Proceedings of DETC'03
ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference
Chicago, Illinois, September 2-6, 2003

DETC2003/DAC-48729

ASSEMBLY SYNTHESIS WITH SUBASSEBLY PARTITIONING FOR OPTIMAL IN-PROCESS DIMENSIONAL ADJUSTABILITY

Byungwoo Lee and **Kazuhiro Saitou***
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan 48109-2125
Email: {byungwoo, kazu}@engin.umich.edu

ABSTRACT

Achieving the dimensional integrity for a complex structural assembly is a demanding task due to the manufacturing variations of parts and the tolerance relationship between them. While assigning tight tolerances to all parts would solve the problem, an economical solution is taking advantage of small motions that joints allow, such that critical dimensions are adjusted *during* assembly processes. This paper presents a systematic method that decomposes product geometry at an early stage of design, selects joint types, and generates subassembly partitioning to achieve the adjustment of the critical dimensions during assembly processes. A genetic algorithm (GA) generates candidate assemblies based on a joint library specific for an application domain. Each candidate assembly is evaluated by an internal optimization routine that computes the subassembly partitioning for optimal in-process adjustability, by solving an equivalent minimum cut problem on weighted graphs. A case study on a 3D automotive space frame with the accompanying joint library is presented.

Keywords: Design for assembly, design optimization, computer-aided design, assembly synthesis.

INTRODUCTION

Structural enclosures of modern mechanical products, such as ships hulls, airplanes and automotive bodies, and cases of some electronic devices are fairly complex, hence it is very expensive, if not impossible, to manufacture them from a single piece of material. Designers therefore have to decompose a complex enclosure into parts such as panels and beams, so that each part could be manufactured with a reasonable cost while satisfying the functional requirements of the assembled enclosure.

As the number of parts increases, however, achieving the dimensional integrity of the final assembly becomes more demanding task due to the inherent variations in manufacturing and assembly operations. For the body structures or frames in which parts are typically forged or bent, it is not economical to manufacture every part with tight tolerance to meet the required dimensional integrity of the final products. For this type of assemblies, it is typical that exact locations of joints (such as welds and fasteners) are not specified in the part design. Instead, it is determined during the assembly operations, when parts are located and fully constrained in fixtures. For this *in-process dimensional adjustment* to work, the mating surfaces to be joined should allow a small amount of relative motion, which is why those mating surfaces are called *slip planes*.

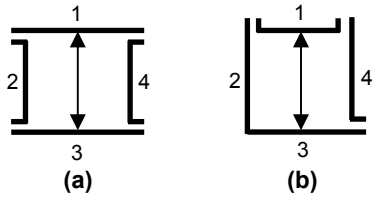

Figure 1. The design in (b) provides adjustability along the critical dimension, while one in (a) lacks proper slip planes.

To achieve a desired dimensional integrity of the final assembly, therefore, a designer must determine not only the decomposition of a product but also the orientations of the slip planes, which, in combination, would provide the optimal in-process adjustability of critical dimensions. See Figure 1 for two example designs of a rectangular box. Suppose the distance between section 1 and 3 is critical and parts are assembled on a fixture. Then, it is obvious that the design in Figure 1 (a) is not proper due to its lack of adjustability along the critical

* Corresponding author

dimension. On the other hand, the design in Figure 1 (b) provides slip planes such that the relative location of parts can be adjusted along the critical dimension. Although slip planes that are not completely parallel to a critical dimension might allow small motions along the critical dimension, they would yield a motion in other directions as well, which is not desirable.

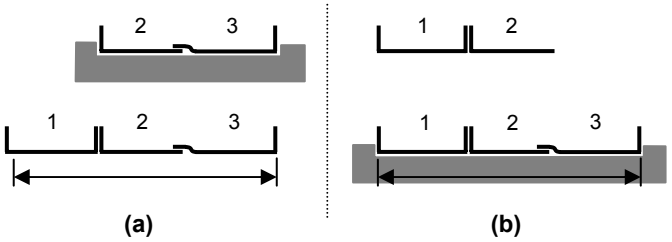

Figure 2. Two assembly sequences for automobile floor pan design (modified from [1]), where total length is critical. (a) poor (cannot adjust total length) and (b) better (can adjust total length).

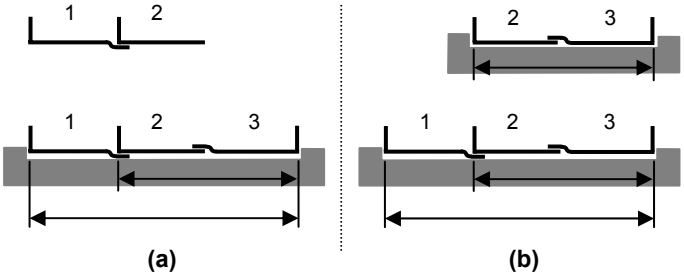

Figure 3. Two assembly sequences (modified from [1]) with two overlapping critical dimensions. (a) poor (two critical dimensions adjusted together at an assembly step) and (b) better (each critical dimensions independently adjusted at each assembly step)

To make things more difficult, the assembly sequences also affect the achievement of critical dimensions as shown in Figure 2 (modified from [1]). In Figure 2 (a), parts 2 and 3 are assembled first and then part 1 is put together. However, when part 1 is attached, there is no slip plane parallel to the critical dimension to absorb manufacturing variations that part {2,3} and 1 might carry. On the other hand, the sequence shown in Figure 2 (b) provides the slip plane at the assembly step the critical dimension is achieved, to absorb a variation in length. The Figure 3 (modified from [1]) shows another aspect of interrelation between in-process adjustability and assembly sequence. While the assembly sequence in Figure 3 (a) realized both of two critical dimensions at the second assembly step, the sequence in Figure 3 (b) does this one at a time: one critical dimension at the first assembly step, and another one at the second, thereby realizing an independent control of the amount of adjustment in each critical dimension.

Provided that the product has a large number of critical dimensions, decomposing the whole product into parts, configuring slip planes, defining datums, assigning tolerances and planning assembly sequences would be a very tedious task requiring several iterations. This motivated us to develop a systematic method which decomposes, selects joints types, and determines an assembly sequence to achieve the optimal in-process adjustability of critical dimensions (Figure 4).

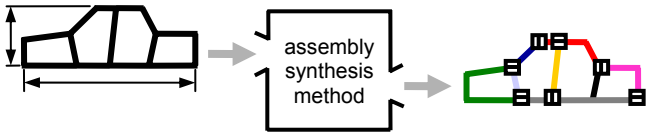

Figure 4. For a given product geometry, assembly synthesis method for in-process adjustability suggests desirable decompositions, joint configurations and assembly sequences.

Assembly synthesis has been defined by Yetis and Saitou [2] as the decomposition of the end product design prior to the detailed component design phase. In our previous work [3], we have presented an algorithm of assembly synthesis focused on the in-process adjustability and non-forced fit. The algorithm enumerates *all* possible assembly syntheses with the accompanying assembly sequences which, in combination, achieve dimensional adjustability for critical dimensions and non-forced fit between parts. It recursively decomposes a product from its final shape into parts and assigns joint configurations according to simple rules drawn from a related literature on assembly design [1]. Based on the AND/OR graph for assembly sequence planning [4], an augmented AND/OR graph has been devised to represent the results.

