Proceedings of IDETC/CIE 2005
ASME 2005 International Design Engineering Technical Conferences
& Computers and Information in Engineering Conference
September 24-28, 2005, Long Beach, California USA

DETC2005-85260

DESIGN FOR PRODUCT-EMBEDDED DISASSEMBLY

Shingo Takeuchi and Kazuhiro Saitou
Department of Mechanical Engineering
University of Michigan
Ann Arbor, MI, 48109-2125
E-mail:{stakeuch, kazu}@umich.edu

ABSTRACT

This paper presents a computational method for designing assemblies with a built-in disassembly pathway that maximizes the profit of disassembly while satisfying regulatory requirements for component retrieval. Given component revenues and components to be retrieved, the method simultaneously determines the spatial configurations of components and locator features on the components, such that the product can be disassembled in the most profitable sequence, via a domino-like “self-disassembly” process triggered by the removal of one or a few fasteners. The problem is posed as optimization and a multi-objective genetic algorithm is utilized to search for Pareto-optimal designs in terms of three objectives: 1) the satisfaction of distance specification among components, 2) the efficient use of locator features on components, and 3) the profit of overall disassembly process under the regulatory requirements. A case study with different costs for removing fasteners demonstrates the effectiveness of the method in generating design alternatives under various disassembly scenarios.

Keywords: Design for disassembly, design optimization, computer-aided design, multi-objective genetic algorithm

INTRODUCTION

Increased regulatory pressures (*e.g.*, EU’s WEEE directive) and voluntary initiatives has placed manufacturers more responsibility for end-of-life (EOL) treatments such as material recycling and component reuse. Since both recycling and reuse typically require disassembly, design for disassembly (DFD) has become a key design issue in almost any mass-produced products. DFD is particularly critical in consumer electrical products due to a large number of production and short cycle time for technological obsolescence. Also, components in these products are typically required to fit into a tight enclosing space, which makes disassembly even more challenging.

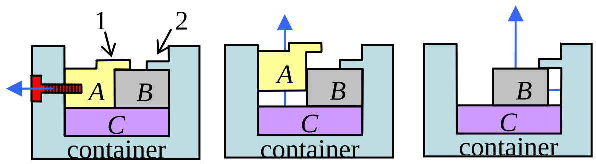
The optimal EOL treatments should be determined based on the profit of disassembly process and environmental impact [1]. In the simplest form, the profit of a disassembly process u can be expressed as:

$$u = \sum_i (r_i - c_i) \tag{1}$$

where r_i is the revenue of the i-th disassembled components and c_i is the disassembly cost of the i-th disassembly operation. While r_i depends only on disassembled components, c_i generally depends on both disassembled components and the spatial configuration of components and fasteners [2]. Note that it is often profitable to stop disassembly before a product is completely disassembled to components.

To prevent manufacturers from pursuing most profitable EOL treatments with potentially serious environmental impacts, regulatory requirements are often imposed. In many countries, for example, the recovery of toxic components such as lead and mercury is obligated regardless of profit. It is therefore desired to design products with the maximum profit of disassembly while satisfying constraints on regulatory requirements.

The above thoughts motivated us to develop a concept of product-embedded disassembly [3], where the relative motions of components are constrained by locator features (such as catches and lugs) integral to components, in such a way that the optimal disassembly sequence is realized via a domino-like “self-disassembly” process triggered by the removal of one or a few fasteners. Figure 1 illustrates the concept. In the assembly shown in Figure 1 (a), suppose the retrieval of component A (valuable material) and component B (toxic material) are desired, and the retrieval of component C (non-valuable material) is not profitable considering disassembly cost. To disassemble the assembly in an optimal fashion, a disassembly operator can simply remove the screw, which activates a disassembly pathway $A \rightarrow B$ as shown in Figures 1 (b) and (c). Thanks to locator 1 on component *A* and locator 2 on the container, the use of fasteners is minimized, which is essential to increase the profit of overall disassembly process.



(a) (b) (c)

Figure 1. Concept of product-embedded disassembly [3].

In our preliminary work [3], the spatial configurations of components and locators are simultaneously determined to retrieve a given set of components in a unique sequence, with no consideration of the profit of overall disassembly process. Also, no rotational motions are considered in determining the spatial configuration of components. As an extension, this paper presents a method where the profit of disassembly process, as defined in Equation (1), is explicitly treated as an objective to maximize. Given component revenues and components to be retrieved, the method simultaneously determines the spatial configurations of components and locator features on the components, such that the product can be disassembled in the most profitable sequence, via a domino-like "self-disassembly" process triggered by the removal of one or a few fasteners. The problem is posed as optimization and a multi-objective genetic algorithm [4, 5] is utilized to search for Pareto-optimal designs in terms of three objectives: 1) the satisfaction of distance specification among components, 2) the efficient use of locator features on components, and 3) the profit of overall disassembly process under regulatory requirements. A case study with different costs for removing fasteners demonstrates the effectiveness of the method in generating design alternatives under various disassembly scenarios.

PREVIOUS WORKS

Design for Disassembly

Design for disassembly (DFD) is a class of design methods and guidelines to enhance the ease of disassembly for product maintenance and/or EOL treatments [6]. Kroll *et al*. [7] utilized disassembly evaluation charts to facilitate the improvements of product design. Das *et al*. [8] introduced the Disassembly Effort Index (DEI) score to evaluate the ease of disassembly. Reap *et al*. [9] reported DFD guidelines for robotic semi-destructive disassembly, where detachable or breakable snap fits are preferred to screws due to their ease of disengagement. O'Shea *et al*. [10] focused on tool selection during disassembly where the optimal tool selection path in terms of the ease of disassembly is produced via dynamic programming. Matsui *et al*. [11] proposed the concept of Product Embedded Disassembly Process, where a means of part separation that can be activated upon disassembly is embedded within a product. As an example, they developed cathode-ray tube (CRT) with a Nichrome wire embedded along the desired separation line, which can induce thermal stress to crack the glass of the CRT tube upon the application of current.

While these works suggest locally redesigning an existing assembly for improving the ease of its disassembly, they do not address the simultaneous decision of the spatial configuration of components and joints for improving an entire disassembly processes.

Disassembly Sequence Planning

Disassembly sequence planning (DSP) aims at generating feasible disassembly sequences for a given assembly, where the feasibility of a disassembly sequence is checked by the existence of collision-free motions to disassemble each component or subassembly in the sequence. Since the disassembly sequence generation problem is NP-complete, the past researches have focused on efficient heuristic algorithms to approximately solve the problem. Based on a number of important research results on assembly sequence planning [12-16], several automated disassembly sequence generation approaches for 2/2.5D components have been developed [17-21]. More recent works are geared towards DSP with special attentions to reuse, recycling, remanufacturing, and maintenance. Lambert [22] built a linear programming model to obtain the optimal EOL disassembly. Li *et al*. [23] used Genetic Algorithm (GA) combined with Tabu search [24, 25] to find the optimal disassembly sequence for maintenance.

