Proceedings of DETC'04
ASME 2004 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
September 28-October 2, 2004, Salt Lake City, Utah, USA

DETC2004-57771

INTEGRATED SYNTHESIS OF ASSEMBLY AND FIXTURE SCHEME FOR PROPERLY CONSTRAINED ASSEMBLY

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

ABSTRACT

This paper presents an integrated approach to design an assembly, fixture schemes and an assembly sequence, such that the dimensional integrity of the assembly is insensitive to the dimensional variations of individual parts. The adjustability of critical dimensions and the proper constraining of parts during assembly process are the keys in achieving the dimensional integrity of the final assembly. A top down design method is developed which recursively decomposes a lump of initial product geometry and fixture elements matching critical dimensions, into parts and fixtures. At each recursion, joints are assigned to the interfaces between two subassemblies to ensure parts and fixtures are properly constrained at every assembly step. A case study on a simple frame structure is presented to demonstrate the method.

INTRODUCTION

Structural enclosures of modern mechanical products, such as ship hulls, airplanes and automotive bodies, typically are made of hundreds or thousands of parts due to their geometric complexity and sizes. As the number of parts increases, however, achieving the dimensional integrity of the final assembly becomes more difficult due to the inherent variations in manufacturing and assembly processes.

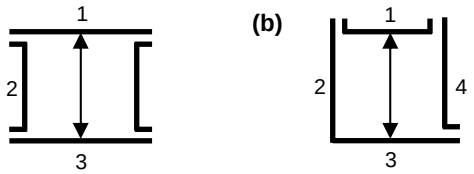

Figure 1. Two box designs (a) without and (b) with adjustable height during assembly [1].

A solution is to adjust critical dimensions *in assembly processes* when parts or subassemblies are located and fully constrained in fixtures. This in-process dimensional adjustment is typically facilitated by slip planes, mating surfaces at joints that allow a small amount of relative motions. For example, Figure 1 shows two designs of a rectangular box. In contrast to design in (a) with no in-process adjustability of the critical dimensions (length between sections 1 and 3), design in (b) provides slip planes such that relative location of parts can be adjusted along the critical dimension.

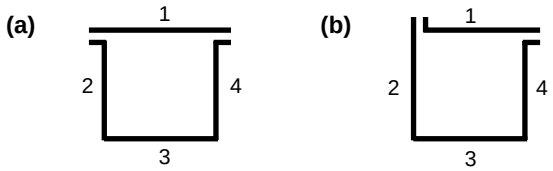

Figure 2. Two box designs (a) without and (b) with properly constrained parts [1].

The dimensional integrity of an assembly is also affected by the post-assembly distortion due to the internal stress induced by joining parts with dimensional mismatches. A solution is to ensure the proper constraining of subassemblies at each assembly step. For example, part 1 in Figure 2 (a) is not properly constrained and therefore the post-assembly distortion might occur, if the length of sections 2 and 4 are slightly different due to manufacturing variation. With two slip planes perpendicular to each other, the design in (b) can absorb manufacturing variations within parts 1 and 2-3-4, provided that variations in angles are negligible.

In addition to the assembly design including joint types at part interfaces, the assembly sequence also influences in-process dimensional adjustability and proper part constraints. In the assembly sequence in Figure 3 (a), the critical dimension

* Corresponding author

(total length) is not adjustable since there is no slip plane parallel to it when the total length is realized with the addition of part 1. On the other hand, the sequence shown in (b) provides the slip plane at the assembly step where the critical dimension is achieved, to absorb the variation in length. As another example, the sequence in Figure 4 (b), where each critical dimension is independently adjusted at each step, is more desirable than the sequence in (a), where both dimensions are adjusted at one step, inevitably requiring a compromise between two potentially conflicting critical dimensions. Figure 5 illustrates an effect of the assembly sequence on proper part constraints, where the sequence in (a) causes over-constraint at the second step, whereas all parts are properly constrained at all steps in (b), thus avoiding potential assembly stress.

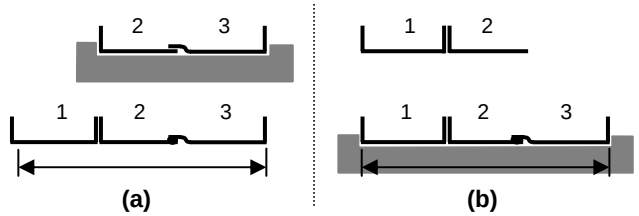

Figure 3. Assembly sequences (a) without and (b) with in-process adjustability (modified from [2]).

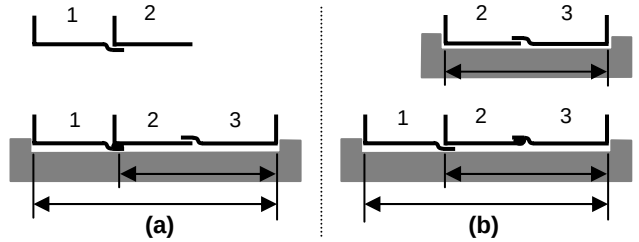

Figure 4. Assembly sequences where two dimensions are adjusted (a) at one step and (b) independently at two steps (modified from [2]).