In [3], slip planes are allowed to take arbitrary angles required for a desired dimensional adjustability. However, only a few joint types (such as lap, butt, and lap-butt) are available in practice, which may results in slip planes that are not completely parallel to the critical dimensions. As a result, there would be degrees of adjustability (as opposed to simply adjustable or not adjustable) in the results of assembly synthesis, among which the best one should be selected via an optimization process. In addition, the incorporation of the other engineering criteria such as structural stiffness and part manufacturability would increase the computational time, hence rendering an enumeration-based approach impractical without an aid of an optimization process.

As an alternative to an enumeration-based approach in [3], this paper presents an optimization-based method of assembly synthesis for in-process dimensional adjustability using a genetic algorithm (GA), and its application to a 3D automotive space frame. A genetic algorithm generates candidate assemblies based on a joint library specific for an application domain. Each candidate assembly is evaluated by an internal optimization routine that computes the subassembly partitioning for optimal in-process adjustability, by solving an equivalent minimum cut problem on weighted graphs.

RELATED WORKS AND BACKGROUNDS

Assembly model and the Key Characteristic

A graph-based representation of mechanical assemblies is first appeared in [5] as the "graphe de liaisons fonctionelles," where a node and an edge represent each part in an assembly and each physical contact that a pair of parts have between them, respectively. Since then, the representation, typically referred to as the liaison diagram (or liaison graph) after De Fazio and Whitney [6], has been adopted in virtually all research in assembly modeling.

The Key Characteristic (KC) has been defined by Lee and Thornton [7] as product features, manufacturing process parameters, and assembly features that significantly affect a product's performance, function, and form. To facilitate the realization (or delivery, as often called) of geometric KCs in complex assemblies, Mantripragada and Whitney [8] devised an augmented liaison diagram called Datum Flow Chain (DFC). The DFC is an efficient tool to analyze how geometric KCs are delivered through datum relationships (represented as directed edges in DFC) among parts and the degrees of freedom joints carry. The critical dimensions mentioned earlier will simply be referred to as KCs in the rest of the paper.

Assembly sequence generation

Bourjault [5] and De Fazio and Whitney [6] are among the earliest researchers in assembly sequence generation. In both works, a user is required to answer a series of questions designed for systematically building the precedence relationships among the liaisons in an assembly, based on which all feasible assembly sequences can be enumerated. Independently, Homem de Mello and Sanderson [4, 9, 10] developed an AND/OR graph representation of assembly sequences and presented the first correct and complete algorithm to enumerate all feasible assembly sequences. Under the assumption that an assembly sequence is the reverse of a disassembly sequence, the algorithm generate an AND/OR graph in top-down fashion by recursively partitioning the liaison diagram of an assembly. Since then numerous research efforts had devoted to extend these pioneer works.

Assembly design via product decomposition

The aforementioned works on assembly sequence generation and their extensions focus on sequencing the assembly operations of a fixed set of physically separate parts. On the other hand, Wang *et al.* [11, 12] presented a method for decomposing a unfolded 3D geometry of a sheet metal product into parts for improved manufacturability. Their method first unfolds a 3D product geometry by searching spanning trees of the face-adjacency graph, and then decomposes the unfolded product into several parts by enumerating cut-sets on the spanning trees.

Independently, we have developed a method termed *decomposition-based assembly synthesis*, for decomposing a given geometry of a structural product into parts for minimum reductions in structural strength [2] and stiffness [13], and for maximum component modularity [14,15], where a graph representing the topology of a product is directly decomposed for respective criteria, using genetic algorithms.

Effect of joint configuration on dimensional variation

Recently, a few researchers have pointed out the effect of joint configurations on the dimensional variations of sheet metal assemblies. Noting the flexibility of sheet metal assemblies, Liu and Hu [16,] utilized engineering structural model combined with statistical analysis to simulate the dimensional variations of a simple rectangular box constructed from three basic joint types: lap, butt and lap-butt joints. They showed that variation characteristics greatly vary depending on the chosen joint type. Ceglarek and Shi [17] provided an analysis of joint configurations on their ability of absorbing part variations. Mantripragada and Whitney [18] presented a state transition model of an assembly sequence, where a joint is viewed as a control vector to absorb variations at each state. Although these works analyzes the effect of joint configurations on the dimensional variations of an assembly, none discusses the simultaneous design of product decompositions, joint configurations, and assembly sequences, as addressed in [3] and the present paper.

TERMINOLOGY

Since the assembly synthesis deals with objects yet to be decomposed into an assembly of separate parts, a few terms need to be defined to avoid confusion with generic meanings used in other literatures, including our own previous work:

- *A product geometry* is a geometric representation of a whole product as one piece (Figure 5 (a)).
- A *member* is a section of a product geometry allowed to be a separate part. A pair of members is *connected* when they meet at a certain point in the product geometry.
- A *configuration* is a group of members which are connected to at least one of the members within the group. A product geometry is a configuration, so as a part (as defined below).
- A configuration graph is a triple:

$$C = (M, T, A) \tag{1}$$

where M is the set of nodes representing members, T is the set of edges representing connections between a pair of members, and A is the set of edges representing KCs between a pair of members (Figure 5 (b)). In other words, a configuration graph is a "liaison diagram" of product geometry augmented with the information on KCs. Since a configuration is a group of connected members, a configuration graph excluding the edges in A is a connected graph[*].

[*] A graph G is termed *connected* if every pair of vertices (or nodes) in G are joined by a path [18].

- A *decomposition* is a transition of a configuration into two or more subconfigurations by removing connections between a pair of members.
- A *part* is a configuration that cannot be decomposed further under given criteria. An example of such criterion is a minimum part size. Note a part may consist of one or more members.
- A *joint library* is a set of joint types available for a specific application domain. In typical two-dimensional sheet metal assemblies, lap, butt and lap-butt joint would form a joint library (Figure 6).
- An (*synthesized*) *assembly* is a set of parts and joints which connect every part in the set to at least one of other parts in the set. This is what conventionally referred to as an assembly in the literatures on assembly sequence generation.
- A liaison graph of a (synthesized) assembly is a triple:

$$L_0 = (V_0, E_0, A_0) \tag{2}$$

where V_0 is the set of nodes representing parts, E_0 is the set of edges representing joints and A_0 is the set of edges representing KCs. The A_0 takes over all the KCs from the A (equation (1)), but connecting the nodes in V_0 that are hyper-nodes of the nodes in M (since a part can consists of one or more nodes). A liaison graph of assembly L_0 is often simply referred to as an assembly if obvious from the context.

- *Assembly synthesis* is a transformation of a product geometry into an assembly. In other words, a configuration graph C of a product geometry is transformed to a liaison graph of an assembly L_0, via assembly synthesis.
- *Subassembly partitioning* is (the process of) building a binary tree which represents a (partial) assembly sequence of a synthesized assembly.

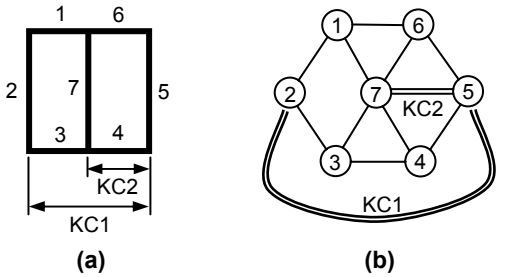

Figure 5. A product geometry and its configuration graph.

Figure 6. An example of joint library for two-dimensional sheet metal products.