These works, however, only address the generation and optimization of disassembly sequences for an assembly with a pre-specified spatial configuration of components. Since the feasibility of disassembly sequences largely depends on the spatial configuration of components, this would seriously limit the opportunities for optimizing an entire assembly. In addition, these works do not address the design of joint configurations, which also has a profound impact on the feasibility and quality of a disassembly sequence.

Configuration Design Problem

While rarely discussed in the context of disassembly, the design of the spatial configuration of given shapes has been an active research area by itself. Among the most popular flavors is the bin packing problem (BPP), where the total volume (or area for 2D problems) a configuration occupies is to be minimized. Since this problem is also NP-complete, heuristic methods are commonly used. Fujita *et al*. [26] proposed hybrid approaches for a 2D plant layout problem, where the topological neighboring relationships of a layout are determined by Simulated Annealing (SA), whereas the generalized reduced gradient (GRG) method determines the geometry. Kolli and Cagan [27] used SA for packing 3D components with arbitrary geometry. GA is also widely used for the configuration design problem. Corcoran *et al*. [28] solved a 3D packing problem with GA using multiple crossover methods. Jain *et al*. [29] adopted discrete representation as an object expression and proposed a geometry-based crossover operation for a 2D packing problem. Grignon *et al*. [30] proposed a configuration design optimization method by using multi-objective GA, where static and dynamic balances and maintainability are considered in addition to configuration volume.

These works, however, do not address the integration with DSP.

METHOD

The proposed method can be summarized as the following optimization problem:

- **Given:** component geometries, component revenues, components to be retrieved, distance specification, and locator library and its priority set
- **Find:** component configuration, locator configuration on each component
- **Subject to:** no floating component, no overlap among components, no unfixed component prior to disassembly
- **Minimizing:** redundant use of locator features, violation of distance specification
- **Maximizing:** profit of disassembly to retrieve required components

Since the problem has three objectives, Pareto optimal solutions will be obtained as outputs, using a multi-objective genetic algorithm (MOGA) [4, 5]. The rest of the section describes the method in detail.

Inputs

The following inputs are assumed as given:

- **Component geometries:** As in [3, 31, 32, 33], the component geometries are represented by voxels, due to the efficiency in checking contacts and the simplicity in modifying geometries. CAD inputs are first voxelized using ACIS® solid modeling kernel.
- **Component revenues** (r_i in Equation (1)): They are the amounts of revenue each component yield through reuse and recycling. Note that the costs of disassembly (c_i in Equation (1)) are calculated based on the disassembly motions of each component.
- **Components to be retrieved**: It is a (small) subset of components that must be retrieved due to regulatory requirements, regardless of their revenues.
- **Distance specification**: The distances among components are often constrained by their functional relationships. For example, a cooling fan should be positioned near a CPU in the component configuration of a laptop computer. Since the distances between some pairs of components are more important than the others, the distance specification is defined as a set of the weights of importance for the distances between pairs of components (measured between two designated voxels) that need to be minimized. If the weight between two components is not defined, their distance is considered unimportant and can be arbitrary chosen. Figure 2 shows an example of the distance specification among five components.
- **Locator library**: Since types of feasible locators depend on manufacturing and assembly processes, they are pre-specified by a designer as a locator library. It is a set of locators for a specific application domain, which can be potentially added on each component to constrain its motion. Figure 3 shows an example of five locators[1] in the locator library used in the following case study. Locator constraint (*LC*) shown in the third column of Figure 3 illustrates a set of directions locators constrain when they are oriented as shown in the second column, formally represented as a subset of {-*x*, +*x*, -*y*, +*y*, -*z*, +*z*}.
- **Priority set**: As seen in Figure 3, multiple locator types in a locator library can constrain the motion in the same direction. Since a component often needs to be constrained in multiple directions, the selection of locators on a component to constrain specified directions can be non-trivial. To minimize the generation of infeasible locator selections during optimization, the locator configuration of a component is dynamically constructed by testing locator types, in a specified sequence, for constraining each specified direction. Priority set is a set of potential sequences (specified by a designer) in which locator types are tested during the construction of locator configurations.

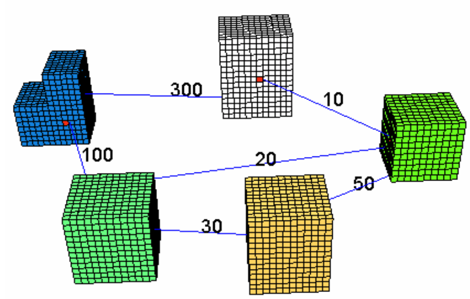


Figure 2. An example of the distance specification. The labeled lines between two voxels indicate the weights of importance of minimizing the respective distances.

Design Variables

There are two design variables for the problem. The first design variable, *configuration vector,* represents the spatial configuration and dimensional change of each component:

$$\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n\text{-}1}) \tag{2}$$

$$\boldsymbol{x}_i = (\boldsymbol{t}_i, \boldsymbol{r}_i, \boldsymbol{d}_i,); \quad i = 0, 1, \ldots n\text{-}1 \tag{3}$$

where n is the number of components in the assembly, $\boldsymbol{t}_i$ and $\boldsymbol{r}_i$ are the vectors of the translational and rotational motions of component i with respect to the global reference frame, and $\boldsymbol{d}_i$ =

[1] Fasteners are considered as a special case of locators and are included in a locator library

(d_0, d_1, …, $d_{f\text{-}1}$) is a vector of the offset values of the faces of component i in their normal directions. Since the voxel representation is used, translations and offsets are limited to the multiples of the size of a voxel. Similarly, rotations are limited to +90°, -90° and +180°. Note that the dimensional changes are considered only for the components whose dimensions are assumed unfixed.

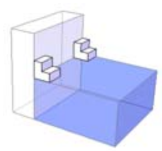
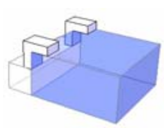
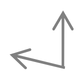
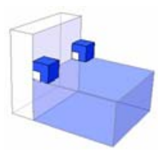
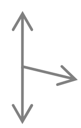
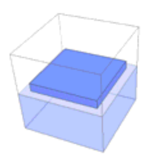
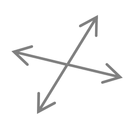
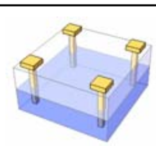
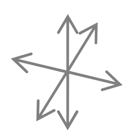
Type	Geometry	*LC*
(a) Catch		
(b) Lug		
(c) Track		
(d) Boss		
(e) Screw		

Figure 3. Locator library used in the case study: (a) Catch, (b) Lug, (c) Track, (d) Boss, and (e) Screw.