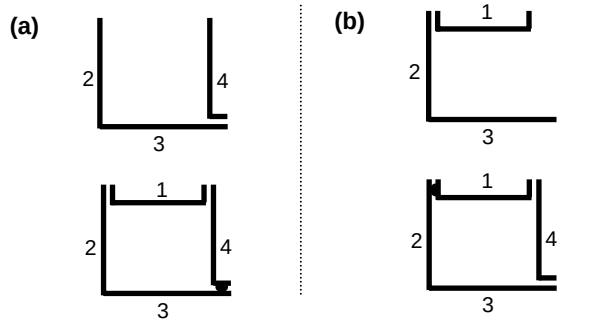

Figure 5. Assembly sequences (a) without and (b) with proper constraints [1].

Let us note Figure 4 again. In Figure 4 (b), each critical dimension is realized on a separate fixture, in which case, it is the only feasible assembly sequence to realize both critical dimensions independently. However, to make the problem more complicated, other assembly sequences are feasible if fixtures are arranged differently. For example, in Figure 6, both critical dimensions are realized independently on the only fixture, in two different assembly sequences, (a) and (b). What is different from Figure 4 (a) is that pins locating part 1 and 2 control the location of part 1 and 2 separately, thus enabling independent realization of the critical dimensions. The pin locating part 3 serves realizing both critical dimensions. Indeed, the fixture in Figure 6 is the union of the two fixtures utilized in Figure 4 (b). Examining different ways of arranging fixtures for multiple KCs is valuable, as using one fixture to deal with several critical dimensions is quite common for large scale assemblies, especially when several parts constitute a flat subassembly.

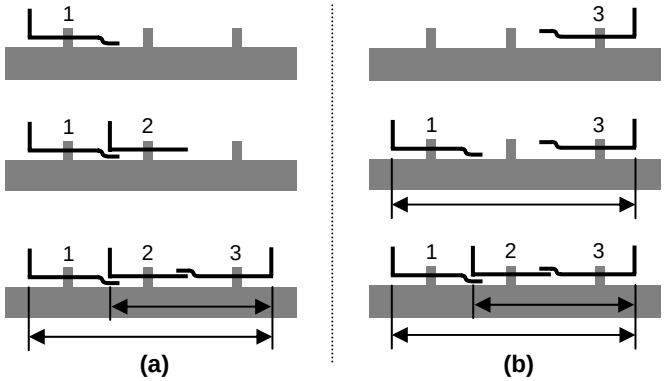

Figure 6. Feasible assembly sequences depend on utilization of fixtures. Compare with Figure 4 (b).

As pointed out by industry practitioners and researchers, the proper constraint and adjustability are key elements in assembly design to achieve high precision and accuracy with low cost parts [3]. Whereas it is important to carefully design and sequence the assembly and fixtures in order to avoid over-constraints and the loss of desired adjustability, industry practices do not come up to systematic approaches. Despite the fact that the proper constraint and adjustability should be ensured between 'subassemblies' at 'every assembly step,' not between parts, current design practices and CAD systems overlook this important property and mistreat joints and tolerances as the attributes of part geometry without considering assembly sequences. For complex mechanical assemblies, this causes many dimensional discrepancies at the manufacturing stage, followed by costly redesigns and reworks. To make matters worse, typical engineering countermeasures in such situations have often been to tighten part tolerances, without examining the assembly design and tolerance relationships as a whole [4].

As a remedy, we have presented a top-down decomposition-based assembly synthesis method [1] to fully enumerate all feasible sets of part decomposition, joint assignments and an assembly sequence, for 2D geometry. Assuming that assemblies can be built in the reverse sequence of decomposition, the method recursively decomposes a given product geometry into two subassemblies until parts become manufacturable. At each recursion, joints are assigned to the interfaces between two subassemblies to ensure in-process dimensional adjustability and properly constraint. The method

has also been applied to 3D beam-based structure [5], where Screw Theory [6] is utilized for the evaluation of in-process adjustability and proper constraints of subassemblies at every assembly step.

However, our previous works [1, 5] assume one fixture to achieve every critical dimension (as shown in Figure 4 (b)), and hence incapable of exploring various *fixture schemes*[†]. This paper extends our previous works to design fixture scheme as an integrated part of assembly synthesis, which enumerates all feasible "designs" (assembly designs, fixture schemes, and assembly sequences) by treating fixtures as an entity of assembly. Not only does this integration explore all feasible fixture schemes along with assembly designs, but also reveals feasible assembly sequences that were illicit in our previous methods [1, 5], such as those shown in Figure 6. A case study on a simple space frame is presented to demonstrate the method. Considering the number of parts, the number of fixtures, the depth of assembly tree, and the number of under-constraints as objectives to minimize, a multi-objective graph search is performed on the enumerated feasible designs, in order to obtain Pareto optimal solutions. Some representative designs in the Pareto set are examined to illustrate the trade-offs among the assembly design, fixture scheme, and assembly sequence.