OPTIMAL ASSEMBLY SYNTHESIS FOR IN-PROCESS DIMENSIONAL ADJUSTABILITY

Overview

Figure 7 depicts an overview of the proposed method of assembly synthesis for in-process dimensional adjustability. It takes as inputs a product geometry with KCs and a joint library, and produces an assembly for the product and an accompanying subassembly partitioning that optimally achieve in-process adjustability of the input KCs. The optimization is done in nested double loops with a genetic algorithm (outer loop) that generates candidate assemblies, and a subsidiary optimizer (inner loop) that estimates the optimal in-process adjustability of the candidate assembly and the accompanying subassembly partitioning. In the following sections, the method will be described in detail with the two-dimensional skeleton of product geometry shown in Figure 5.

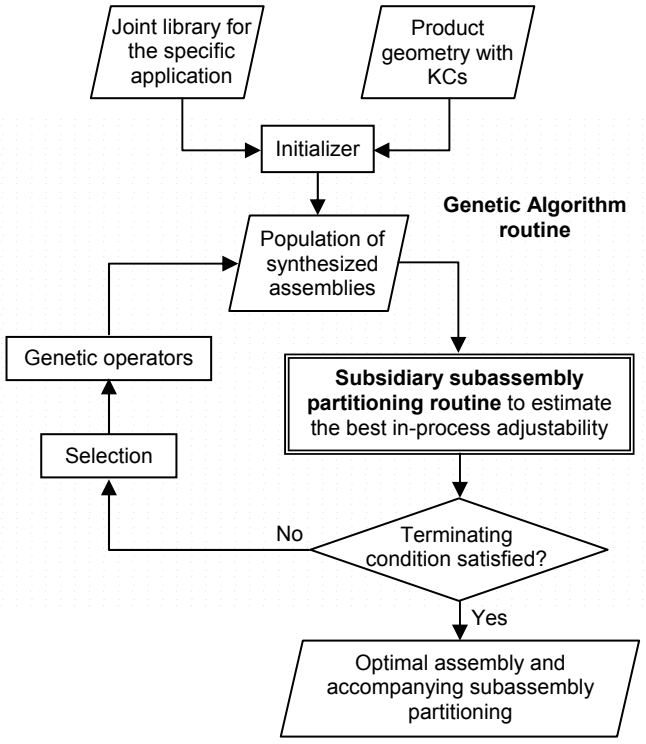

Figure 7. Overview of proposed method.

Assembly synthesis with joint library

Using the terminology defined above, the problem of optimal assembly synthesis for in-process adjustability is formulated as follows:

given: $C = (M, T, A)$
find: $T \mapsto JL$ (3)
satisfying: manufacturability constraints
that maximizes: *adjustability*(L_0)

where C is the configuration graph of a (given) product geometry, JL is a (given) joint library, L_0 is the liaison graph of the assembly resulting from the joint assignment $T \mapsto JL$ onto C, and *adjustability* is the subsidiary routine ("inner" loop) for subassembly partitioning, which evaluates the dimensional adjustability of L_0. Note that the formulation assumes JL includes "connected" to represent no joints. For example, the joint library in Figure 6 is JL = {connected, L, B, LB, BL}.

The manufacturability constraints in formulation (3) depend on the application domain. An example adopted in the following case study is that the prohibition of KCs within a part, since part tolerances are assumed to be much lower than the tolerance required to achieve KCs. Another example is the decomposition types in Figure 8, which lists typical two- and three member connections for sheet metal parts or extruded frames. For each decomposition type in Figure 8, a dotted line represents a joint of a type in a given joint library. Note two-member decompositions have either zero or one joint, whereas three-member decompositions have one or two joint(s). There are no three-member decompositions with zero joint, since such shape are not usually manufactured with sheet metals and extruded space frames.

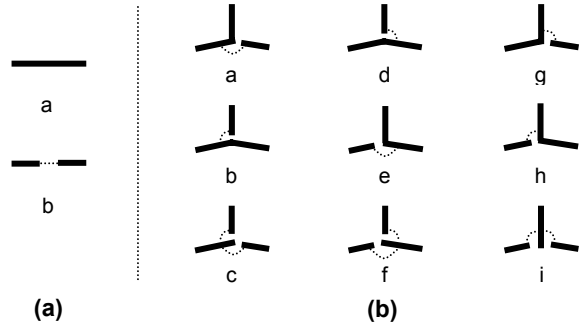

Figure 8. Decomposition types for (a) two- and (b) three-member connections. A dotted line represents a joint which will be assigned between members that it is attached to.

In the "outer" loop in Figure 7, a genetic algorithm routine generates a population of candidate assemblies (more precisely candidate mappings $T \mapsto JL$), each of which is represented by a decomposition type and a joint type for every connection point (intersections of members) of the product geometry. A chromosome, an internal representation of design variables for GA, is a vector $\boldsymbol{c} = (c_1, c_2, .., c_n)$ of n components, where n is the number of connection points in the product geometry. Each component c_i in $\boldsymbol{c}$ is in turn a vector of three components:

$$c_i = (d_i, j_{i1}, j_{i2}) \tag{4}$$

where d_i is a decomposition type and j_{i1} and j_{i2} are joint types. As specified in Figure 8, d_i is either a or b if connection point i is a two-member connection, whereas d_i can take any values in $\{a, \ldots, i\}$ if connection point i is a three-member connection. While j_{i1} and j_{i2} can take any values specified in the joint library, the value of j_{i2} is ignored if d_i indicates the decomposition types with only one joint, and the values of both j_{i1} and j_{i2} are ignored if d_i indicates the decomposition types with only zero joint.

For example, Figure 9 (a) shows a chromosome for the product geometry in Figure 5, where each column represents a component for a connection point denoted by the indices of connecting members. Joint types are chosen from the joint library in Figure 6. Figures 9 (b) and (c) illustrate the assembly represented by the chromosome and the corresponding configuration graphs, respectively. For the evaluation of dimensional adjustability, the configuration graphs in Figure 9 (c) are transformed to a compact liaison graph of the assembly depicted in Figure 10, where each rectangular node represents a part which consists of one or more members.

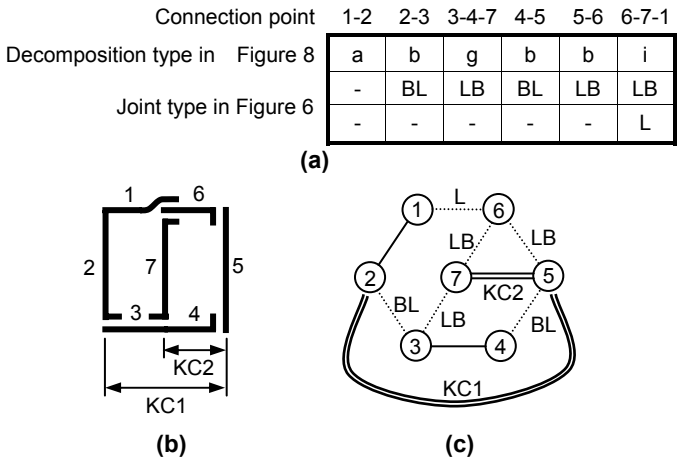

Figure 9. (a) A chromosome for the product geometry shown in Figure 5, (b) represented assembly, and (c) corresponding configuration graphs with dotted lines representing joints.

