Design for Product Embedded Disassembly Sequence

Shingo Takeuchi Kazuhiro Saitou

University of Michigan
Department of Mechanical Engineering
Ann Arbor, Michigan, 48109-2125, United States

Abstract

Due to the increased responsibility for the end-of-life (EOL) treatments of products by the manufacturers, the ease of disassembly has become a key design issue for mass-produced products. As an extension of the conventional design for disassembly (DFD), this paper presents a computational method for designing assemblies that can be disassembled only in specified sequences. Given an ideal disassembly sequence that maximizes the profit in the absence of geometric constraints, the method simultaneously determines the spatial configurations of components, locaters, and joints, such that disassembly can occur only in the sequence. A multi-objective genetic algorithm is utilized to search for Pareto-optimal designs in terms of the unique realization of the given disassembly sequence, the satisfaction of distance specifications among components, and the efficient use of locators on components. A case study on a simplified laptop computer assembly demonstrates the effectiveness of the method.

1 Introduction

The recent increase in the abandoned products prompted the regulatory (*eg.*, EU's WEEE directive) and voluntary initiatives for recycle and reuse around the world. Consequently manufacturers are becoming more responsible for the end-of-life (EOL) treatments of their products. Since both material recycling and component reuse typically requires the disassembly of products, Design for Disassembly (DFD) has become a key design issue in mass-produced, assembled products, such as consumer electronics products.

The profitability of a disassembly process U can be determined by

$$U = \sum_i (R_i - C_i) \qquad (1)$$

where R_i is the revenue of the i-th disassembled components and C_i is disassembly cost of i-th disassembly operation. Although R_i depends only on the disassembled components, C_i generally depends on both the disassembled components and the spatial configuration among components.

For modern consumer electronics products assembled predominantly in z-(vertical) direction, however, the cost of disassembling a component on top of an assembly (hence easily accessible from above) depends mainly on the number of fasteners [1]. If the arbitrary spatial arrangements of components are possible with minimum use of fasteners, therefore, an "ideal" disassembly sequence that maximizes the profitability U can be obtained by sorting the components in the descending order of its revenue until $R_i - C_i$ becomes non-negative. The corresponding component configuration is a simple "stack" of the disassembled components in the reverse order of disassembly.

Although this ideal disassembly sequence may not be realizable due to component geometry, distance specifications among components, and regulatory requirements for component/material removal, it is desired to design a product that disassembles in a similar sequence as possible. In order to reduce wrong operations during the disassembly process, it is further desired that the product (non-destructively) disassembles only in that sequence.

The above thoughts motivated us to develop a concept of product embedded disassembly sequence, where the relative motions of components are constraint mostly by the locator features integral to the components, such that only the desired disassembly sequence is possible. Figure 1 illustrates the concept. Upon the removal of the bolt that fixes component A and the container, components A, B, C can be disassembled *only* in a (desired) sequence <A, B> thanks to locator features 1 of component A and locator feature 2 on the container. Since components are "fixtured" by other components, the use of fasteners is minimized, satisfying the assumption to allow the a-priori specification of a desired disassembly sequence for maximum profitability. If a product allows only one disassembly sequence as illustrated in this figure, we call that the disassembly sequences is *embedded* in the product.

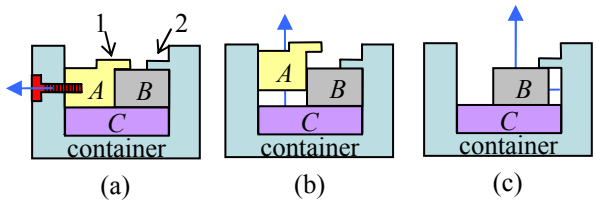

Figure 1: Concept of product embedded disassembly sequence. Integral locator features 1 and 2 allows only disassembly sequence of <A, B>.