The second design variable, *locator vector*, indirectly represents the spatial configuration of the locator features on each component:

$$\boldsymbol{y} = (\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{m\text{-}1}) \quad (4)$$

$$\boldsymbol{y}_i = (CD_i, p_i); \quad i = 0, \ldots, m\text{-}1 \quad (5)$$

where $m = n\,(n\text{-}1)/2$ is the number of pairs of components in the assembly, and $CD_i \subseteq \{-x, +x, -y, +y, -z, +z\}$ is a set of directions in which the motion of component c_0 in the i-th pair (c_0, c_1) is to be constrained, and p_i is a sequence in the priority set, in which locator types in the locator library are tested during the construction of the locator configuration of the i-th pair.

Given $\boldsymbol{y}_i = (CD_i, p_i)$, the locator configuration of the i-th pair of components c_0 and c_1 is constructed by testing locator types, in sequence p_i, for constraining each direction in CD_i as follows:

1. For each $d \in CD_i$, remove d from CD_i if the motion of c_0 in $d \in CD$ is constrained by other components or locators. This step is necessary to reduce the redundant use of locator features.
2. Remove locator type t at the beginning of p_i. If p_i is empty, return FALSE.
3. Select direction $d \in CD$.
4. Find an orientation of o of locator type t whose locator constraint LC (after re-orientation) contains d. If several orientations are found, select an orientation with maximum $|\, LC \cap CD_i|$. If none is found, go to step 2.
5. Add t to c_0 or c_1 in o.
6. $CD_i \leftarrow CD_i \setminus LC$. If $CD = \varnothing$, return TRUE. Otherwise, go to step 3.

The above procedure returns TRUE if a locator configuration constraining all directions in CD_i is found by using the locator types in p_i, and FALSE otherwise. During optimization, the value of $\boldsymbol{y}_i$ returning FALSE is considered as infeasible.

Figure 4 shows an example construction of locator configuration of components c_0 and c_1 according to the above procedure with CD = {+z} and p = <Catch, Screw, Lug, Track, Boss>:

- Step 1: Since component c_1 does not constrain the motion of c_0 in +z (Figure 4 (a)), +z remains in CD
- Step 2: Remove Catch from p. Since p = <Screw, Lug, Track, Boss> is non-empty, proceed.
- Step 3: Select +z from CD.
- Step 4: Systematically examine the possible orientations of Catch on c_0 and c_1 to find the orientations that constraint +z (o_0 through o_7 in Figure 4 (b) and (c)). Note, however, that the orientations other than o_0 and o_5 in Figure 4 (d) are infeasible due to the lack of an adjacent component. Since both o_0 and o_5 has $|\, LC \cap CD_i| = |\{+z\} \cap \{+z\}| = |\{+z\}| = 1$, o_0 is chosen.
- Step 5: Catch in orientation o_0 is added to c_1 (Figure 4 (e)).
- Step 6: Since $CD_i \setminus LC = \{+z\} \setminus \{+z\} = \varnothing$, $CD_i = \varnothing$. Return TRUE.

Figure 5 illustrates how two different values of priority sequence p with the same CD can result in the different locator configurations. For the two components in Figure 5 (a) with CD = {-x, +x, +z}, sequence p = <Track, Boss, Screw, Catch, Lug> results in the locators in Figure 5 (b), whereas sequence p = <Catch, Lug, Screw, Track, Boss> results in the locators in Figure 5 (c). In Figure 5 (c), two locator types, Catch and Lugs are used since Catch (top priority) cannot be oriented to constrain c_0 in +z direction while Lug (second priority) can.

While indirect, constraint direction CD and priority sequence p realizes a compact representation of a locator configuration of a pair of components. Compared to the direct representation in [3] that specifies the existence of a locator type in an orientation at a potential location on a component, it can generate far fewer infeasible locator configurations during the "generate and test" process of genetic algorithms. As a result, the computational efficiency is dramatically improved. Instead of treating the priority sequence as a design variable, one might imagine checking for locator types always in the (fixed) ascending sequence of their manufacturing costs is sufficient. However, such costs are difficult to determine *a priori*, since the actual geometry (and hence the cost) of a locator heavily dependents on the configuration of the surrounding components.

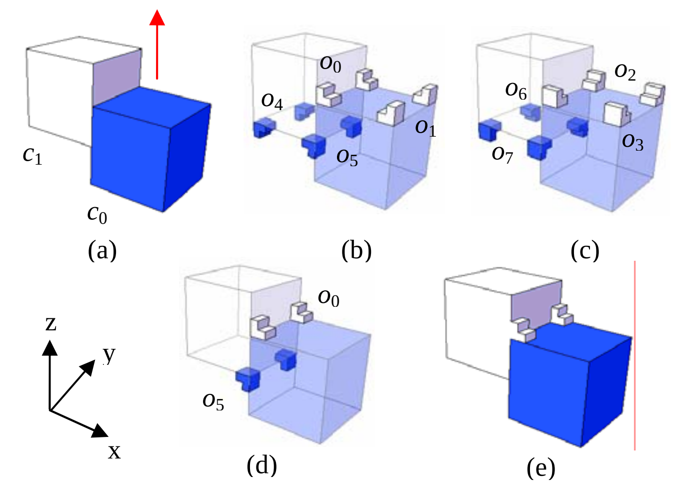


Figure 4. An example construction of locator configuration: (a) two components, (b) and (c) possible orientations of Catch, (d) two feasible orientations, and (e) final locator configuration.

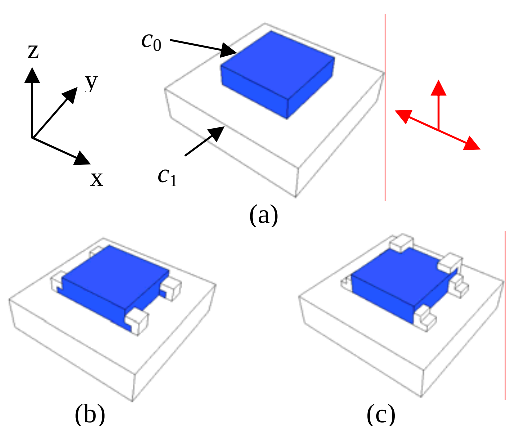


Figure 5. Influence of priority sequence *p* in locator configurations: (a) two components with *CD* = {-*x*, +*x*, +*z*}, (b) locators constructed with *p* = <Track, Boss, Screw, Catch, Lug>, and (c) locators constructed with *p* = <Catch, Lug, Screw, Track, Boss>.

Constraints

The locations of components as specified by ***x***, whose geometries are altered by adding the locator features constructed from ***y***, must satisfy the following three constraints:

1. No floating components
2. No over-lap components
3. No unfixed component prior to disassembly

Prior to the evaluation of the objective functions for ***x*** and ***y***, these constraints are checked by using the standard geometric algorithms for mobility and contact analyses. Voxel representation allows the very efficient execution of these algorithms. While constraint 1 is a necessary condition for constraint 3, they are separated here to indicate the fact that constraint 1 is used as a pre-screeding for constraint 3 during the optimization process.

Objective Functions

The configurations of components and locator features on each component specified by ***x*** and **y** are evaluated according to three criteria: 1) the satisfaction of distance specification among components, 2) the efficient use of locator features on components, and 3) the profit of overall disassembly process under regulatory requirements for component retrieval.