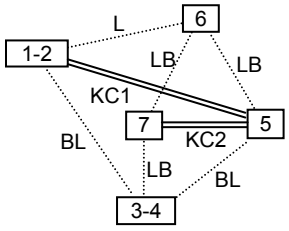

Figure 10. Liaison graph of the assembly in Figure 9 (c). The members in a configuration are combined into a node, which represents a part in the assembly.

OPTIMAL SUBASSEMBLY PARTITIONING

Binary tree of subassembly partitioning

Each candidate assembly L_0 generated by GA is evaluated for dimensional adjustability by the subsidiary routine ("inner" loop) for subassembly partitioning, represented as function *adjustability* in Equation (3). A subassembly partitioning of a given assembly L_0 can be represented as a binary tree [4,20]:

$$BT = (S, P) \tag{5}$$

where S is a set of subsets V of nodes V_0 representing parts in subassemblies, and P is a set of hyper-edges representing binary partitioning (branching) in the binary tree. Suppose subassembly $L_i = (V_i, E_i, A_i)$ is partitioned to two subassemblies $L_j = (V_j, E_j, A_j)$ and $L_k = (V_k, E_k, A_k)$, the partitioning is represented as a hyper-edge $p = (V_i, (V_j, V_k))$.[†] For each hyper-edge $p \in P$, let us define a set of KC $KC_p \subseteq A_0$ "broken" by the partition as:

$$KC_p = A_i \setminus (A_j \cup A_k) \tag{6}$$

Similarly, we can define a *cut-set*[‡], $CS_p \subseteq E_0$ for a hyper-edge $p \in P$ as:

$$CS_p = E_i \setminus (E_j \cup E_k) \tag{7}$$

Viewing in a reverse ("bottom up") fashion, a hyper-edge (V_i, (V_j, V_k)) can be interpreted as an assembly process of V_j and V_k into V_i. Under this view, KC_p is a set of KCs realized by the assembly operation, and CS_p is a set of joints that need to be joined during the assembly of V_j and V_k.

For example, let us consider the liaison graph of assembly $L_0 = (V_0, E_0, A_0)$ in Figure 10, where V_0 = {1-2, 3-4, 5, 6, 7}, E_0 = {(1-2,3-4), (1-2, 6), (3-4, 7), (3-4, 5), (5, 6), (6, 7)}, and A_0 = {(1-2, 5), (5,7)}. If the liaison graph L_0 is partitioned into node 1-2 and the other nodes as depicted in Figure 11, the corresponding *BT* has three nodes V_0, V_1 = {1-2} and V_2 = {3-4, 5, 6, 7} in S, and a hyper-edge p = {(V_1, (V_2, V_3)) in P, of which KC_p = {(1-2, 5)} and CS_p = {(1-2, 3-4), (1-2, 6)}. A graphical representation of the *BT* is shown in Figure 12. This binary tree can be interpreted as an assembly sequence consisting on one assembly step: join a subassembly {1-2} with another {3-4, 5, 6, 7} to a final assembly {1-2,3-4, 5, 6, 7}. Since subassembly {3-4, 5, 6, 7} consists of multiple parts, it is of course possible to recursively partition the subassembly into smaller subassemblies, thereby "growing" the binary tree, as discussed later in the section on subassembly partitioning.

[†] A parenthesized pair of objects usually has an order between the objects. However, in this context, (V_b, V_c) represents a pair of child subassemblies of V_a without order. Thus, (V_b, V_c) and (V_c, V_b) are not distinguished in the rest of the paper.

[‡] A *cut-set* in a connected graph $G = (V, E)$, is a minimal set of edges of E whose removal from G, renders G disconnected [19]. In the liaison graph we do not count edges representing KC to a cut-set.

Problem formulation

As observed in Figure 2, it is desirable that a slip plane is provided at the very assembly operation where a KC is realized, and the slip plane is parallel to the KC. In the context of the binary tree of subassembly partitioning, this means at an assembly operation corresponding to a hyper-edge p, (the slip planes of) all joints in CS_p should be parallel to KC's in KC_p.

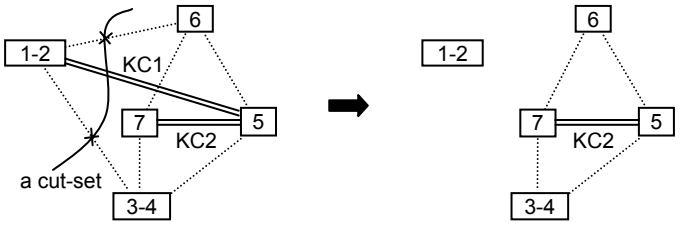

Figure 11. A subassembly partitioned to two subassemblies.

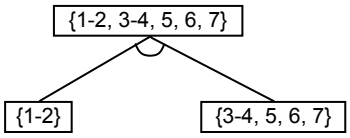

Figure 12. Binary tree of subassembly partitioning illustrated in Figure 11.

However, if KC_p contains more than one KC in different directions, making all joints in CS_p parallel to the KCs is obviously impossible. Therefore, KC_p should contain *only* one KC at every assembly operation corresponding to p. In other words, a partition p should not break more than one KC. This constraint should be kept even if KCs are in the same direction, as achieving more than one KC at a single assembly operation always requires a compromise as illustrated in Figure 3.

Even if KC_p contains only one KC, it may not be possible to make all joints in CS_p parallel to the KC, depending on the available joint types in the joint library. Therefore, the adjustability of a partition p can be defined as the degree to which joints in CS_p are close to the parallel to the KC:

$$\sum_{a \in KC_p} \sum_{e \in CS_p} \left| \mathbf{k}(a) \bullet \mathbf{n}(e) \right| \tag{8}$$

where a is the KC in KC_p, e is a joint in CS_p, and $\mathbf{k}(a)$ and $\mathbf{n}(e)$ are the direction vectors of a and e, respectively. Although the sum is taken over $a \in KC_p$, there should be only one KC in KC_p. The value of Equation (8) is zero when all joints are completely perpendicular to each other (*i.e.*, no adjustability), and a positive number otherwise. Similarly, the adjustability of a binary tree of subassembly partition BT = (S, P) can be defined as the sum of the adjustabilities of all partitions in P:

$$\sum_{p\in P}\sum_{a\in KC_p}\sum_{e\in CS_p}\left|\mathbf{k}(a)\bullet\mathbf{n}(e)\right| \tag{9}$$

The aim of the subsidiary routine *adjustability*(L_0) in Equation (3) is to find the *BT* that gives the maximum value of Equation (9) for a given assembly L_0, and return the maximum value to the "outer loop" GA optimizer. Using the aforementioned notations, the problem can be formulated as follows:

given: $L_0 = (V_0, E_0, A_0)$

find: $BT = (S, P)$

satisfying:

- $V_0 \in S$ (10.1)
- for $\forall V_i \in S$ satisfying $A_i \neq \varnothing$ in its liaison graph, $\exists V_j, V_k \in S$, $\exists p_i = (V_i, (V_j, V_k)) \in P$ such that
  - $V_j \neq \varnothing, V_k \neq \varnothing$ (10.2)
  - (V_i, E_i), (V_j, E_j) and (V_k, E_k) are connected (10.3)
  - $V_i = V_j \cup V_k$ (10.4)
  - $V_j \cap V_k = \varnothing$ (10.5)
  - $\left|KC_{p_i}\right| = 1$ (10.6)
- S and P are minimal sets satisfying (10.1) to (10.6) (10.7)

that maximizes:

$$\sum_{p_i\in P}\sum_{a\in KC_{p_i}}\sum_{e\in CS_{p_i}}\left|\mathbf{k}(a)\bullet\mathbf{n}(e)\right| \tag{11}$$

Constraints (10.1) – (10.6) are the definition of *BT*: Constrains (10.1) – (10.5) are necessary conditions for binary partitioning, and constraint (10.6) states KC_p should contain only one KC. Constraint (10.7) implies that the *BT* is a binary tree (only one out-going hyper-edge from a node) without any nodes or hyper-edges that do not satisfy (10.1) to (10.6).