RELATED WORKS

Since previous works in general relevance to assembly synthesis are reviewed in [1], this section focuses on the literature directly related to the present extension of the assembly synthesis method, namely on property constrained assembly designs and fixture designs.

The advantages of properly constrained assemblies are well known to practitioners in precision machinery design, and several methods have been proposed in literatures including: Kinematic Design [7], Minimum Constraint Design [8] and Exact Constraint Design [3, 4]. These works describe disadvantages of over-constraints and provide good practices as well as analytical methods to compute constraints. In these works, the most commonly cited merit of properly constraint design is repeatability that leads to high precision. Downey *et al.* [9] analyzed and classified elements of assemblies that absorb manufacturing variations of parts.

A universal analytical method for motion and constraint analysis dates back to Screw Theory, a pioneering work by Ball [6]. Since then, Screw Theory has been applied to areas of mechanism, robotics and machine design. Among others, Waldron [10] utilized the screw theory to build a general method which can determine all relative degrees of freedom (DOF) between any two rigid bodies making contacts to each other. Blanding [4] shows the application of screw theory to assembly design. Adams and Whitney [11] also used screw theory to compute the constraints on parts and applied it to rigid body assemblies with mating features such as pin-slot joint. Asada and By [12] proposed kinematic analysis method for fixture layout design by modeling kinematic constraints of fixture locators as a Jacobian matrix, which should have full rank to locate a given work piece uniquely at a desired position.

While these works provide tools for analyzing constraints in a given assembly and design guidelines, they do not address a systematic and integrated synthesis of an assembly and fixture scheme with desired constraint characteristics such as in-process dimensional adjustability and proper part constraints, as discussed in this paper. Although design of fixture scheme should precede physical fixture layout design, authors could not find previous works attacking this problem in a systematic way.

TERMINOLOGY

Since the assembly synthesis deals with objects yet to be decomposed into an assembly of separate parts, a few terms and concepts need to be defined to avoid confusion with generic meanings used in other literatures.

- A *product geometry* is a geometric representation of a whole product as one piece before decomposition into parts.
- 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 member within the group. A product geometry is a configuration, so as a part (as defined below).
- The *Key Characteristics* (KCs) are defined by Lee and Thornton [13] as product features, manufacturing process parameters, and assembly features that significantly affect a product's performance, function and form. In this paper, a KC refers to a critical dimension to be achieved in assemblies.
- A *decomposition* is a transition of a configuration into two sub-configurations by removing connections between two members.
- A *part* is a configuration that is not decomposed further under given criteria, *e*.g., a minimum part size. A part may consist of one or more members.
- A *joint library* is a set of joint types available for a specific application domain (Figure 7).

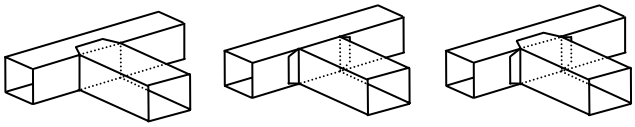

Figure 7. An example of joint library for 3-D beam based assemblies consisting of lap, butt and lap-butt.

- An (*synthesized*) *assembly* is a set of parts and joints that connect every part in the set to at least one of other parts in the set.
- *Assembly synthesis* is a transformation of a product geometry into an assembly.
- A *fixture element* is an imaginary part of a fixture to control a KC. Physically, a KC will be controlled by a set of locators, and the fixture element is abstract

[†] Fixture scheme is defined as a plan showing which fixture will control what critical dimensions where in assembly sequence. More formal definition will follow in terminology section.

representation of this set of locators. Thus, each KC will have a fixture element corresponding to it. A *fixture* is a group of fixture elements and contols corresponding KCs.

- *Fixture scheme* is partitioning the whole set of fixture elements into groups and assigning them into assembly sequence.

SCREW THEORY‡

In Screw Theory, a *screw* is defined as a pair of a straight line (*screw axis*) in a 3D Cartesian space and a scalar (*pitch*). It is commonly represented by *screw coordinates***,** a pair of two row vectors $\mathcal{S} = (\boldsymbol{s}; \boldsymbol{s}_0)$ in 3D Cartesian coordinates, where $\boldsymbol{s}$ is a unit vector parallel to the screw axis and $\boldsymbol{s}_0$ is given as:

$$\boldsymbol{s}_0 = \boldsymbol{r} \times \boldsymbol{s} + p\boldsymbol{s} \tag{1}$$

where $\boldsymbol{r}$ is the position vector of a point on the screw axis and p is the pitch. Equivalently, p can be expressed using $\boldsymbol{s}$ and $\boldsymbol{s}_0$ as:

$$p = \frac{\mathbf{s} \cdot \mathbf{s}_0}{\mathbf{s} \cdot \mathbf{s}} \tag{2}$$

A screw with an infinite pitch does not follow Equation (1), therefore it is denoted by $\boldsymbol{s}$ being the zero vector and $\boldsymbol{s}_0$ being the unit vector parallel to the screw axis.