This paper presents a computational method for designing assemblies with such embedded disassembly sequences. With an ideal disassembly sequence (IS) (assumed as given in this work) as an input, the method simultaneously determines the spatial configurations of components, locaters, and joints, such that disassembly can occur only in the sequence, while all components are completely fixed prior to disassembly. A multi-objective genetic algorithm (GA) [2,3] is utilized to search for Pareto-optimal designs in terms of the unique realization of the given disassembly sequence, the satisfaction of distance specifications among components, and the efficient use of locators on components. A case study on a simplified laptop computer assembly demonstrates the effectiveness of the method.

2 Related Work

2.1 Design for Disassembly

Design for disassembly (DFD) is a class of design method and guidelines to enhance the ease of disassembly for product maintenance and/or EOL treatments. Many researchers proposed the general DFD guidelines from the viewpoint of practical disassembly processes [1]. Reap *et al.* [4] reported DFD guidelines for the robotic semi-destructive disassembly, where detachable or breakable snap fits are preferred to screws due to their ease of disengagement. Matsui et al. [5] proposed a concept Products 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 a thermal stress to crack the glass upon the application of current.

While these works suggest redesigns to improve the ease of separation for individual joints, they do not address the issues of improving entire disassembly processes involving the removal of multiple joints and components.

2.2 Disassembly Sequence Planning

Disassembly sequence planning (DSP) aims at generating the disassembly sequences that are feasible for a given assembly, where the feasibility of a disassembly sequence is checked by the existence of collision-free motions to disassemble each component in the sequence. Since the disassembly sequence generation problem is NP-complete, the past researches have focused on the efficient heuristic algorithms to approximately solve the problem. Based on a number of important research results on assembly sequence planning [6, 7], several automated disassembly sequence generation approaches for 2/2.5D components have been developed [8]. More recent works are geared towards DSP with special attention to reuse, recycling, remanufacturing and maintenance [9, 10].

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 accessibility of a component is heavily dependent on the spatial configuration of its surrounding components, this would seriously limit the opportunity for optimizing an entire assembly. In addition, these works do not address the design of locator and joint configurations, which also have profound impact on the feasibility and quality of a disassembly sequence.

2.3 Configuration Design Problem

While rarely discussed in the context of disassembly, the design of spatial configuration of given shapes has been an active research area by itself. Among the most popular flavors is bin packing problem (BPP), where the total volume (or area for 2D problem) the configuration occupies is to be minimized. Since this problem is also NP-complete, heuristic methods are commonly used. Fujita *et al*. [11] proposed hybrid approaches for 2D plant layout problem, where the topology and geometry of a layout are determined by simulated annealing (SA) [12] and generalized reduced gradient (GRG) method, respectively. Corcoran *et al*. [13] solved a 3D packing problem with GA using multiple crossover methods. Jain *et al*. [14] adopted discrete representation as the object expression and proposed a geometry-based crossover operation for 2D packing problem. Grignon *et al*. [15] proposed a configuration design optimization method by using multi-objective GA, where static and dynamic balance and maintainability considered in addition to configuration volume.

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

3 Design for Product Embedded Disassembly Squence

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

- **Given:** component geometries, ideal disassembly sequence (IS), locator library (LL), distance specifications among components (DS)
- **Find:** component configuration, locator configuration on each component, disassembly motion of each component
- **Subject to:** no floating component, no over-lap among components, no unfixed component prior to disassembly, adjacency of components with interlocking locators
- **Minimizing:** non-unique realization of IS, violation of DS, redundant use of locator features

Since the problem has three objectives and the design variables (component and locator configuration) are discrete a multi-objective genetic algorithm is utilized to obtain Pareto optimal solutions. The following sections will describe the method in detail.

3.1 Inputs

Discrete geometry representations, such as voxels and octrees, have been successfully applied to DSP [16, 17] and BPP [12, 14] due to their efficiency in checking contacts and intersections and simplicity in modifying component geometries. For this reason, this work also adopts voxel representation, where CAD inputs of component geometries are first voxelized using ACIS® solid modeling kernel.