Greedy approach for subassembly partitioning

The exact solution to the above optimal subassembly partitioning problem requires a complete AND/OR graph of L_0, which requires O(2^n) computations for the assembly with n number of parts [4,20]. Since the problem must be solved for each candidate assembly L_0 generated by the GA, it is not practical to spend a large amount of time to obtain the true optimal solution. As such, the problem is solved approximately by a greedy approach, where, at each depth of recursion, a subassembly is partitioned into two subassemblies by the cut that provides the maximum adjustability. Starting with the input assembly L_0, the recursion proceeds until the resulting subassemblies contain no KCs. In other words, for $\forall V_i \in S$ satisfying $A_i \neq \varnothing$ in its liaison graph, a partition p_i is given as:

$$\begin{aligned} p_i = (V_i, (V_j, V_k)) &= \arg\max_{q\in P_i}\left\{\sum_{a\in KC_q}\sum_{e\in CS_q}\left|\mathbf{k}(a)\bullet\mathbf{n}(e)\right|\right\} \\ &= \arg\max_{q\in P_i}\left\{\sum_{e\in CS_q}\left|\mathbf{k}(a_q)\bullet\mathbf{n}(e)\right|\right\} \end{aligned} \tag{12}$$

where P_i is a set of all hyper-edges partitioning of V_i satisfying constraints (10.2)-(10.6) and a_q is the only element in KC_q (as $|KC_q| = 1$ from (10.6)). Since $0 \le \left|\mathbf{k}(a_q)\bullet\mathbf{n}(e)\right| \le 1$, Equation (12) can be rewritten as:

$$p_i = \arg\min_{q\in P_i}\left\{\sum_{e\in CS_q}\left(1-\left|\mathbf{k}(a_q)\bullet\mathbf{n}(e)\right|\right)\right\} \tag{13}$$

Since the minimization in Equation (13) is over all binary partitioning of L_i that breaks only one KC, it is equivalent to the minimization over all cut-sets of L_i that cuts only one KC. This is in turn equivalent to the minimization over all cut-sets of L_i that cuts a KC in A_i, followed by the minimization over all KCs in A_i. Namely,

$$\begin{aligned} &\min_{q\in P_i}\left\{\sum_{e\in CS_q}\left(1-\left|\mathbf{k}(a_q)\bullet\mathbf{n}(e)\right|\right)\right\} \\ &\quad= \min_{a\in A_i}\left[\min_{CS\in CS_1(a)}\left\{\sum_{e\in CS}\left(1-\left|\mathbf{k}(a)\bullet\mathbf{n}(e)\right|\right)\right\}\right] \end{aligned} \tag{14}$$

where $CS_1(a)$ is a set of cut-sets of L_i that does not cut any KCs in A_i other than a. Let min-cut(a) and *MCS*(a) be the quantity inside of the square brackets in Equation 14 and its argument, respectively:

$$\text{min-cut}(a) \triangleq \min_{CS\in CS_1(a)}\left\{\sum_{e\in CS}\left(1-\left|\mathbf{k}(a)\bullet\mathbf{n}(e)\right|\right)\right\} \tag{15}$$

$$MCS(a) \triangleq \arg\min_{CS\in CS_1(a)}\left\{\sum_{e\in CS}\left(1-\left|\mathbf{k}(a)\bullet\mathbf{n}(e)\right|\right)\right\} \tag{16}$$

As implied by its name, min-cut(a) (and *MCS*(a) as a result) can be obtained by solving a well known minimum cut problem on graph $(V_i, E_i \cup A_i \setminus \{a\})$ with edge weight

$$w(e) = \begin{cases} 1-\left|\mathbf{k}(a)\bullet\mathbf{n}(e)\right| & \text{if } e \in E_i \\ \text{a large number} & \text{if } e \in A_i \setminus \{a\} \end{cases} \tag{17}$$

between two nodes incident on a. Since a large weight is assigned to KCs other than a, the solution of the minimum cut problem is the cut-set that only cuts a with the minimum weight, which is $MCS(a)$ in Equation 16. By solving the minimum cut problem for all $a \in A_i$, one can obtain the optimal cut-set $MCS(a_i^*)$ of L_i where

$$a_i^* = \arg\min_{a \in A_i} \left[\text{min-cut}(a) \right] \quad (18)$$

and finally p_i from $MCS(a_i^*)$. The following pseudocode outlines the overall algorithm.

greedy-subassembly-partitioning(L_0)
1. $S \leftarrow \{V_0\}$ and $P \leftarrow \varnothing$
2. greedy-partitioning(L_0, BT)
3. return BT.

greedy-partitioning(L_i, BT)
4. If $A_i = \varnothing$, return.
5. for each $a \in A_i$, solve the minimum cut problem as defined in Equation (17) to obtain the min-cut(a).
6. obtain a_i^* and the $MCS(a_i^*)$ from Equation (18).
7. obtain partition $p_i = (V_i, (V_j, V_k))$ using $MCS(a_i^*)$.
8. partition L_i into L_j and L_k using p_i.
9. $S \leftarrow S \cup \{V_j, V_k\}$ and $P \leftarrow P \cup \{p_i\}$.
10. greedy-partitioning(L_j, BT).
11. greedy-partitioning(L_k, BT).

At termination, greedy-subassembly-partitioning(L_0) returns the "fully grown" BT, with which the value of *adjustability*(L_0) in Equation 3 can be calculated as:

$$adjustability(L_0) = \sum_{p_i \in P} \sum_{e \in MCS(a_i^*)} \left(1 - \left|\mathbf{k}(a_i^*) \bullet \mathbf{n}(e)\right|\right) \quad (19)$$

Greedy-partitioning is a greedy algorithm that always takes the KCs with the highest adjustability of the current subassembly L_i without considering the subsequent partitionings. Although the algorithm does not provide the exact solution of the optimal subassembly partitioning problem, it can generate a high-quality solution at moderate computational efforts. It is observed that the problem seems suitable for a greedy strategy since the KC with very low adjustability at the early stages of partitioning (which the algorithm would not pick) tends to become highly adjustable in subsequent stages due to the smaller sizes of subassemblies.

While greedy-subassembly-partitioning could be used to obtain the optimal assembly sequence for an existing assembly, a dynamic programming algorithm is desired for this purpose since greedy-subassembly-partitioning sacrifices the solution accuracy for speed.