Two types of screws, twist and wrench, are utilized in this paper. A *twist* is a screw representing a motion of a rigid body simultaneously rotating around and translating along an axis. Using screw coordinates, it is denoted as $\mathcal{T} = (\boldsymbol{\omega}; \boldsymbol{v})$, where $\boldsymbol{\omega}$ is the angular velocity and $\boldsymbol{v}$ is the linear velocity of a point on the body (or its extension) located at the origin of global reference frame. A *wrench* is a screw representing a force along and a moment around an axis exerted on a rigid body. Using screw coordinates**,** it is denoted as $\mathcal{W} = (\boldsymbol{f}; \boldsymbol{m})$, where $\boldsymbol{f}$ is the force and $\boldsymbol{m}$ is the moment that a point on the body (or its extension) located at the origin of global reference frame should resist.

Two screws $\mathcal{S}_1 = (\boldsymbol{s}_1; \boldsymbol{s}_{01})$ and $\mathcal{S}_2 = (\boldsymbol{s}_2; \boldsymbol{s}_{02})$ are *reciprocal* to each other, if and only if they satisfy:

$$\boldsymbol{s}_1 \cdot \boldsymbol{s}_{02} + \boldsymbol{s}_{01} \cdot \boldsymbol{s}_2 = 0. \tag{3}$$

If a twist $\mathcal{T}$ is a reciprocal of wrench $\mathcal{W}$ (or vise versa), $\mathcal{W}$ does no "work" to a rigid body moving according to $\mathcal{T}$.

When a body can receive linear combinations of several screws (either twist or wrench), this set of screws are typically represented as a matrix where each screw in the set forms a row vector of the matrix. This matrix is called a *screw matrix*. As its row space is the screw space (a space formed by the set of screws in the matrix), the rank of a screw matrix is equal to the dimension of the screw space.

The function reciprocal(**S**) returns a screw matrix whose row consists of the screws reciprocal to those in **S**. It can be obtained by exchanging the former three columns and the letter three columns of the null space of **S**.

‡ The terminology and formalization in this section are summarized from [6], [11], [14], [15] and [16].

The *union* of screw matrices represents the sum of the screw spaces defined by the matrices, and can be obtained by simply "stacking" them on top of one another:

$$\bigcup_{i=1}^{n} \mathbf{S}_i \equiv \begin{pmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \\ \cdots \\ \mathbf{S}_n \end{pmatrix} \tag{4}$$

The *intersection* of screw matrices is the set of screws common to the screw matrices, and can be computed through double reciprocals:

$$\bigcap_{i=1}^{n} \mathbf{S}_i \equiv \mathrm{reciprocal}\left(\bigcup_{i=1}^{n} \mathrm{reciprocal}(\mathbf{S}_i)\right) \tag{5}$$

Since a twist and a wrench are also screws, the definitions of reciprocal, union, and intersection hold.

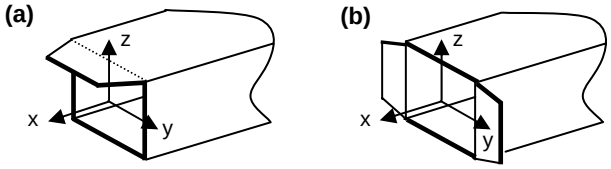

Figure 8. Lap (a) and lap-butt joint (b) of a beam based model and the local coordinate frames for twists.

Woo and Freudenstein [14] presents kinematic properties of various joint types in screw coordinates, which are adopted to build twist matrices of beam joint types. Figure 8 (a) shows a typical lap joints found in beam-based structures. When it is attached to another beam, the tab allows planar motion parallel to x-y plane. Also, if we assume that the length of the tab is very small compared to the length of the beam, it can be treated as a line contact along y-axis, allowing the rotation about y-axis. Thus, the lap join, with respect to the local coordinate frame shown in the figure, can be modeled as a twist matrix:

$$\mathbf{T}_{lap} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

Similarly, a butt joint in Figure 8 (b) allows the motion parallel to y-z plane, can be modeled as:

$$\mathbf{T}_{butt} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

In twist matrices in Equation (6) and (7), each row represents an independent motion, and each non-zero number represents rotation or translation along a corresponding axis –

ω_x, ω_y, ω_z, v_x, v_y or v_z. For example, the first row in Equation (6) has 1 at the second column, which means the lap joint allows rotational motion about y-axis. In the third row, it has 1 at the fourth column, meaning translation along the x-axis is allowed. Since these matrices are used only to give information on which DOFs are not constrained for a joint type, the magnitude of each twist (row) of these twist matrices (*i.e.*, the magnitudes of the angular and linear velocities in the twist) does not have significant meaning in this paper.