The first objective function (to be minimized) is for the satisfaction of the distance specification, given as:

$$f_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_i w_i d_i \tag{6}$$

where w_i is the weight of importance of the i-th distance in the distance specification and d_i is the distance between two designated voxels.

The second objective function (to be minimized) is for the efficient use of locator features, given as the total increase in manufacturing cost due to the addition of locator features to components:

$$f_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_i c_i \tag{7}$$

where c_i is the manufacturing cost of the i-th locators in the assembly.

The third objective function (to be maximized) is for the profit of the overall disassembly process under the regulatory requirements of component retrieval. Since assembly $a(\boldsymbol{x}, \boldsymbol{y})$ specified by ***x*** and ***y*** can generally be disassembled in multiple sequences, the objective function is defined as the profit of the best (most profitable) disassembly sequence with the penalty of un-retrieving components in *RC*, the input set of components to be retrieved:

$$f_3(\boldsymbol{x}, \boldsymbol{y}) = \max_{q \in Q_{xy}} \{ \max_{pq \in P_q} u\,(a, pq) - w \cdot v(a, pq^*) \} \tag{8}$$

where:

- Q_{xy} is the set of all 2-disassembly sequences [31] (each disassembly step consists of less than two consecutive translations of a
- P_q is the set of sub-sequences of $q \in Q_{xy}$ in which a is only partially disassembled
- $u(a, pq)$ is the profit of disassembling a in $pq \in P_q$
- pq^* is the sub-sequence of q that gives $\max_{pq \in P_q} u\,(a, pq)$
- $v(a, pq^*)$ is the number of components in RC that are not retrieved by disassembling a in pq^*
- w is weight

It is assumed that a disassembly sequence and a set of disassembly sequences are represented as a binary tree and a AND/OR graph [11], respectively. Accordingly, Q_{xy} is computed as follows:

1. Set a component (*eg*., container) as the fixed component, and push the assembly to stack S and Q_{xy}
2. Pop a subassembly s from S
3. For each subassembly $ss \subset s$ that does not contain any fixed components, check the 2-disassemblability of ss and $st = s \backslash ss$. If ss and st are 2-disassemblable, add ss and st to Q_{xy}. If ss is composed of multiple components and contains components in RC_i, push ss to S. Also, do the same for st.
4. If $S = \emptyset$, return. Otherwise go to step 2.

where the 2-disassembleability of two subassemblies ss and st are checked as follows [31, 34]:

1. For each mating surfaces between ss and st (including the ones of the locators), obtain a set of constrained directions as a subset of six possible translational directions $D = \{-x, +x, -y, +y, -z, +z\}$.
2. Compute constrained directions CD_{st} between ss and st as a union of all constrained directions obtained in step 1.
3. If $D \backslash CD_{st} = \emptyset$, return FALSE.
4. If there exist a direction in $D \backslash CD_{st}$ along which ss can be moved infinitely without a collision, return *TRUE* (*ss* is 1-disassembleable).
5. Select a direction d in $D \backslash CD_{st}$. If all have been selected, return infeasible. Otherwise, go to the next step.
6. Move ss by unit length along d. If ss collides with other components, go to step 5.
7. If ss is 1-disassembleable at the current location, return TRUE (*ss* is 2-disassembleable). Otherwise, go to step 6.

Given a 2-disassembly sequence $q \in Q_{xy}$, the maximum profit $u_a \equiv u\,(a, pq^*)$ among all partial disassembly sequence of q in Equation (8) can be obtained by following the disassembly steps in q until the continuation is unprofitable. Considering a disassembly step in q that disassembles subassembly s into two subassemblies ss and st, the maximum profit u_s of partially disassembling s in sub-sequences of q can be recursively defined as follows:

$$u_s = \begin{cases} r_s & \text{if } s \text{ is a component} \\ 0 & \text{if } v(s) = 0 \text{ and } u_{ss} + u_{st} - c_s < 0 \\ u_{ss} + u_{st} - c_s & \text{otherwise} \end{cases} \quad (9)$$

where r_s is the revenue of s (if s is a component), $v(s)$ is the number of components in RC contained in s, and c_s is the cost of disassembling s into ss and st. The condition $v(s) = 0$ is necessary for the case $u_s = 0$, in order to ensure that disassembly continues as long as there is a chance of retrieving the components in RC regardless of the profit.

The disassembly cost c_i in Equation (9) depends on the orientation changes, the moved distance, and the accessibility of fasteners during the disassembly operation, and is given by:

$$c_i = \sum_{j=0}^{2} \omega_j \cdot dc_j \quad (10)$$

where dc_0 is the number of orientation changes, dc_1 is the sum of the moved distances of disassembled components, dc_2 is the sum of accessibilities ac_f of removed screws and ω_j is the weight of dc_j. The accessibility ac_f of a screw is defined as:

$$ac_f = 1.0 + \omega_a / (aa + 0.01) \quad (11)$$

where ω_a is weight and aa is the area of the mounting face of the screw, accessible from outside of the product in its normal direction.

Optimization Algorithm

Since design variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are discrete and there are three objectives, the problem is solved by using a multi-objective genetic algorithm [4,5]. A multi-objective genetic algorithm is an extension of the conventional (single-objective) genetic algorithms, which does not require multiple objectives to be aggregated to one value, for example, as a weighted sum. Instead of static aggregates such as a weighted sum, it dynamically determines an aggregate of the values of multiple objective functions of a candidate solution based on its relative quality in the current population. The proposed research will use the non-dominated sorting genetic algorithm, where the relative quality of a candidate solution is measured in terms of the number of solutions dominating it in the current population.

Chromosome c, an internal representation of design variables for genetic algorithms, is defined as a simple list of the two design variables:

$$c = (\boldsymbol{x}, \boldsymbol{y}) \quad (12)$$

Since the information in $\boldsymbol{x}$, $\boldsymbol{y}$ are linked to the geometry of a candidate design, the conventional one point or multiple point crossovers for linear chromosomes are ineffective in preserving high-quality building blocks [35]. Accordingly, a geometry-based crossover operation based on [29] is adopted:

1. Randomly select a point in the bounding box of the assembly.
2. Cut two parent designs p_1 and p_2 with the three planes parallel to x, y, z axes, and passing through the point selected in step 1, into eight pieces each (Figure 6 (a)).
3. Assemble two child designs c_1 and c_2 by alternately swapping the pieces of p_1 and p_2 (Figure 6 (b)).
4. Repair c_1 and c_2 by moving each component C to the child containing the larger volume (of the sliced piece) of C. If c_1 and c_2 contain the same volume, C is placed in the same way as the parent with the higher rank [4,5].
5. Add locators to c_1 and c_2 by checking which parent each pair of component is inherited from. If a child contains both components of a pair, the corresponding locator is added to the child. Otherwise, a locator is randomly added to either child.