Optimal subassembly partitioning example

Let us consider again the liaison graph of assembly in Figure 10 and run greedy-subassembly-partitioning with it. Recall $L_0 = (V_0, E_0, A_0)$ where V_0 = {1-2, 3-4, 5, 6, 7}, E_0 = {(1-2,3-4), (1-2, 6), (3-4, 7), (3-4, 5), (5, 6), (6, 7)}, and A_0 = {(1-2, 5), (5,7)}. After initializing S with $\{V_0\}$ and P with an empty set at line 1, a subroutine greedy-partitioning is called at line 2 with L_0 and BT:

- **Line 4:** greedy-partitioning does not return since A_0 is not empty.
- **Line 5:** for two KCs in A_0, KC1 = (1-2, 5) and KC2 = (5, 7), the minimum cut problems are formulated based on the geometry in Figure 9 (b), as illustrated in Figures 13 (a) and (b). Note that a large number (100) is assigned to another KC in each figure. Other weights on edges are the same in both figures since the direction vectors of KC1 and KC2 are identical (see Figure 9 (b)). By inspection, min-cut(KC1) = min-cut(KC2) = 0 as illustrated also in Figures 13 (a) and (b).
- **Line 6:** Since both KCs have the same value of min-cut, either one can be chosen. Suppose we chose KC1. Then, a_0^* = KC1 = (1-2, 5) and $MCS(a_0^*)$ = {(1-2, 6), (1-2, 3-4)} from Figure 13 (a).
- **Line 7:** the corresponding hyper-edge is $p_0 = (V_0, (V_1, V_2))$ where V_1 = {1-2} and V_2 = {3-4, 5, 6, 7}.
- **Line 8:** L_0 is partitioned into L_1 and L_2, where V_1 = {1-2}, $E_1 = A_1 = \varnothing$, V_2 = {3-4, 5, 6, 7}, E_2 = {(3-4, 7), (3-4, 5), (5, 6), (6, 7)}, and A_2 = {(5,7)}.
- **Line 9:** BT is updated with $S = \{V_0, V_1, V_2\}$ and $P = \{p_0\}$.
- **Line 10:** greedy-partitioning recursively is called with L_1 and the updated BT.
- **Line 11:** after the return of the recursive call in line 10, greedy-partitioning recursively is called again with L_2 and BT (which might have been updates within the recursive call of line 10).

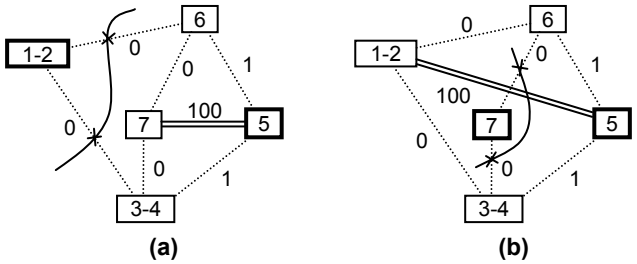

Figure 13. Solutions of minimum cut problems for (a) KC1 and (b) KC2 for the assembly in Figure 10. In each figure, the respective KC is incident on the nodes with thick lines.

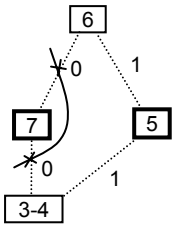

Figure 14. Solution of minimum cut problem for KC2 for the subassembly resulted from the partition in Figure 13 (a).

Since A_1 is empty, the recursive call of greedy-partitioning at line 10 returns immediately with no update to *BT*. The recursive call at line 11 proceeds as follows:

- **Line 4:** greedy-partitioning does not return since A_2 is not empty.
- **Line 5:** for KC2 = (5, 7) in A_2, the minimum cut problem is formulated as Equation 16. As illustrated in Figure 14, min-cut(KC2) = 0.
- **Line 6:** Since there is only one KC in A_2, a_2^* = KC2 = (5,7) and $MCS(a_2^*)$ = {(3-4, 7), (6, 7)} from Figure 14.
- **Line 7:** the corresponding hyper-edge is $p_2 = (V_2, (V_3, V_4))$ where V_3 = {3-4, 5, 6} and V_4 = {7}.
- **Line 8:** L_2 is partitioned into L_3 and L_4, where V_3 = {3-4, 5, 6}, E_3 = {(3-4, 5), (5, 6)}, $A_3 = \varnothing$, V_4 = {7}, and $E_4 = A_4 = \varnothing$.
- **Line 9:** *BT* is updated with $S = \{V_0, V_1, V_2, V_3, V4\}$ and $P = \{p_0, p_2\}$.
- **Line 10:** greedy-partitioning recursively is called with L_3 and the updated *BT*.
- **Line 11:** after the return of the recursive call in line 10, greedy-partitioning recursively is called again with L_4 and *BT* (which might have been updates within the recursive call of line 10).

Since both A_3 and A_4 are empty, the recursive calls at lines 10 and 11 return immediately with no updates to *BT*. At this point, the original call of greedy-partitioning(L_0, *BT*) returns, and greedy-subassembly-partitioning(L_0) terminates at line 3, by returning the "fully grown" *BT*.

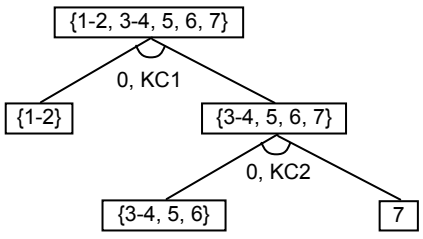

Figure 15. Subassembly partitioning for the assembly in Figure 10 by the greedy approach.

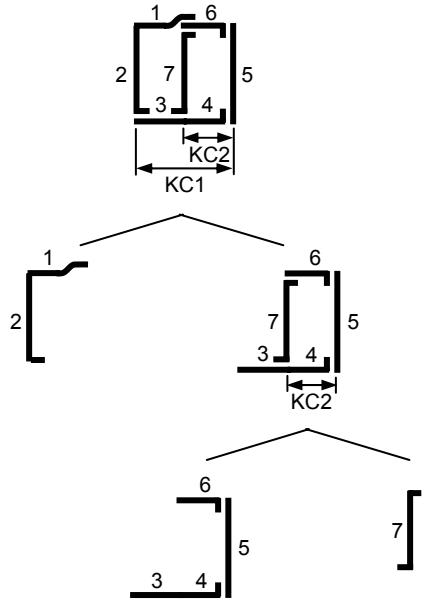

Figure 16. Subassembly partitioning in Figure 15 shown in part geometry.

Figure 15 shows a graphical representation of the resulting *BT*, where the broken KC and its cost at each partitioning are annotated under the corresponding hyper-edge. According to the equation (19), *adjustability*(L_0) = 0, which means the input assembly L_0 with the resulting *BT* achieves the perfect adjustability of all KCs. This is possible because the orientations of the members (Figure 9 (b)) and joint types in the library (Figure 6) can provide the joints exactly parallel to every KC, as illustrated in Figure 16.

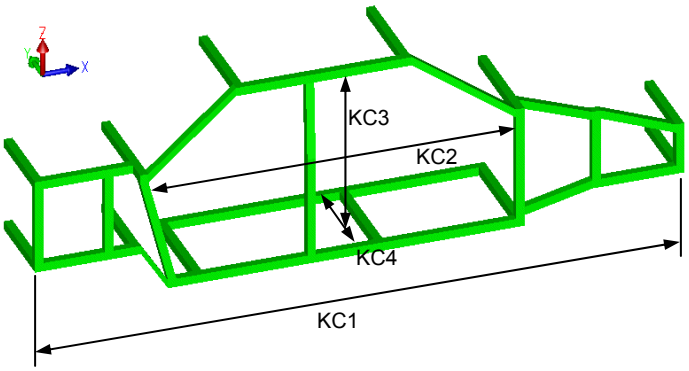

Figure 17. A 3D automotive space frame model with four KCs.