The ideal disassembly sequence (IS), and disassembly sequences in general, are represented as a linear sequence of subassemblies to be disassembled from an assembly. For example, disassembly sequence $< \{A, B\}, C, A>$ of the assembly $\{A, B, C, D\}$ represents the following series of disassembly operations:

1. subassembly $\{A, B\}$ from assembly $\{A, B, C, D\}$
2. component C from subassembly $\{C, D\}$
3. component A from subassembly $\{A, B\}$

As illustrated in figure 1, locators on components constrain their relative motions. Locator library is a set of locating features that can be potentially added on each component. Figure 2 shows the seven (7) locators in the locator library used in the following case study. Since the addition of a locator on a component alters the component geometry, some locators in figure 2 cannot co-exist in the same configuration relative to a component. For example, FaceRest, FaceSlot, and FaceTab cannot co-exist on the same face and edge, whereas FaceRest and Boss, or FaceRest and EdgeRest can co-exist. Based on the observation, the seven locators are classified to three types, Face (figures 2 (a) – (c)), Facefit (figure 2 (d)), and Edge (figures 2 (e) – (g)), to indicate the fact no locator of the same type can co-exist on the same face and edge of a component.

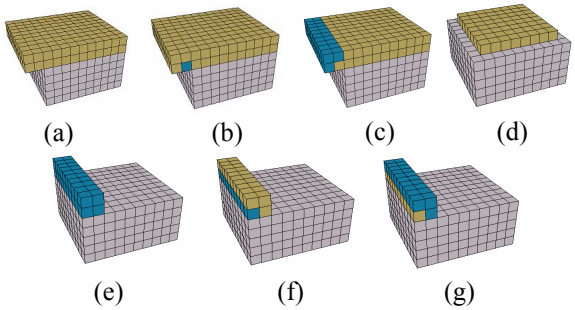

Figure 2: Locator library used in the case study: (a) FaceRest, (b) FaceSlot, (c) FaceTab, (d) Boss, (e) EdgeRest, (f) EdgeSlot, and (g) EdgeTab.

Distances or adjacency among components are often constrained by their functional relationship. For example, the cooling fan should be positioned near the CPU in the component configuration of a laptop computer. Since the distance between some pairs of components are more important than the others, the distance specification (DS) is a set of the weights of importance for the distances between two components (measured between two designated voxels) that needs to be minimized. If the weight between to two components is zero, the distance between the two components is considered as unimportant and can be arbitrary chosen. Figure 3 shows an example of the distance specifications among three components.

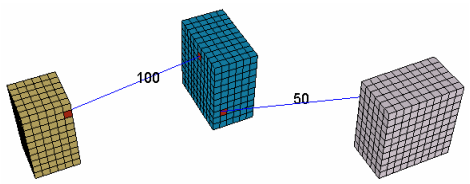

Figure 3: Example of distance specification (DS). The labeled lines between two voxels indicate distance specifications.

3.2 Design variables

There are two design variables for the problem. The first variable, *configuration vector,* is a vector of the translations of components relative to the global reference frame:

$$\boldsymbol{x} = (x_0, y_0, z_0, x_1, y_1, z_1, \ldots, x_{n-1}, y_{n-1}, z_{n-1}) \qquad (1)$$

where n is the number of components in the assembly, and x_i, y_i, and z_i, ($i = 0, 1, \ldots n$-1) are the translation of the i-th component in x-, y-, and z-directions, respectively. Note that no rotational motions are considered in the present work.

The second variable, *locator vector*, is a vector of the locator id# of each type in LL, for each pair of a face and its adjacent edge for components:

$$\boldsymbol{y} = (fa_0, ff_0, ed_0, fa_1, ff_1, ed_1 \ldots, fa_{m-1}, ff_{m-1}, ed_{m-1}) \qquad (2)$$

where m is the number of potential locator positions in the assembly (pairs of a face and its adjacent edge in all components), and fa_i, ff_i, and ed_i ($i = 0, 1, \ldots m$-1) are the locator id#'s of type Face, Facefit, and Edge in LL, respectively.

Variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are simply concatenated to form a chromosome in multi-objective genetic algorithm used to solve the optimization problem. Since the information in $\boldsymbol{x}$, $\boldsymbol{y}$ are linked to the geometry of a candidate design, the conventional one point or multiple point crossover for linear chromosomes are ineffective in preserving high-quality building blocks. Accordingly, a geometry-based

crossover operation [14] is adopted in the proposed method.

