0	0	0	158.2
	B2	10^7	0	7256501	2028470	715029	159.8
Step Ladder -Chair	B1	10^7	10^7	0	0	0	181.1
	B2	10^7	0	6834478	3165463	59	192.0
Bed-Cot-Desk	B1	10^7	10^7	0	0	0	91.8
	B2	10^7	0	9977899	22075	26	93.5
Ladder-Stool-Hand Truck	B1	10^7	10^7	0	0	0	675.3
	B2	10^7	0	8845052	1154943	5	690.4
Bookshelf-Table-Chairs	B1	10^7	10^7	0	0	0	768.5
	B2	10^7	0	9975577	24423	0	783.5

Ladder-Stool-Truck as an example, it has eleven common parts altogether: nine of the parts are used in Ladder, ten in Hand Truck, and all in Stool. Moreover, we intentionally specify the functional stability constraints on the keys, so that they cannot move out in the same direction as gravity or as any external force during casual usage, e.g., the horizontal keys in Ladder against stepping and the top key on Stool against sitting; see Fig. 1. Lastly, for Bed in Fig. 13(a), we need to manually split the large planar cage into two, so that the top half can be split by the co-decomposition process, since our current implementation does not recursively split and match parts.

We implement our method in C++ and execute it on a desktop PC with a 3.4GHz CPU and 8GB memory. The last two columns in Table 1 show the time performance of the whole co-decomposition and co-construction processes. In general, the time performance depends on the similarity between the local shapes in the input designs, the structure of the designs, the number of common parts, and the amount of reuse in the parts and joints across the designs (in other words, the complexity of the half-joint graph).

Evaluation. To evaluate our method's performance on planning joints in reconfigurable interlocking furniture (which is a very complex problem compared to previous works), we implement two baseline methods (B1 and B2) that randomly assign values to $\{d_{i,j}\}$ with $\{T_{i,k}\}$ initialized based on the orientation association constraints; see Section 5. If we randomly assign $\{T_{i,k}\}$ as well, the search space will be too large for the baseline methods to find any solution.

B1 randomly assigns $\{d_{i,j}\}$ from the choices given by the joint geometry constraint, and checks if each result i) satisfies the joint compatibility constraint, ii) can be assembled without deadlocking, and iii) is multi-key interlocking with at most three global keys in each design. Since no solutions can be found by B1, we introduce B2 that randomly assigns $\{d_{i,j}\}$ using the joint compatibility constraint, i.e., randomizing one $\{d_{i,j}\}$ per subgraph in the half-joint graph.

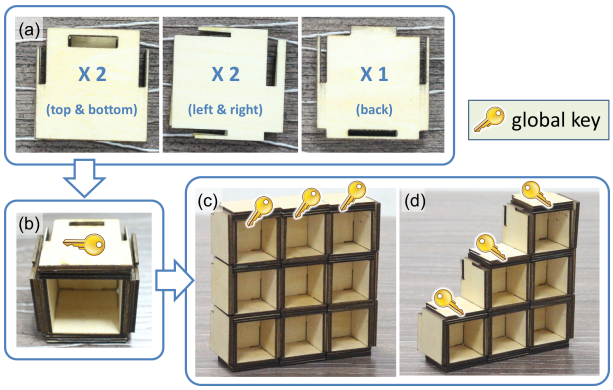


Fig. 14. Reconfigurable interlocking cupboards composed of tileable part instances with recursive connections. The key icons mark the global keys.

Table 2 shows the results. For simple inputs with a small number of common parts and half joints, e.g., Bookshelves, B2 may find solutions. For complex inputs, the search space becomes intractable for B2 to find solutions, even though we have already considered the orientation association and joint compatibility information in B2. Thanks to the half-joint graph and the backward interlocking model, we can efficiently explore the search space and plan joints with our co-construction method; see again the last column in Table 1.