CASE STUDY

The proposed method is applied to an assembly composed of 10 components with a distance specification shown in Figure 7, where component A (container) is considered as fixed, the revenues r_c of components are listed in Table 1, and $RC = \{C, G\}$. The locator library in Figure 3 is used and the manufacturing costs of locators in the locator library are listed in Table 2. Note that the manufacturing cost of screws is low, while their disassembly cost tends to be higher than other locators reflecting additional efforts to remove them.

In order to examine the effect of the cost of removing screws on assembly design, the results are obtained with two sets of weights in Equations (10) and (11). The difference between the weights for Cases 1 and 2 are , ω_2 in the fourth column, the weight for the sum of the accessibilities of the screws removed during disassembly. For both cases, the number of population of 150 and the maximum number of generation of 1200 are used for the genetic algorithm. The running time for both cases is approximately 336 hours (two weeks) with a standard desktop PC.

For case 1, thirty-eight (38) Pareto optimal designs are obtained. Figure 8 shows five designs R_{11}, R_{12}, R_{13}, R_{14} and R_{15} that enable the retrieval of all components in RC, whose objective function values are listed in Table 4. For case 2, forty-five (45) Pareto optimal designs are obtained. Figure 9 shows four designs R_{21}, R_{22}, R_{23} and R_{24} that enable the retrieval of all components in RC, whose objective function values are listed in Table 5.

Designs R_{13} and R_{23} utilize only one fastener, whose removal activates a disassembly pathways as illustrated in Figure 1. Figure 10 shows one of 7332 optimal disassembly sequences for A_{13} obtained by evaluating 11228 feasible disassembly sequences. Upon the removal of the screw that fixes component A and F, all components are disassembled to gain the maximum profit of disassembly. Similarly, Figure 11 shows one of 2400 optimal disassembly sequences for R_{23} obtained by evaluating 178018 feasible disassembly sequences. Upon the removal of the screw that fixes component A and C, all components except for J are disassembled to gain the maximum profit. This is because the orientation change is required to disassemble J, and hence, the disassembly cost to disassemble J becomes higher than its revenue. Although the orientation change is also required to retrieve B and C in R_{13}, they are disassembled since B is included in RC and the revenue of C is still higher than the disassembly cost.

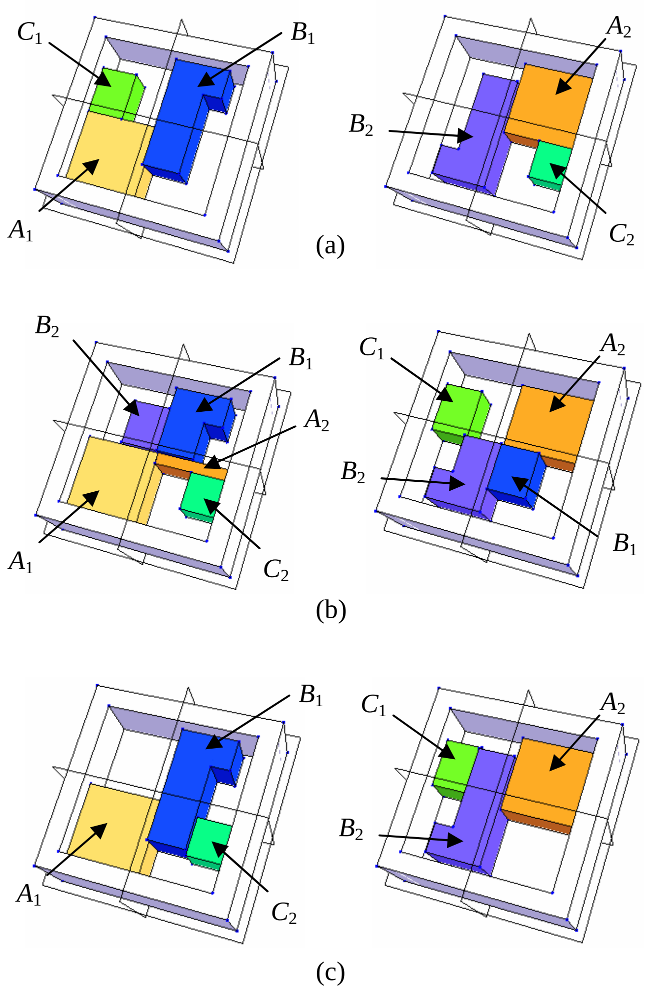


Figure 6. Geometry-based crossover operator. (a) two parents p_1 (left) and p_2 (right), (b) two children c_1 (left) and c_2 (right) after crossover, and (c) two children c_1 (left) and c_2 (right) after repair.

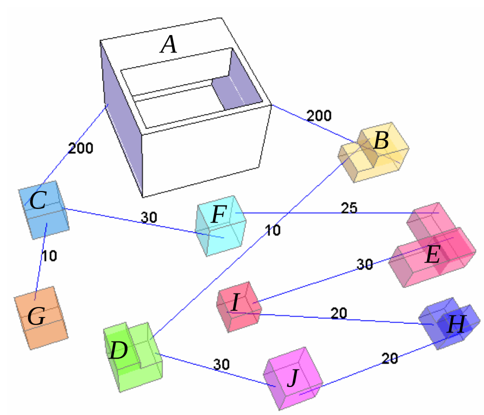


Figure 7. Distance specifications of the example assembly for the case study.

Table 1. Revenues of components in assembly in Figure 7.

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
fixed	200	50	1000	300	300	50	200	50	100

Table 2. Manufacturing cost of the locators in the locator library.

Locator	Lug	Track	Catch	Boss	Screw
Mfg cost	20	30	10	70	20

Table 3. Weights in Equations 10 and 11 for Cases 1 and 2.

parameter	ω_0	ω_1	ω_2	ω_a
Case 1	1.5	37.5	100	10
Case 2	1.5	37.5	10	10

Designs R_{11} and R_{21} are the same design with the minimum f_2 (manufacturing cost). Since the manufacturing cost of screws is inexpensive, seven screws are used instead of locators in the design. Figure 12 shows one of 92112 optimal disassembly sequences for R_{11} and R_{21} obtained by evaluating 385358 feasible disassembly sequences. Due to high ω_2 in Case 1, component *B* and *I* are not disassembled in R_{11}, whose retrieval would require the removal of two screws. Due to low ω_2 in Case 2, on the other hand, all components are disassembled in R_{21}.

CONCLUSION AND FUTURE WORK

This paper presented a computational method for designing an assembly that can be disassembled via a domino-like “self-disassembly” process in the most profitable sequence, triggered by the removal of one or a few fasteners. The problem is posed as the simultaneous determination of the spatial configurations of components and locators, which minimize the violation of the distance specification among components and the cost of locators on components, and maximize the profit of overall disassembly process under the regulatory requirements. A simple case study with different costs for removing fasteners demonstrated that the method can effectively generate design alternatives.