Once the twist matrix is obtained for a joint type, the reciprocal wrench matrix can be computed as described above. For instance, the wrench matrices corresponding to twist matrices in (6) and (7) are:

$$\mathbf{W}_{lap} = \text{reciprocal}(\mathbf{T}_{lap}) = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (8)$$

$$\mathbf{W}_{butt} = \text{reciprocal}(\mathbf{T}_{butt}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (9)$$

Each non-zero number now represents force or moment along a corresponding axis – f_x, f_y, f_z, m_x, m_y or m_z. Since a wrench that is a reciprocal of a twist does no "work" to a rigid body moving according to the twist, these are the forces and moments the joint supports (hence resulting no work). For example, in the first row in Equation (8) has 1 at the third column, which means the lap joint can support a force along z-axis.

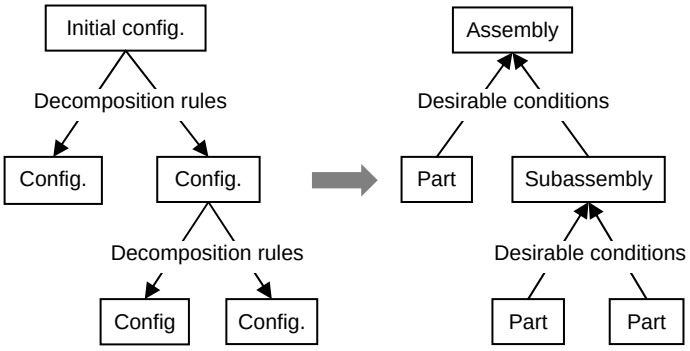

Figure 10. Assembly synthesis by top-down hierarchical decomposition. Assembly sequence is the reverse of the decomposition sequence.

ASSEMBLY AND FIXTURE SCHEME SYNTHESIS

Assembly synthesis via recursive decomposition

There are numerous issues related to assembly design. Among others, adjustability and proper constraint are the key necessary conditions for dimensional integrity. Dissimilar to other issues such as structural stiffness and product function, these two conditions should be satisfied *at every assembly step*, as illustrated in Figures 1-5. By taking advantage of this fact, one can hierarchically decompose a given product geometry such that (sub)geometries at each decomposition step satisfy the above desired conditions when they are assembled back together in the reverse order (see Figure 9). Our previous works [1, 5] suggested the framework of assembly synthesis via such hierarchical decomposition, which was successfully applied to simple 2-D [1] and 3-D [5] geometries.

Generation of fixture elements

A KC, in this paper, is assumed to be a critical dimension between parts to be achieved by the adjustment during the assembly process. Thus, the dimension noted as a KC will be constrained by a fixture, according to which parts being assembled will be located. In this context, we know the fixture would have to constrain *at least* the DOFs specified by the KC, regardless of its physical embodiment. Provided a KC is controlled by a fixture, the assembly of two subassemblies connected by a KC can be viewed as two assembly steps, involving two subassemblies and a fixture, such as {{part1, fixture}, part2}. As depicted in Figure 10, this allows each KC to be replaced by a fixture element connecting the same members. The graph representation shown in Figure 11 is what we call *configuration graph*. After replacing KCs with fixture elements, the configuration graph is a pair:

$$C = (M, E) \quad (10)$$

, where M is the set of nodes representing members and fixture elements, and E is the set of edges representing connections. Each node in M is associated with its type (members are in white and fixture element are in black in Figure 10), and each edge in E between a member node and a fixture element node is associated with a wrench matrix representing the DOFs to be constrained by the replaced KC. For example, if *kc1* in Figure 10 is the distance between members 1 and 3 in y-direction measured at x = 1.5 in the global reference frame, then the wrench matrix associated with edges {1, *f1*} and {3, *f1*} is

$$\mathbf{W}_{l1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1.5 \end{pmatrix}. \quad (11)$$

where subscript l indicates "locator". Similarly, the wrench matrix associated with for edges {1, *f2*} and {2, *f2*} are:

$$\mathbf{W}_{l2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (12)$$

While seemingly subtle, this replacement of KCs with the corresponding fixture elements is a major advance beyond our previous works [1, 5], which enables an elegant integration of the synthesis of a fixture scheme into assembly synthesis process. Initially, each fixture element is connected to all the other fixture elements, in order to allow the exploration of all possible fixture schemes. The connections between a fixture element and a member represent *minimum locators* that constrain at least the DOFs specified by the replaced KC. Any additional DOFs needed to uniquely locate the part will be computed during assembly and fixture scheme synthesis as described in the following section.

Further, the configurations after the replacement of KCs with fixture elements are classified to three classes:

- *Incomplete configuration*: a configuration with unconnected members or with a fixture element connected to less than two members. For example, the second step of Figure 6 (b) is an incomplete configuration since, members are not connected and the fixture element controlling the distance between members 2 and 3 has only one connection (to part 3) due to the absence of member 2.
- *Fixture*: a configuration consisting of only fixture elements.
- *Complete configuration*: a configuration that is neither an incomplete configuration nor a fixture.