Extensible and hierarchical assemblies. We further extend our computational framework by considering half-joint graphs with recursive connections. It means that a node in the half-joint graph can connect to nodes of the same part; in other words, a half joint can connect to half joints of different instances of the same part. In this way, we can form tileable parts and tileable substructures, and employ them to design and construct reconfigurable interlocking assemblies that are extensible and hierarchical. Fig. 14(a,b) show three unique parts designed with this idea; their instances can form a basic unit, which is an interlocking box assembly with a single key on top; see Fig. 15 (left). This basic unit can form extensible interlocking cupboards with a few keys; see Fig. 14(c,d).

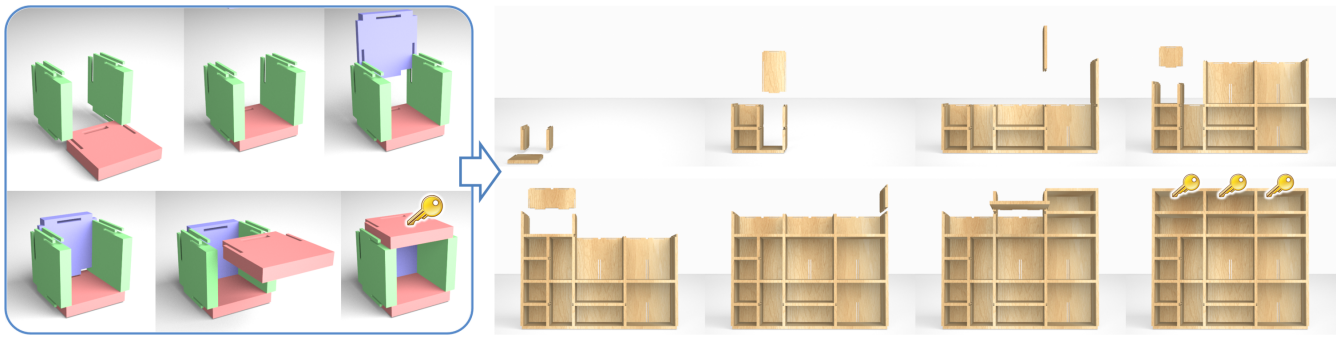


Fig. 15. Left: a single-key interlocking box assembled from three unique parts. Right: Armoire composed of tileable boxes and three global keys on top.

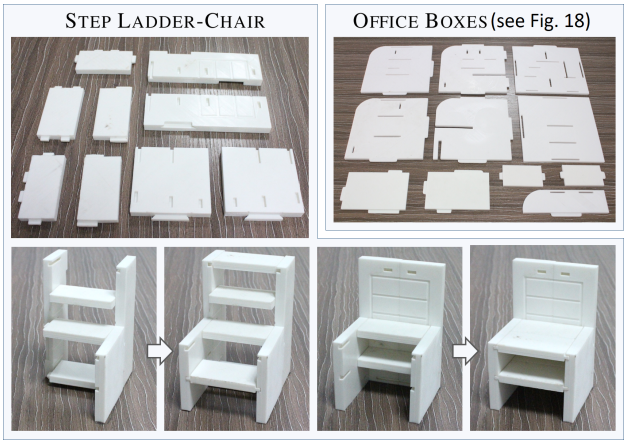


Fig. 16. 3D printing results: Step Ladder-Chair (nine common parts) and Office Boxes (eleven common parts; see Fig. 18 for the assemblies).

By further expanding the dimensions of the basic parts horizontally and/or vertically, we can recursively connect the part instances in a hierarchy, and produce complex but yet reconfigurable assemblies such as the storage combination "Armoire" shown in Fig. 15 (right). This is an extensible, hierarchical and interlocking structure with 65 parts but only three global keys and seven unique parts.

3D Fabrications. We create several physical models of our results using three different types of fabrication.