3.3 Constraints

The locations of components as specified by $\boldsymbol{x}$, whose geometries are altered by adding the locators as specified by $\boldsymbol{y}$, must satisfy the following four constraints:

- No floating component
- No over-lap among components
- No unfixed component prior to disassembly
- Adjacency of components with interlocking locators.

The last constraint is necessary since locators FaceSlot, FaceTab, Boss, EdgeRest, EdgeSlot and EdgeTab require an adjacent component with interlocking features, which is not specified by $\boldsymbol{y}$. If a component with these locators lacks an adjacent component to which the interlocking feature should be added, the configuration is considered as infeasible. Figure 4 illustrates an example.

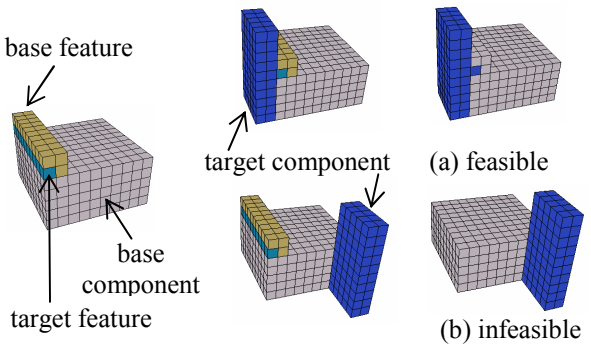

Figure 4: An example of the feasibility of interlocking locators: (a) feasible, and (b) infeasible.

3.4 Objective Functions

The configurations of components and locators on each component as specified by $\boldsymbol{x}$ and $\boldsymbol{y}$ are evaluated according to three criteria: 1) unique realization (*i.e.,* embedment) of IS, 2) satisfaction of DS, and 3) efficient use of locator features.

The first objective function (to be minimized) is for the embedment of IS defined as:

$$f_1(\boldsymbol{x},\boldsymbol{y}) = \mathsf{invalidity}(\boldsymbol{x},\boldsymbol{y}) + \mathsf{non\_uniqueness}(\boldsymbol{x},\boldsymbol{y}) \quad (3)$$

where the first term returns the degree to which IS is invalid, and the second term returns the degree to which the disassembly sequences other than IS is allowed.

Function $\mathsf{invalidity}(\boldsymbol{x}, \boldsymbol{y})$ is the number of infeasible disassembly operations in IS. The infeasibility of disassembly operation of subassemblies s and t is checked by 2-disassembleability criterion (if s can be disassembled within two consecutive motions) as follows [16]:

1. For each mating surfaces between s and t (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 s and t as a union of all constrained directions obtained in step 1.
3. If $D \backslash CD_{st} = \emptyset$, return *infeasible*.
4. If there exist a direction in $D \backslash CD_{st}$ along which s can be moved infinitely without a collision, return feasible (s 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 s by unit length along the direction. If s collides with other components, go to step 5.
7. If s is 1-disassembleable at the current location, return feasible (s is 2-disassembleable). Otherwise, go to step 6.

Checking for the non-uniqueness of IS by directly searching for all other feasible disassembly sequences will be prohibitively expensive to compute. Therefore, function $\mathsf{non\_uniqueness}(\boldsymbol{x}, \boldsymbol{y})$ approximate this as the number of directions in which subassemblies can be locally removed in a different manner from IS:

1. $n = 0$.
2. Select subassembly s in IS. If all have been selected, return n.
3. Select subassembly $t \neq s$ from the parent assembly r of s. If all have been selected, go to step 2.
4. $n = n + |D \backslash CD_{st}|$.
5. Go to step 3.

The second objective function (to be minimized) is for the satisfaction of DS, given as:

$$f_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_i w_i d_i \quad (4)$$

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

Finally, the third 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 locators to components:

$$f_3(\boldsymbol{x}, \boldsymbol{y}) = \sum_i c_i \quad (5)$$

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

4 Case Study

The proposed method is applied to a simplified laptop computer assembly composed of 10 components. DS is shown in figure 5 (a), where components *A*, *D* and *E* are fixed. IS is given as <*B*, *C*, *F*, *G*, *H*, *I*, *J*>, and the corresponding disassembly tree with the order of disassembly operations is shown in figure 5 (b), where each disassembly operation disassembles one component. LL in figure 3 is used.