Although the resulting designs cannot be used directly as the final design due to a number of other design factors, they would provide early insights to designers during conceptual design stages. The future work includes the integration with an LCA to quantify the trade-off between economical profitability and environmental impact of products as studied in [1], and the application to more realistic examples. The improvements in the computational speed will also be addressed the use of an alternative optimization algorithm.

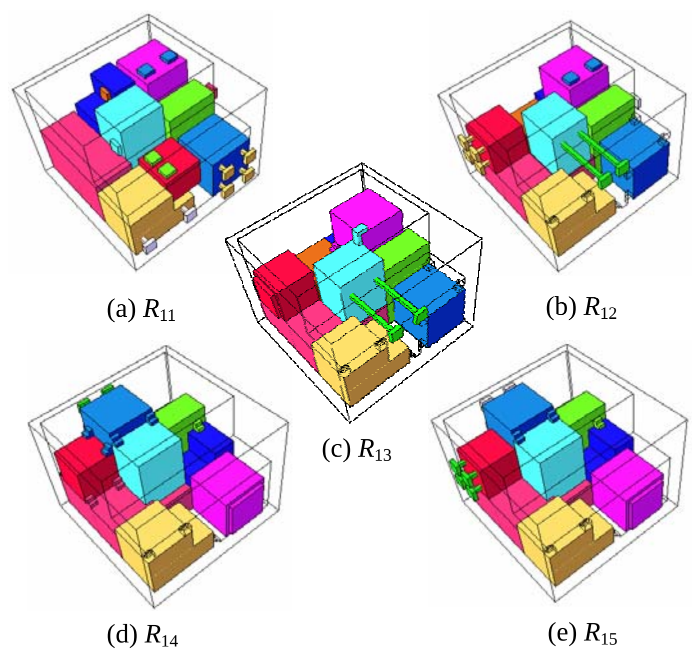


Figure 8. Pareto Optimal Solutions (a) R_{11}, (b) R_{12}, *(c)* R_{13}, (d) R_{14}, (e) R_{15}.

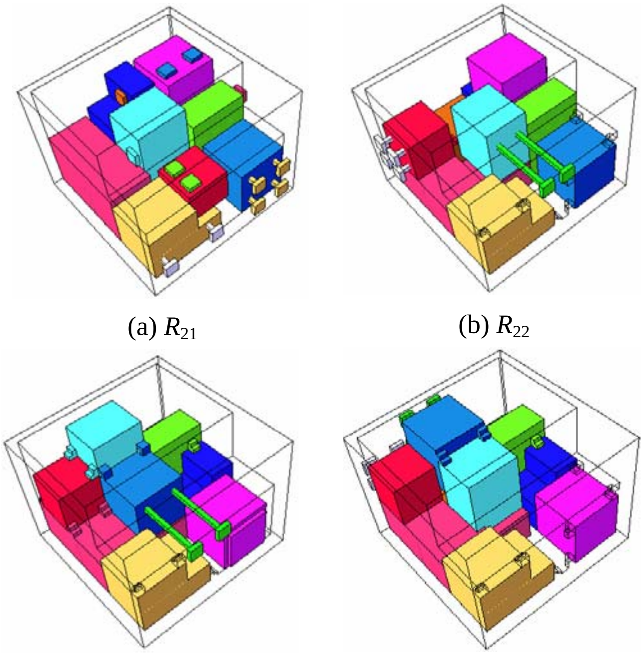
Table 4. Objective function values for R_{11}, R_{12}, R_{13}, R_{14} and R_{15}.

	f_1 (distance spec.)	f_2 (mfg. cost)	f_3 (dissasm. cost)
R_{11}	4908.66	160	937.826
R_{12}	5581.06	200	1103.53
R_{13}	5668.18	480	1355.61
R_{14}	12114.5	360	1390.06
R_{15}	12114.5	400	1470.06

ACKNOWLEDGMENTS

The authors acknowledge funding provided by National Science Foundation (BES-0124415) for this research. Any options, findings, and conclusions or recommendations expressed in this material are those of the authors and do not

necessarily reflect the views of the National Science Foundation.



(a) R_{21} (b) R_{22}

(c) R_{23} (d) R_{24}

Figure 9. Pareto Optimal Solutions (a) R_{21}, (b) R_{22}, *(c)* R_{23}, (d) R_{24}.

Table 5. Objective function values for R_{21}, R_{22}, R_{23} and R_{24}.

	f_1 (distance spec.)	f_2 (mfg. cost)	f_3 (dissasm. cost)
R_{21}	4908.66	160	1466.86
R_{22}	5573.34	180	1509.36
R_{23}	10298.0	450	1564.92
R_{24}	12170.6	330	1555.31

REFERENCES

[1] A. Hulla, K. Jalali, K. Hamza, S. Skerlos, K. Saitou, 2003, "Multi-criteria Decision Making for Optimization of Product Disasembly under Multiple Situatioins," *Environmental Science and Technology*, Special issue on Principles of Green Engineering, vol. 37, No. 23, pp. 5303 -5313.

[2] D. Shetty, K. Rawolle, and C. Campana, 2000, "A New Methodology for Ease-Of-Disassembly in Production Design," *Recent Advances in Design for Manufacture (DFM)*, vol. 109, pp. 39-50, ASME.

[3] S. Takeuchi and K. Saitou, 2005, "Design for Product-Embedded Disassembly Pathways," Proceedings of the *IEEE Conference on Automation Science and Engineering*, August 1-2, Edmonton, Canada, in press.

[4] C. M. Fonseca and P. J. Fleming, 1993, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, pp. 416-423.

[5] C. A. Coello, D. A. Veldhuizen, and G. B. Lamont, 2002, *Evolutionary algorithms for solving multi-objective optimization problems*, Kluwer Academic Publishers, ISBN-0306467623.

[6] G. Boothroyd and L. Alting, 1992, "Design for assembly and disassembly," *Annals of CIRP*, vol. 41, no. 2.

[7] E. Kroll, B. Beardsley, and A. Parulian, 1996, "A Methodology to Evaluate Ease of Disassembly for Product Recycling," *IIE Transactions*, vol. 28, No 10, pp. 837-845.

[8] S. K. Das, P. Yedlarajiah, and R. Narendra, 2000, "An Approach for Estimating the End-Of-Life Product Disassembly Effort and Cost," *International Journal of Production Research*, vol. 38, No. 3, 657-673.

[9] J. Reap and B. Bras, 2002, "Design for Disassembly and the Value of Robotic Semi-Destructive Disassembly," Proceedings of the *ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (Paper No. DETC2002/DFM-34181), September 29 – October 2, Montreal, Canada.

[10] B. O'Shea, H. Kaebernick, S. S. Grewal, H. Perlewitz, K. Müller, and G. Seliger, 1999, "Method for Automatic Tool Selection for Disassembly Planning," *Assembly Automation*, vol. 19, No. 1, pp. 47-54.

[11] K. Matsui, K. Mizuhara, K. Ishii, and R. M. Catherine, 1999, "Development of Products Embedded Disassembly Process Based on End-Of-Life Strategies," *EcoDesign 1999, First International Symposium on Environmentally Conscious Design and Inverse Manufacturing*, February 1-3, pp. 570-575.