- Laser cutting. We use 2D laser cutting to produce planar parts of 1.5mm thickness, and stack four layers of laser-cut parts to create each unique part shown in Fig. 14. Since there are only three unique parts, we only need to prepare three laser-cut contours, which can be efficiently reused to produce different designs.
- 3D Printing. We fabricate Step Ladder-Chair (0.2m high) and Office Boxes (0.3m high) using FDM 3D printers and PLA plastic material; see Figs. 16 and 18. The working volume of the two printers are $0.23\times0.23\times0.20m^3$ and $0.5\times0.4\times0.4m^3$. The time taken to print the common set of parts for Step Ladder-Chair and Office Boxes are 78.2 and 69.0 hours, respectively, where we print the smaller model, i.e., Step Ladder-Chair, in higher fidelity, so it took longer printing time. In the assemblies, the parts can tightly interlock with one another; see the supplementary video.

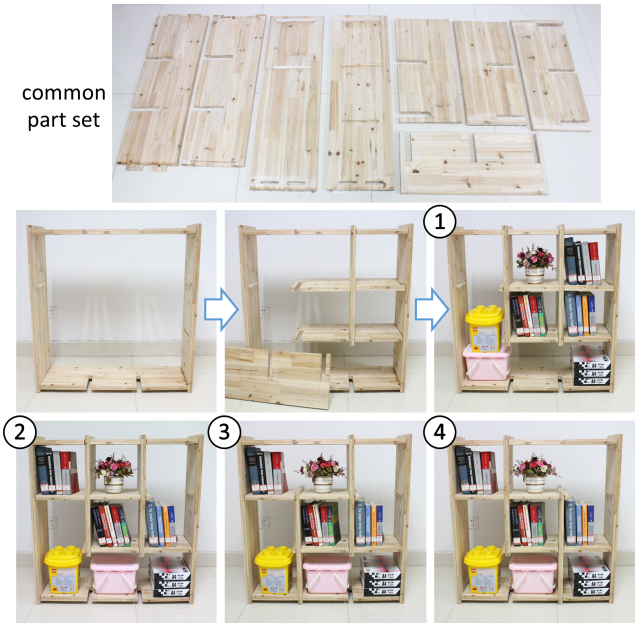


Fig. 17. Bookshelves fabricated by woodworking; the common set of parts can be assembled into four different interlocked forms.

- Woodworking. We fabricate two sets of results in real physical size using woodworking: Ladder-Stool-Truck (see Fig. 1(right)) and Bookshelves (see Fig. 17), which are 1.6m and 1m high, respectively. The information that we passed to the carpenter include: i) the dimension of each common part; ii) the type, size, and location of each half joint on the parts; and iii) the wheel sets on Ladder-Stool-Truck. After that, it took the carpenter two days to make Ladder-Stool-Truck and one day to make Bookshelves, since Ladder-Stool-Truck has a large number of half joints that require more time to create. Due to the weight and size of the assembly result (i.e., Ladder and Truck), we found that we need to put in pins that get through the parts at the three-way joints to strengthen the connections. By then, the construction would be strong enough to support larger external forces applied on the assemblies; e.g., the student who climbed the ladder in Fig. 1, and many objects put on the bookshelves in Fig. 17. Please watch the supplementary video for details.

Fig. 18. Reconfigurable Office Boxes, which has a common set of eleven parts and three different assembly forms.

Limitations. First, the co-decomposition process may not be able to construct cages with correspondences if the local shapes in the given designs have very different dimensions (compare Fig. 19(a) and (b)) or very different substructures (compare Fig. 19(b) and (c)). As such, we may not be able to create reusable common parts. Second, the current interlocking model relies on cyclic substructures; it requires the user to put in extra part(s) in case there are insufficient cyclic substructures; see Fig. 19(a). Moreover, it considers only orthogonal joint connections. Third, our current implementation requires manual works in some of the steps; e.g., add extra parts for cyclic substructures, identify undesirable results, and create compatible local geometric features. Fourth, the co-decomposition and co-construction processes are performed sequentially, where failure in the co-construction does not feedback to the co-decomposition to guide the modifications on the input designs for improving the co-decomposition results. Lastly, the co-construction process does not incorporate a formal stability analysis, and it may not achieve minimal number of global keys in the final results.