THREE-DIMENSIONAL AUTOMOTIVE SPACE FRAME

The proposed method is applied to a 3D model of an automotive space frame with four KCs, shown in Figure 17. It is assumed that the frame is made out of extruded aluminum members. Typical joint types for extruded aluminum members [21] are listed in Figure 18, where the joints with the same dimensional adjustability are regarded as a joint type in the

joint library. Depending on the orientations of the mating members, however, not all joint types in Figure 18 are feasible at every connection point in the frame. For example, the butt joint allowed for (nearly) perpendicular connections is not available for coaxial connections. For joints with plane adjustability such as the simple lap joints and butt joints, the normal vector of the plane has been used to compute the inner product with KC in Equation 17, without subtracting it from 1, such that the cost assigned to those edges is still 0 when the KC is parallel to the plane of adjustability.

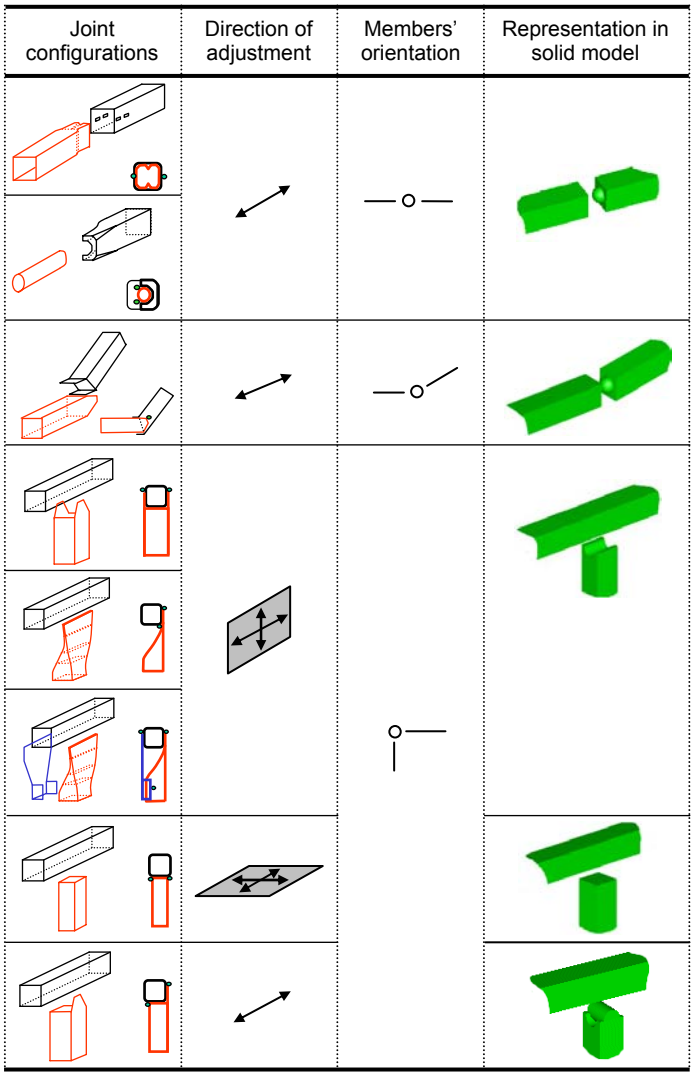

Figure 18. A joint library for automotive space frame [21].

Because the number of potential joint locations is much larger than the number of of KCs in the model, achieving perfect adjustability for all KCs is not very difficult. Therefore, the following two criteria are considered as the objectives to minimize, in addition to the maximization of dimensional adjustability for all KCs:

1. Total number of parts: as a smaller number of parts would reduce handling and assembly efforts.
2. Sum of the bend angles of in all pars: as a straight member with no bends would be the easiest to manufacture with extrusion.

The resulting assembly that achieves perfect adjustability for all four KCs is presented in Figure 19, and the accompanying subassembly partitioning is shown in Figure 20. The effects of the additional two criteria on the resulting assembly can be seen by the several long, straight or nearly straight members, among the shorter members whose joints contribute to the overall adjustability.

The computer software used in this example is written in C++ with intense use of data types and algorithms of LEDA developed at Max-Planck-Institut für Informatik, Saarbrücken, Germany. The genetic algorithm routine is implemented with GALib developed at MIT CADLAB. The C source codes written by Edward Rothberg are used for solving the minimum cut problem, which implement Goldberg and Tarjan's maximum flow algorithm [22].

SUMMARY AND FUTURE WORKS

This paper presented an optimization-based method of assembly synthesis for in-process dimensional adjustability, where a genetic algorithm (GA) generates candidate assemblies based on a joint library specific for an application domain. Each candidate assembly is evaluated by solving a subassembly partitioning problem for optimal in-process adjustability, posed as an equivalent minimum cut problem on weighted graphs. A case study on a 3D automotive space frame with the accompanying joint library is presented.

As a future work, the authors plan to extend the approach in [3] to a three-dimensional model with a joint library, with additional criteria (such as the ones considered in the above example) to prune the enumeration tree of the AND/OR graph of assembly synthesis. For this, major development would be necessary to generalize the concept of non-forced fit introduced in [3] into three dimensional assemblies and joints. Currently under development is a unified representation of motion constraints imposed by joints and fixtures tailored for the assembly synthesis process, which will be presented at a future opportunity.

ACKNOWLEDGMENTS

The authors acknowledge Mr. Yasuaki Tsurumi at Toyota Central R&D Laboratory and Prof. Shinji Nishiwaki at Kyoto University for providing us with the body model. We also like to acknowledge Dr. Malen at the University of Michigan for the joint library of the automotive space frames. This work has been supported by the National Science Foundation with a CAREER Award (DMI-9984606). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of