[12] L. S. Homem dé Mello and A. C. Sanderson, 1990, "AND/OR Graph Representation of Assembly Plans," *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 188-99.

[13] T. L. De Fazio and D. E. Whitney, 1987, "Simplified Generation of All Mechanical Assembly," *IEEE Journal of Robotics and Automation*, vol. 3, No. 6, pp. 640-658.

[14] S. Lee and Y. G. Shin, 1990, "Assembly Planning Based on Geometric Reasoning," *Computer and Graphics*, vol. 14, No. 2, pp. 237-250.

[15] L. S. Homem dé Mello and A. C. Sanderson, 1991, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 228-40.

[16] D. F. Baldwin, T. E. Abell, M.-C. M. Lui, T. L. De Fazio, and D. E. Whitney, 1992, "An Integrated Computer Aid for Generating and Evaluating Assembly Sequences for Mechanical Products," *IEEE Transactions on Robotics and Automation*, vol. 7, No. 1, pp. 78-94.

[17] T. C. Woo and D. Dutta, 1991, "Automatic Disassembly and Total Ordering in Three Dimensions," *Transactions of ASME, Journal of Engineering for Industry*, vol. 113, pp. 207-213.

[18] D. Dutta and T. C. Woo, 1995, "Algorithm for Multiple Disassembly and Parallel Assemblies," *Transactions of ASME, Journal of Engineering for Industry*, vol. 117, pp. 102-9.

[19] S.-F. Chen, J. H. Oliver, S.-Y. Chou and L.-L. Chen, 1997, "Parallel disassembly by onion peeling," *Transactions of ASME, Journal of Mechanical Design*, vol. 119, pp. 267-274.

[20] H. Srinivasan and R. Gadh, 2000, "Efficient Geometric Disassembly of Multiple Components from An Assembly Using Wave Propagation," *Transactions of ASME, Journal of Mechanical Design*, vol. 122, pp. 179-184.

[21] S. G. Kaufman, R. H. Wilson, R. E. Jones, T. L. Calton and A. L. Ames, 1996, "The Archimedes 2 Mechanical Assembly Planning System," Proceedings of the *1996 IEEE International*

Conference on Robotics and Automation, April, 1996, Minneapolis, Minnesota.

[22] A. J. D. Lambert, 1999, "Optimal Disassembly Sequence Generation for Combined Material Recycling and Part Reuse," *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning*, Portugal, pp. 146-151.

[23] J. R. Li, S. B. Tor and L. P. Khoo, 2002, "A Hybrid Disassembly Sequence Planning Approach for Maintenance," *Transactions of ASME, Journal of Computing and Information Science in Engineering*, vol. 2, pp. 28-37.

[24] F. Glover, 1977, "Heuristics for Integer Programming Using Surrogate Constraints," *Journal of Decision Science*, vol. 8, pp. 156-166

[25] F. Glover, 1986, "Further Paths for Integer Programming and Links to Artificial Intelligence," *Journal of Computer and Operations Research*, vol. 13, pp. 533-549.

[26] K. Fujita, S. Akagi, and S. Shimazaki, 1996, "Optimal Space Partitioning Method Based on Rectangular Duals of Planar Graphs," *JSME International Journal*, vol. 60, pp. 3662-3669.

[27] A. Kolli, J. Cagan and R. Rutenbar, 1996, "Packing of Generic, Three-Dimensional Components Based on Multi-Resolution Modeling," Proceedings of the *1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, August 18-22, 1996, Irvine, California.

[28] A. L. Corcoran III and R. L. Wainwright, 1992, "A Genetic Algorithm for Packing in Three Dimensions," Proceedings of the *1992 ACM/SIGAPP Symposium on Applied Computing*, Kansas City, Missouri.

[29] S. Jain and H. C. Gea, 1998, "Two-Dimensional Packing Problems Using Genetic Algorithm," *Journal of Engineering with Computers*, 1998, vol. 14, pp. 206-213.

[30] P. M. Grignon and G. M. Fadel, 1999, "Configuration Design Optimization Method," Proceedings of the *1999 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, (Paper No. DETC99/DAC-8575), September 12-15, Las Vegas, Nevada.

[31] D. Beasley and R. R. Martin, 1993, "Disassembly Sequences for Objects Built from Unit Cubes," *Journal of Compute-Aided Design*, vol. 25, No. 12.

[32] S. Minami, K. F. Pahng, M. J. Jakiela and A. Srivastave, 1995, "A Cellular Automata Representation for Assembly Simulation and Sequence Generation," *IEEE International Symposium on Assembly and Task Planning*, pp.56-65, August 10-11, Pittsburgh, Pennsylvania.

[33] R. C. W. Sung, J. R. Corney and D. E. R. Clark, 2001, "Automatic Assembly Feature Recognition and Disassembly Sequence Generation," *Transactions of ASME, Journal of Computing and Information Science in Engineering*, vol. 1, pp. 291-299.

[34] T. C. Woo and D. Dutta, "Automatic Disassembly and Total Ordering in Three Dimensions," *Transactions of ASME, Journal of Engineering for Industry*, 1991, vol. 113, pp. 207-213.

[35] D. E. Goldberg, 1989, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, Reading, Massachusetts.