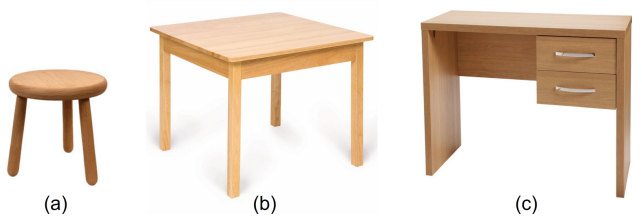


Fig. 19. Typical failure cases: i) the co-decomposition process cannot construct cages with correspondences between the two designs shown in (a) and (b); ii) it cannot find similar substructures between the two designs shown in (b) and (c); and iii) in the absence of cyclic substructures (a), users need to manually add extra parts to form local interlocking groups.

8 CONCLUSION

This paper presents computational methods that serve as tools to aid the design and construction of reconfigurable furniture. By co-analyzing multiple input designs, our method can construct a common set of parts with compatible connecting joints, such that the parts can be reused for building different assemblies, each interlocked by a well-planned network of joints. In summary, this work presents the following contributions. First, we develop a co-decomposition method based on a dynamic bipartite graph model to iteratively decompose the input designs into parts with correspondences, while maximizing the parts reuse and considering the geometric semantics in the given designs. Second, we formulate the half-joint graph to model the joint connections in reconfigurable assemblies; hence, we can efficiently explore the search space of reconfigurable assemblies with various constraints. Third, we devise two new interlocking models, the backward interlocking and multi-key interlocking models, and then develop the co-construction method based on them to explore common substructures and plan joints compatibly over multiple assemblies. In the end, we extend our method with recursive connections to produce extensible and hierarchical reconfigurable assemblies, demonstrate the applicability of our method by constructing reconfigurable interlocking assemblies of various functions and complexities, and fabricate a number of them using 3D printing, 2D laser cutting, and woodworking.

In the future, we plan to include more joint models and analyze the joint stability and strength based on the geometric parameters and orientations of joints. Moreover, we would like to generalize this work to consider external connectors such as hinges, and explore modeling methods like Shao et al. [2016] to prepare the input design models. Lastly, we would also like to generalize the construction of interlocking parts with nonorthogonal joint connections.

ACKNOWLEDGMENTS

We thank the reviewers for the valuable comments, Ziqi Wang for his help in preparing the laser cutting and 3D printing results, and Thomas Maitz of perludi.com for the photos shown in Fig. 2. This work is supported in part by the Hong Kong RGC General Research Funding Projects (14203416 & 14201717), the National Natural Science Foundation of China (61403357, 61672482, 11626253), the One Hundred Talent Project of the Chinese Academy of Sciences, and the ISF-NSFC joint research program (2217/15).

REFERENCES

Timothy G. Abbott, Zachary Abel, David Charlton, Erik D. Demaine, Martin L. Demaine, and Scott Duke Kominers. 2012. Hinged Dissections Exist. *Discrete & Comp. Geom.* 47, 1 (2012), 150–186.

Melinos Averkiou, Vladimir G. Kim, and Niloy J. Mitra. 2016. Autocorrelation Descriptor for Efficient Co-Alignment of 3D Shape Collections. *Comp. Graph. Forum (Eurographics)* 35, 1 (2016), 261–271.

Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. 2012. Fabricating Articulated Characters from Skinned Meshes. *ACM Trans. Graph. (SIGGRAPH)* 31, 4 (2012). Article No. 47.

Jacques Calì, Dan A. Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 2012. 3D-Printing of Non-Assembly, Articulated Models.

Fig. 20. Re-assembling Shoe Rack into Laundry Box. In the two images shown at the beginning and at the end of the animation sequence, colors on parts reveal the part correspondence. See the supplementary video for an animated version of this result.