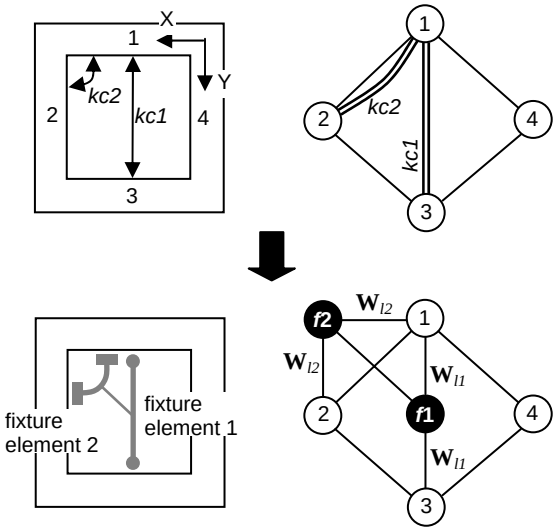

Figure 10. Replacement of KCs with fixture elements whose locators constrain the same DOFs.

Feasible binary decomposition

The assembly synthesis algorithm [1] adopted in this paper assumes every assembly step combines a pair of subassemblies. Conversely, the algorithm decomposes a configuration into two (sub)configurations by removing some connections, which is equivalent of finding a *cut-set* [17] of the configuration graph. Decomposition is made only when the reverse of it yields a feasible assembly step, for which there are two criteria. Firstly, the assembly step is binary - only two subassemblies are joined at the assembly step. This is justified by the fact that a non-binary assembly step (*eg*., assembly of multiple parts on a fixture in one step) can be broken down to an equivalent sequence of binary assemblies. Secondly, at least one of two subassemblies joined at the assembly step is a complete configuration, which is justified in the next paragraph.

When a configuration is incomplete, subassemblies should remain on the fixture because subassemblies are either not connected or the fixture has at least one assigned KC yet to realize (such as the status shown in Figure 6 (b)). Since fixtures are usually heavy or grounded, it would be very rare that a subassembly attached to a fixture is assembled to another subassembly in the same situation, or to another fixture. For the same reason, assembly of two fixtures is considered infeasible. On the other hand, when a configuration is complete, it has only one connected subassembly and, if any, a fixture with all assigned KCs realized. Therefore, it is ready to leave the fixture for further assembly with any configuration including a fixture.

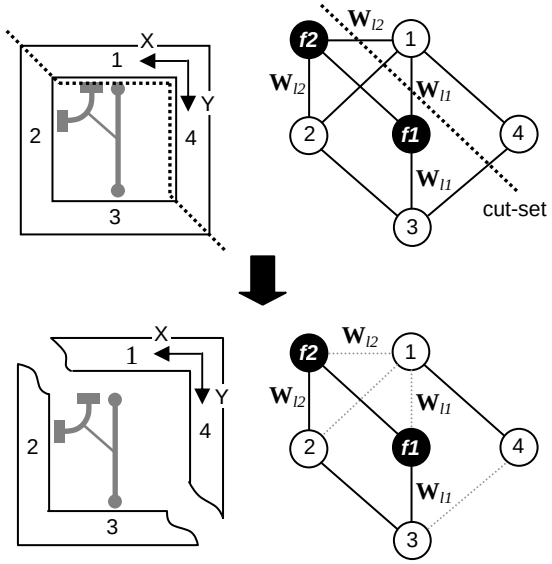

Figure 11. A feasible decomposition

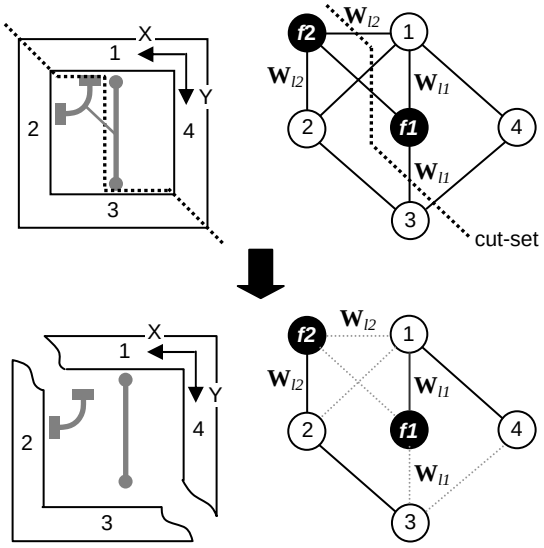

Figure 12. An infeasible decomposition that results in two incomplete sub-configurations.

For example, Figure 11 shows a feasible decomposition yielding one complete and one incomplete sub-configurations. In Figure 12, both sub-configurations are incomplete, thus will not be considered as a feasible decomposition.

More formally, decomposition from configuration C_a = (M_a, E_a) to two sub-configurations C_b = (M_b, E_b) and C_c = (M_c, E_c) is feasible if the following conditions are satisfied:

- $M_b \neq \varnothing$ and $M_c \neq \varnothing$.
- C_b and C_c are connected.

- At least one of C_b and C_c is a complete configuration.
- $M_a = M_b \cup M_c$.
- $M_b \cap M_c = \varnothing$. (13)

The 1st condition states sub-configurations should be nonempty. The 2nd condition states the sub-configurations must be connected. The 4rd and 5th conditions specify the configuration should split into a pair of disjoint sub-configurations.