Figure 6 (a) shows the 115 Pareto optimal solutions for f_1, f_2, and f_3, obtained by a muiti-objective genetic algorithm [2,3] with population of 250 and at generation 800. These numbers are chosen so that GA runs return with consistent results. Figures 7 (a)-(c) show the three solutions with $f_1 = 0$ (*i.e.*, IS is perfectly embedded) in figure 6 (b). These solutions indicate trade-offs between the satisfaction of DS and the efficient use of locators. For example, compared with solution 2 (figure 7 (b)), solution 3 (figure 7 (c)) uses less locators due to the utilization of the contacts between E and G, F and H, and G and H at the expense of the satisfaction of DS.

Figure 7 (d) illustrates the disassembly process of the solution 2. It can be seen that the assembly can only be disassembled in IS.

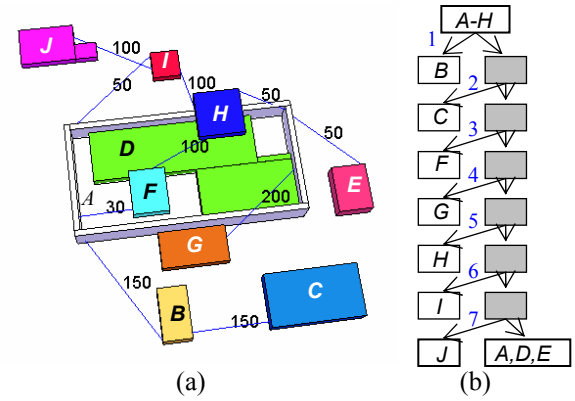

Figure 5: Simplified laptop computer assembly. (a) component geometries and DS, and (b) the disassembly tree corresponding to IS.

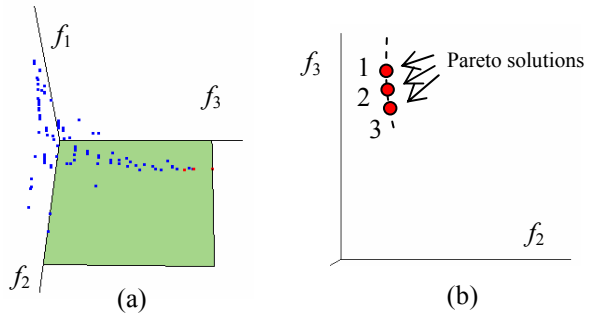

Figure 6: Pareto optimal solutions of the case study. (a) all 115 solutions, and (b) three solutions with $f_1 = 0$.

5 Conclusion

This paper presented a computational method for designing assemblies that could be disassembled only in specified sequences to accomplish easy EOL disassembly. A multi-objective genetic algorithm was utilized to search for Pareto-optimal designs in terms of the unique realization of the given disassembly sequence, the satisfaction of distance specifications among components, and the efficient use of locators on components. A case study on a simplified laptop computer assembly demonstrates the effectiveness of the method. While cannot be used to obtain the final design due to a number of other design factors, the proposed method would provide early insights on designers during the conceptual design stages.

The future work will address the discovery of ideal disassembly sequence as an outcome of maximizing the profit of the disassembly process.

Acknowledgement

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.

Reference

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

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

[3] C. A. Coello, D. A. Veldhuizen, and G. B. Lamont, "Evolutionary algorithms for solving multi-objective optimization problems", *volume 5 of Book Series on Genetic Algorithms and Evolutionary Computation*, 2002, Kluwer Academic Publishers, ISBN-0306467623.