ACM Trans. Graph. (SIGGRAPH Asia) 31, 6 (2012). Article No. 130.

Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and Fabricating Mechanical Automata from Mocap Sequences. *ACM Trans. Graph. (SIGGRAPH Asia)* 32, 6 (2013). Article No. 186.

Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav Ponamgi. 1995. I-COLLIDE: An Interactive and Exact Collision Detection System for Large-scale Environments. In *Proc. Symp. on Inter. 3D Graph.* 189–196.

M. J. Cohn. 1975. Economical Triangle-square Dissection. *Geometriae Dedicata* 3, 4 (1975), 447–467.

Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Trans. Graph. (SIGGRAPH)* 32, 4 (2013). Article No. 83.

Noah Duncan, Lap-Fai Yu, and Sai-Kit Yeung. 2016. Interchangeable Components for Hands-On Assembly Based Modelling. *ACM Trans. on Graph. (SIGGRAPH Asia)* 35, 6 (2016). Article No. 234.

Noah Duncan, Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. 2017. Approximate Dissections. *ACM Trans. on Graph. (SIGGRAPH Asia)* 36, 6 (2017). to appear.

Greg N. Frederickson. 1997. *Dissections: Plane and Fancy.* Cambridge University Press.

Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. 2015. Computational Interlocking Furniture Assembly. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015). Article No. 91.

Akash Garg, Alec Jacobson, and Eitan Grinspun. 2016. Computational Design of Reconfigurables. *ACM Trans. on Graph. (SIGGRAPH)* 35, 4 (2016). Article No. 90.

Ruizhen Hu, Lubin Fan, and Ligang Liu. 2012. Co-Segmentation of 3D Shapes via Subspace Clustering. *Comp. Graph. Forum (SGP)* 31, 5 (2012), 1703–1713.

Qixing Huang, Vladlen Koltun, and Leonidas Guibas. 2011. Joint Shape Segmentation with Linear Programming. *ACM Trans. Graph. (SIGGRAPH Asia)* 30, 6 (2011). Article No. 125.

Yi-Jheng Huang, Shu-Yuan Chan, Wen-Chieh Lin, and Shan-Yu Chuang. 2016. Making and Animating Transformable 3D Models. *Computers & Graphics (Proc. of CAD/Graphics)* 54 (2016), 127–134.

Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. 2012. Exploring Shape Variations by 3D-Model Decomposition and Part-based Recombination. *Comp. Graph. Forum (Eurographics)* 31, 2 (2012), 631–640.

Javor Kalojanov, Michael Wand, and Philipp Slusallek. 2016. Building Construction Sets by Tiling Grammar Simplification. *Comp. Graph. Forum (Eurographics)* 35, 2 (2016), 13–25.

Bongjin Koo, Wilmot Li, JiaXian Yao, Maneesh Agrawala, and Niloy J. Mitra. 2014. Creating Works-Like Prototypes of Mechanical Objects. *ACM Trans. Graph. (SIGGRAPH Asia)* 33, 6 (2014). Article No. 217.

Evangelos Kranakis, Danny Krizanc, and Jorge Urrutia. 2000. Efficient Regular Polygon Dissections. *Geometriae Dedicata* 80, 1 (2000), 247–262.

Vladislav Kreavoy, Dan Julius, and Alla Sheffer. 2007. Model Composition from Interchangeable Components. In *Pacific Graphics*. 129–138.

Honghua Li, Ruizhen Hu, Ibraheem Alhashim, and Hao Zhang. 2015. Foldabilizing Furniture. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015). Article No. 90.

Han Liu, Ulysse Vimont, Michael Wand, Marie-Paule Cani, Stefanie Hahmann, Damien Rohmer, and Niloy J. Mitra. 2015. Replaceable Substructures for Efficient Part-Based Modeling. *Comp. Graph. Forum (Eurographics)* 34, 2 (2015), 503–513.