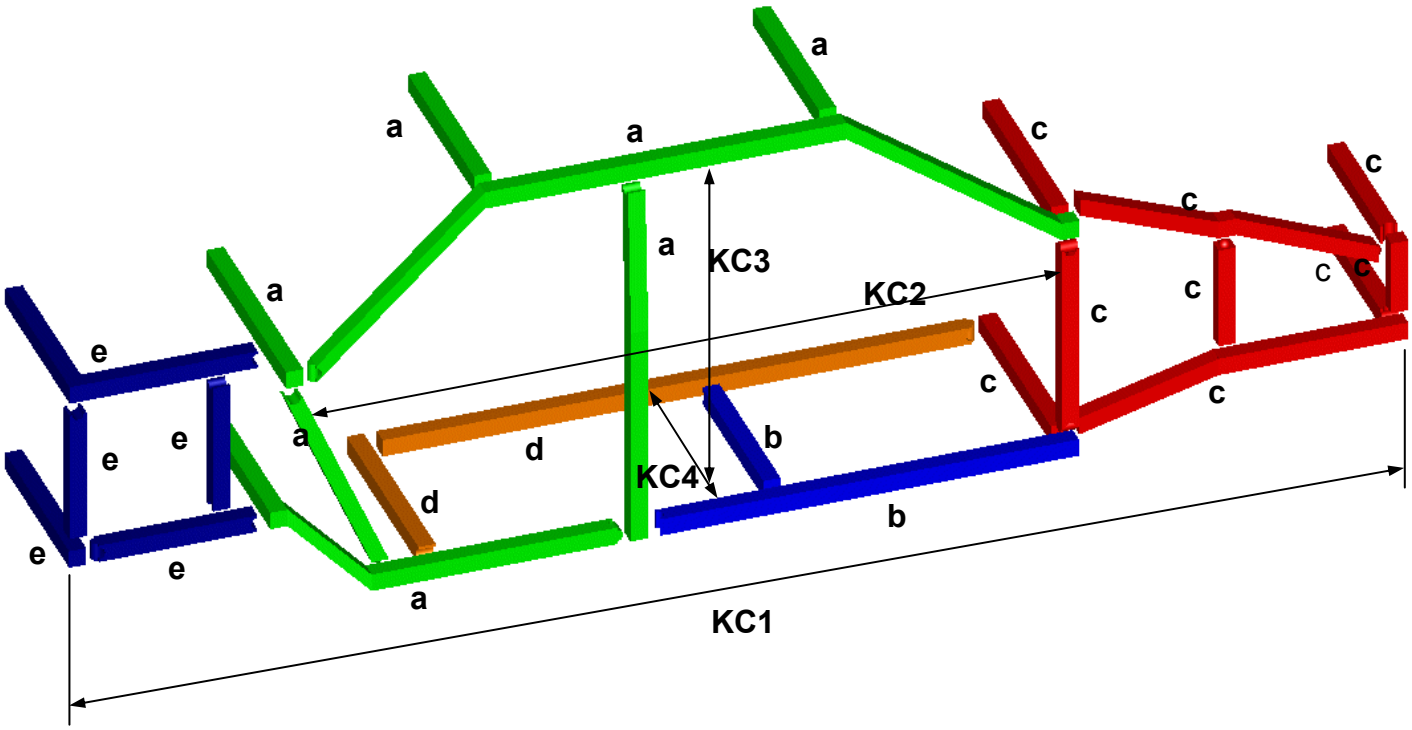

Figure 19. A synthesized assembly from the model in Figure 17 for the optimal in-process adjustability. Also, see Figure 20 for the accompanying subassembly partitioning.

the National Science Foundation. A matchable fund for this grant has been provided by Toyota Motor Company. These sources of support are gratefully acknowledged.

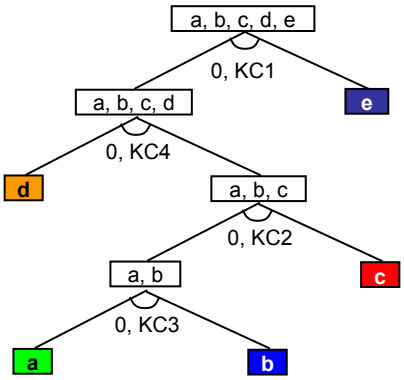

Figure 20. The binary tree of subassembly partitioning for the solution presented in Figure 19.

REFERENCES

[1] Whitney, D. E., Mantripragada, R., Adams, J. D., and Rhee, S. J., 1999, "Designing assemblies," *Research in Engineering Design*, Vol. 11, pp. 229-253.

[2] Yetis, F. A. and Saitou, K., 2002, "Decomposition-based assembly synthesis based on structural considerations," *Journal of Mechanical Design*, Vol. 124, pp. 593-601.

[3] Lee, B. and Saitou, K., 2002, "Decomposition-based assembly synthesis for in-process dimensional adjustability", Proceedings of the *2002 ASME International Design Engineering Technical Conferences*, Montreal, Canada, Paper no. DETC2002/DAC-34086. An extended version has been accepted to *Journal of Mechanical Design*.

[4] Homem de Mello, L. S. and Sanderson, A. C., 1990, "AND/OR graph representation of assembly plans," *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 2, pp. 188-199.

[5] Bourjault, A., 1984, *Contribution a une approche methodologique de l'assemblage automatise: elaboration automatique des sequences operatoires*, Ph.D. Thesis, Universite de Franche-Comte.

[6] De Fazio, T. L. and Whitney, D. E., 1987, "Simplified generation of all mechanical assembly sequences," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 6, pp. 640-658.

[7] Lee, D. J. and Thornton, A. C., 1996, "The identification and use of key characteristics in the product development process," Proceedings of the *1996 ASME Design Engineering Technical Conferencses*, Irvine, California, Paper no. 96-DETC/DTM-1506.

[8] Mantripragada, R. and Whitney, D. E., 1998, "The datum flow chain," *Research in Engineering Design*, Vol. 10, pp. 150-165.

[9] Homem de Mello, L. S. and Sanderson, A. C., 1991, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pp. 228-240.

[10] Homem de Mello, L. S. and Sanderson, A. C., 1991, "Representations of mechanical assembly sequences," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pp. 211-227.

[11] Wang, C.-H., 1997, *Manufacturability-driven decomposition of sheet metal products*, Ph.D. Thesis, Carnegie Mellon University.

[12] Wang, C.-H. and Bourne, D., 1997, "Concurrent decomposition for sheet-metal products", Proceedings of the *1997 ASME Design Engineering Technical Conferences*, Sacremento, California. Paper no. DETC97/ DFM-4328.

[13] Lyu, N. and Saitou, K., 2002, "Decomposition-based assembly synthesis based on structural stiffness considerations," Proceedings of the *2002 ASME International Design Engineering Technical Conferences*, Montreal, Canada, Paper no. DETC2002/DAC-34083. An extended version has been accepted to *Journal of Mechanical Design*

[14] Cetin, O. and Saitou, K., 2001, "Decomposition-based assembly synthesis for maximum structural strength and modularity," Proceedings of the *2001 ASME International Design Engineering Technical Conferences*, Pittsburgh, Pennsylvania, Paper no. DETC2001/ DAC-21121. An extended version has been accepted to *Journal of Mechanical Design*.

[15] Cetin, O., Saitou, K., Nishigaki, H., Nishiwaki, S., Amago, T., and Kikuchi, N., 2001, "Modular Structural Component Design using the First Order Analysis and Decomposition-based Assembly Synthesis," *P*roceedings of the *2001 ASME* International Mechanical Engineering Congress and Exposition, New York, New York, November 11–16, IMECE2001/DE-23265.

[16] Liu, S. C. and Hu, S. J., 1998, "Sheet metal joint configurations and their variation characteristics", *Journal of Manufacturing Science and Engineering*, Vol. 120, No. 2, pp. 461-467.

[17] Ceglarek, D. and Shi, J., 1998, "Design Evaluation of Sheet Metal Joints for Dimensional Integrity," *Journal of Manufacturing Science and Engineering*, Vol. 120, No. 2, pp. 452-460.

[18] Mantripragada, R. and Whitney, D. E., 1999, "Modeling and controlling variation propagation in mechanical assemblies using state transition models," *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 1, pp. 124-140.

[19] Foulds, L. R., 1991, *Graph Theory Applications*, Springer-Verlag, New York, New York.

[20] Lee, S., 1991, "Backward assembly planning," Proceedings of the *1991 IEEE International conference on Tools for AI*, San Jose, California, pp. 2382-2391.

[21] Malen, D. E., 2002, Course notes from *Fundamentals of Automotive Body Structures*, Dollar Bill Copying, Ann Arbor, MI.

[22] Goldberg, A. V. and Tarjan, R. E., 1988, "A new approach to the maximum-flow problem," *Journal of the Association for Computing Machinery*, Vol. 35, No. 4, pp. 921-940.