[4] J. Reap and B. Bras, "Design for Disassembly and The Value of Robotic Semi-Destructive Disassembly," *Proceedings of DETC'02, ASME 2002 Design Engineering Technical Conferences, And Computers and Information in Engineering Conferences,* September 29 – October 2, 2002, Montreal, Canada

[5] K. Matsui, K. Mizuhara, K. Ishii, and R. M. Catherine, "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, 1999, pp. 570-575

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

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

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

[9] K. Lee and R. Gadh, "Destructive Disassembly to Support Virtual Prototyping," *Journal of Design and Manufacturing, IIE,* 1996

[10] A. J. D. Lambert, "Optimal Disassembly of Complex Products," *Journal of Technovation*, 1997, vol. 35, No. 9, pp. 2509-2523

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

[12] A. Kolli, J. Cagan and R. Rutenbar, "Packing of Generic, Three-Dimensional Components Based on Multi-Resolution Modeling," *Proceedings of DETC96, 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference,* August 18-22, 1996, Irvine, California.

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

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

[15] P. M. Grignon and G. M. Fadel, "Configuration Design Optimization Method," *Proceedings of DETC99, 1999 ASME Design Engineering Technical Conferences,* September 12-15, 1999, Las Vegas, Nevada.

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

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

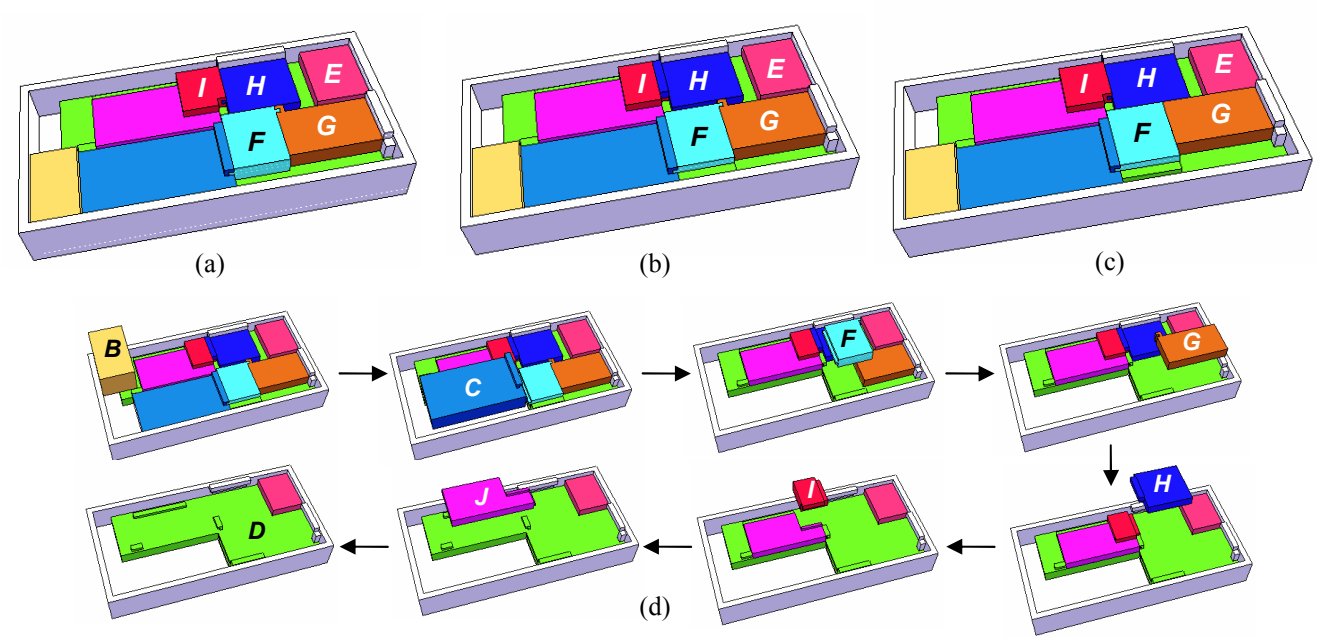

Figure 7: Three Pareto solutions with $f_1 = 0$ shown in figure 6 (b). (a) solution 1, (b) solution 2, (c) solution 3, and (d) disassembly process of solution 2. Solutions 1-3 differ at the interfaces among E, F, G, H, and I.