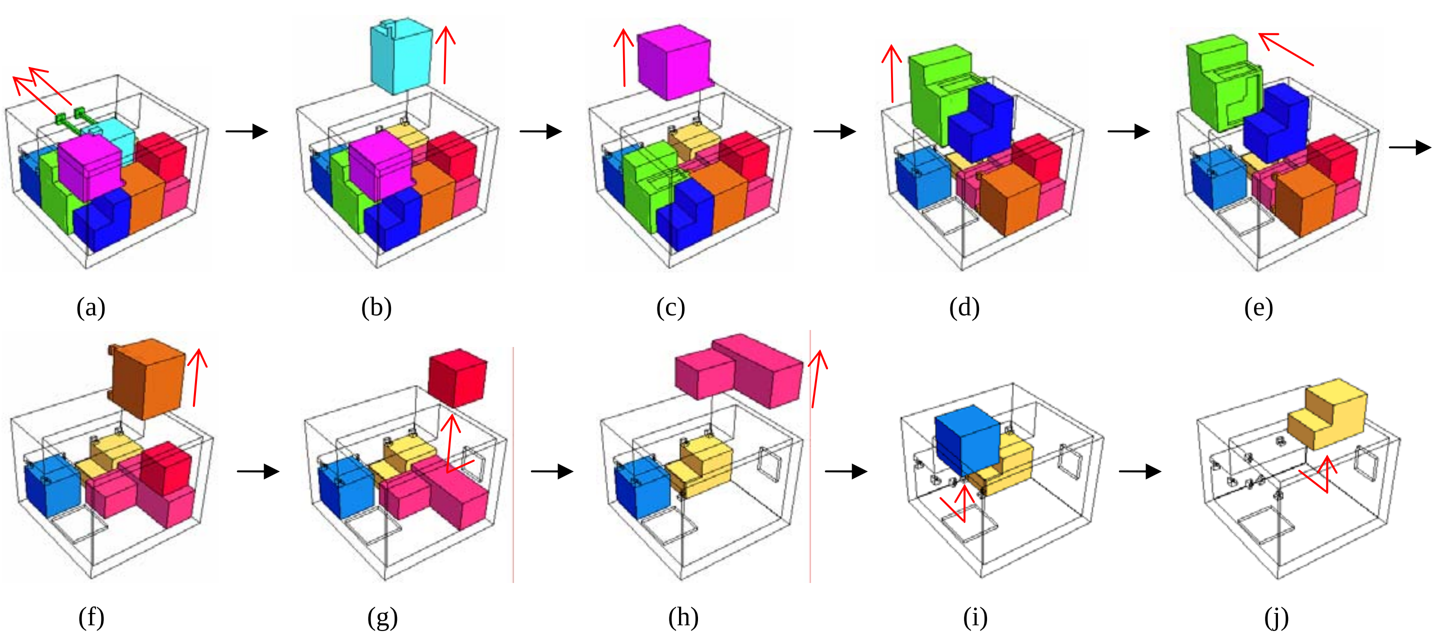


Figure 10. An optimal disassembly sequence of R_{13}.

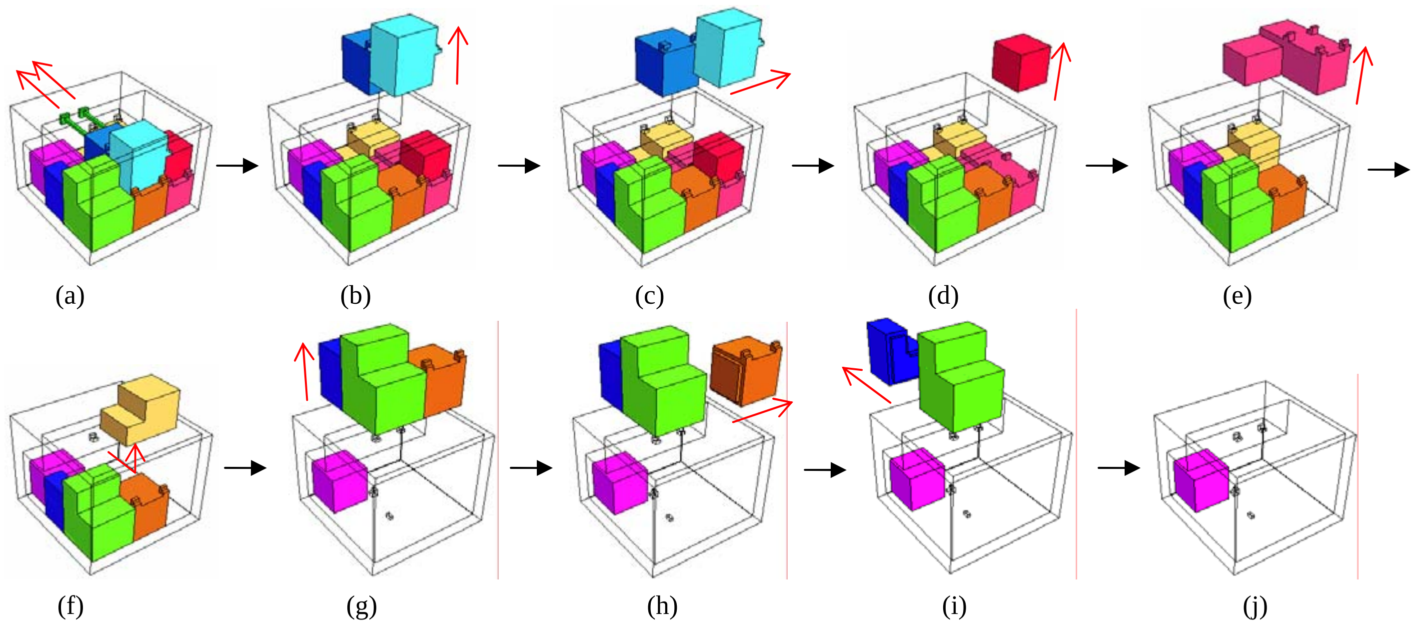


Figure 11. An optimal disassembly sequence of R_{23}

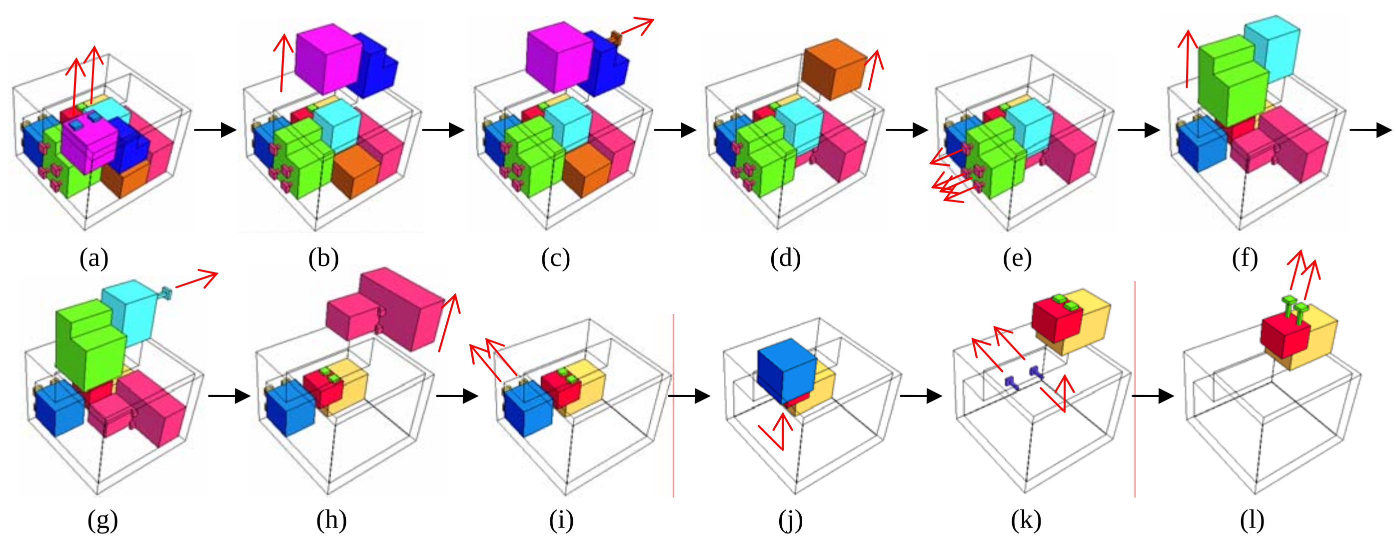


Figure 12. An optimal sequence of R_{11} ((a)-(j)) and of R_{21} ((a)-(l)).