Decomposition rule for dimensional integrity

Once a decomposition satisfying conditions in Equation (13) is found, feasible joint types are assigned to broken connections, which is represented as mapping $\gamma_d : CS_d \mapsto JL$, where CS_d is the cut-set broken by decomposition d and JL is a library of joint and locator types. With the joint assignment, (binary) decomposition d can be uniquely specified as $d = (M_a, \gamma_d, (M_b, M_c))$. See Figure 13 for an example. Note that feasible joint types may depend on the local geometry near the joint location. For example, feasible joint types between two perpendicular beams would be different from those for two coaxial beams. The broken connections with the assigned joints are associated with the wrench matrices computed according to the assigned joint types and orientations. Every connection between a member and a fixture element already has a wrench matrix computed in the previous step; therefore no action is taken even if it is broken by a decomposition.

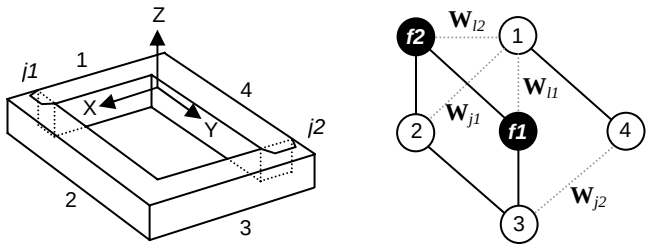

Figure 13. Joints assigned to broken connections, for which wrench matrices are computed accordingly (fixture elements omitted in the left figure).

Having replaced all KCs with the corresponding fixture elements, the only criterion that needs consideration for assigning joint types to broken connections is the proper constraint of the mixture of subassemblies and fixtures at every assembly step. In particular, there is no need to explicitly consider the adjustability for KCs as required in our previous work [1] since the proper constraint including the DOF constrained by KCs implies the assigned joints do not interfere with the DOFs constrained by KCs, automatically ensuring their individual adjustability.

In order for subassemblies being assembled to be properly constrained, the joints should not constrain the same DOF more than once, as illustrated in Figure 5. This assembly rule is inversely stated as the decomposition rule for proper constraint in our previous work [1], which only allows the combination of joints yielding no over-constraint of parts. Although when one subassembly is located on an empty fixture, the fixture should constrain all six DOFs [12], when the next subassembly is put together contacting the other subassembly on the fixture, it would be constrained by both the other subassembly and the fixture. Therefore, joints must be selected in such a way that no DOF is constrained twice not only among joints but also with locators. In such cases, the intersection of the wrench matrix corresponding to any subset of CS_d and the wrench matrix of any other disjoint subset must result in the zero matrix:

$$\forall C_1, C_2 \subseteq CS_d, C_1 \cap C_2 = \varnothing, \quad (\bigcup_{e \in C_1} \mathbf{W}_{\gamma_d(e)}) \cap (\bigcup_{e \in C_2} \mathbf{W}_{\gamma_d(e)}) = \mathbf{O} \tag{14}$$

which is also equivalent to:

$$\mathrm{rank}(\bigcup_{e \in CS_d} \mathbf{W}_{\gamma_d(e)}) = \sum_{e \in CS_d} \mathrm{rank}(\mathbf{W}_{\gamma_d(e)}) \tag{15}$$

Further, in order to have all six DOFs constrained,

$$\mathrm{rank}(\bigcup_{e \in CS_d} \mathbf{W}_{\gamma_d(e)}) = \sum_{e \in CS_d} \mathrm{rank}(\mathbf{W}_{\gamma_d(e)}) = 6 \tag{16}$$

When any set of joint types and fixture elements satisfies Equation (15) with the total rank less than six, it is considered to be feasible, assuming that additional fixtures or locators on existing fixtures will be arranged. Six less the number of DOFs constrained is counted as the number of under-constraints for each feasible joint assignment and recorded as $uc(\gamma_d)$. A predicate of a decomposition $d = (M_a, \gamma_d, (M_b, M_c))$ for complying the rule is given as:

$$\mathrm{de}: 2^{M_0} \times (2^{E_0} \mapsto JL) \times (2^{M_0} \times 2^{M_0}) \mapsto \{true, false\} \tag{17}$$

where $\mathrm{de}(M_a, \gamma_d, (M_b, M_c))$ is *true* if and only if the conditions in (13) and Equation (15) is satisfied.

There is an important exception to Equation (15), for which compensation should be made before it is checked against Equation (15); when there are connections in CS_d, from multiple fixture element to one member, the wrench matrices associated to the connections should be unionized such that the intersection among them could be ignored. This is based on a basic assumption that there would be no over-constraint between a fixture and a member. Suppose there is a set of fixture elements that are connected to a member and more than one of these connections are broken by a decomposition. In this case, even if there is a DOF constrained by more than one fixture elements, the DOF will be constrained by one locator in actual implementation. For example, see the first step in Figure 6 (b). When part 3 is placed on the fixture, the fixture is constraining two KCs of the same DOF, the distances to 1 and 2. However no one will use one locator for each KC, which will certainly yield over-constraint. When this step is generated through decomposition, the decomposition would break connections between the member 3 and each of the two fixture elements transformed from the two KCs. In order to match the assumption that there would be one locator for a DOF, the wrench matrices for these connections should be unionized.