László Lovász and Michael D. Plummer. 2009. *Matching Theory.* American Mathematical Society.

Yuliang Rong, Youyi Zheng, Tianjia Shao, Yin Yang, and Kun Zhou. 2016. An Interactive Approach for Functional Prototype Recovery from a Single RGBD Image. *Computational Visual Media* 2, 1 (2016), 87–96.

Tianjia Shao, Dongping Li, Yuliang Rong, Changxi Zheng, and Kun Zhou. 2016. Dynamic Furniture Modeling Through Assembly Instructions. *ACM Trans. on Graph. (SIGGRAPH Asia)* 35, 6 (2016). Article No. 172.

Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. 2011. Unsupervised Co-segmentation of a Set of Shapes via Descriptor-space Spectral Clustering. *ACM Trans. Graph. (SIGGRAPH Asia)* 30, 6 (2011). Article No. 126.

Mélina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. 2015. Interactive Surface Design with Interlocking Elements. *ACM Trans. Graph. (SIGGRAPH Asia)* 34, 6 (2015). Article No. 224.

Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. 2016. CofiFab: Coarse-to-Fine Fabrication of Large 3D Objects. *ACM Trans. on Graph. (SIGGRAPH)* 35, 4 (2016). Article No. 45.

Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. 2012. Recursive Interlocking Puzzles. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6 (2012). Article No. 128.

Peng Song, Zhongqi Fu, Ligang Liu, and Chi-Wing Fu. 2015. Printing 3D Objects with Interlocking Parts. *Comp. Aided Geom. Des.* 35-36 (2015), 137–148.

Timothy Sun and Changxi Zheng. 2015. Computational Design of Twisty Joints and Puzzles. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015). Article No. 101.

Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-Based Characters. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014). Article No. 64.

Francisca Gil Ureta, Chelsea Tymms, and Denis Zorin. 2016. Interactive Modeling of Mechanical Objects. *Comp. Graph. Forum (SGP)* 35, 5 (2016), 145–155.

Oliver van Kaick, Kai Xu, Hao Zhang, Yanzhen Wang, Shuyang Sun, Ariel Shamir, and Daniel Cohen-Or. 2013. Co-hierarchical Analysis of Shape Structures. *ACM Trans. Graph. (SIGGRAPH)* 32, 4 (2013). Article No. 69.

Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2012. Active Co-analysis of a Set of Shapes. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6 (2012). Article No. 165.

Shi-Qing Xin, Chi-Fu Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. 2011. Making Burr Puzzles from 3D Models. *ACM Trans. Graph. (SIGGRAPH)* 30, 4 (2011). Article No. 97.

Jiaxian Yao, Danny M. Kaufman, Yotam Gingold, and Maneesh Agrawala. 2017. Interactive Design and Stability Analysis of Decorative Joinery for Furniture. *ACM Trans. on Graph.* 36, 2 (2017). Article No. 20.

Mehmet Ersin Yumer and Levent Burak Kara. 2012. Co-abstraction of Shape Collections. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6 (2012). Article No. 166.

Yinan Zhang and Devin Balkcom. 2016. Interlocking Structure Assembly with Voxels. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. 2173–2180.

Youyi Zheng, Daniel Cohen-Or, Melinos Averkiou, and Niloy J. Mitra. 2014. Recurring Part Arrangements in Shape Collections. *Comp. Graph. Forum (Eurographics)* 33, 2 (2014), 115–124.

Youyi Zheng, Daniel Cohen-Or, and Niloy J. Mitra. 2013. Smart Variations: Functional Substructures for Part Compatibility. *Comp. Graph. Forum (Eurographics)* 32, 2 (2013), 195–204.

Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. 2014. Boxelization: Folding 3D Objects into Boxes. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014). Article No. 71.

Yahan Zhou and Rui Wang. 2012. An Algorithm for Creating Geometric Dissection Puzzles. In *Bridges Towson: Mathematics, Music, Art, Architecture, Culture*. 49–56.