Consider the product geometry decomposed in Figure 11 and the joint assignment shown in Figure 13, which has two lap

joints *j1* and *j2*, and two locators *l1* and *l2*, for edges cut by the decomposition. Because the decomposition is breaking multiple connections from member 1 to fixture elements, wrench matrices for these connections should be unionized as described in the previous paragraph. From Equation (12) and (13), we can compute:

$$\mathbf{W}_{l1} \cup \mathbf{W}_{l2} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1.5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

Suppose the location of *j1* and *j2* in global reference frame X-Y-Z are (3, 0, 0) and (0, 4, 0). Then, based on the local coordinate frame of lap joint shown in Figure 8 and orientation of *j1* and *j2*, $\mathbf{W}_{lap}$ (Equation (8)) can be transformed into $\mathbf{W}_{j1}$ and $\mathbf{W}_{j2}$ in global reference frame:

$$\mathbf{W}_{j1} = \begin{pmatrix} 0 & 0 & 1 & 0 & -3 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{19}$$

$$\mathbf{W}_{j2} = \begin{pmatrix} 0 & 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{20}$$

Unionizing the three matrices in Equation (18) – (20),

$$\bigcup_{e \in CS_d} \mathbf{W}_{\gamma_d(e)} \sim \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{\S} \tag{21}$$

Whereas the summation of the ranks of individual matrices is six, the rank of union is only five, which implies that this combination of joints yield an over-constraint of one DOF. In fact, the intersection of $\mathbf{W}_{j1}$ and $\mathbf{W}_{j2}$ is not a zero matrix. As this joint assignment does not satisfy the decomposition rule, Equation (15), the assembly synthesis process will discard it.

Part manufacturability

The decomposition stops when the resulting subconfigurations become manufacturable by a chosen manufacturing process. In the following case study on frame structures, components are assumed to be extruded and bent. Therefore, a predicate of a configuration M_a for stopping decomposition is given as:

$$\text{stop\_de} : 2^{M_0} \mapsto \{true, false\} \tag{22}$$

, where stop_de(M_a) is true (*i.e.*. decomposition continues) if and only if none of the first four conditions are satisfied or the fifth condition is satisfied:

1. M_a contains both member(s) and fixture member(s).
2. The induced subgraph on members in M_a has a closed loop (cannot extrude such parts).
3. Three or more members in M_a are connected to each other at a single point (cannot extrude such parts).
4. Members in M_a lie on more than one plane (difficult to handle/fixture).
5. M_a consists of only fixture elements (assembly of two fixtures is not considered).

See Figure 11 for example. The configuration, {2, 3, *f1*, *f2*}, satisfies the first condition, thus stop_de returns *false*, subject to further decomposition. On the other hand, the other configuration, {1, 2}, satisfies none of the first four conditions, the decomposition is stopped for this configuration.

AND/OR graph of assembly synthesis

A series of decompositions can be typically represented in a tree as shown in Figure 9. However, the aim of the presented method is to enumerate all such trees, AND/OR graph [18] is adopted to facilitate the assembly synthesis, in which multiple trees share common nodes. Although the AND/OR has been previously used to enumerate assembly sequences for a given assembly design [19], it is augmented in this paper in order to embody joint assignments. Figure 14 shows a partial AND/OR graph of assembly synthesis [1] for the 2D rectangular box shown in Figure 1. Each node in white background contains a configuration, ($M_a \subseteq M_0$), and each node in black background contains joint assignment $\gamma_i : CS_i \mapsto JL$. A set of three lines which connects a configuration M_a, joint assignment γ_i, and two sub-configurations (M_b, M_c) is a hyper-edge, represented as (M_a, γ_i, (M_b, M_c)) which is also the representation of a decomposition defined earlier. The AND/OR graph of assembly synthesis is then represented as a triple:

$$AO = (S, J, F) \tag{23}$$

, where S is a set of nodes representing configurations, J is a set of nodes representing joint assignments, and F is a set of hyper-edges (M_a, γ_i, (M_b, M_c)) satisfying the following necessary conditions.

1. stop_de(M_a) = *false*.
2. de(M_a, γ_i, (M_b, M_c)) = *true*. (24)

Then $AO = (S, J, F)$ is recursively defined as:

1. If stop_de(M_0) = false, $M_0 \in S$.
2. For $\forall M_a \in S$, if $\exists \gamma_i$, M_b, M_c such that $f = (M_a, \gamma_i, (M_b, M_c))$ satisfies necessary conditions (24), then $\gamma_i \in J$, M_b, $M_c \in S$ and $f \in F$.
3. No element is in S, J and F, unless it can be obtained by using rules 1 and 2. (25)

The recursive definition in Equation (25) can be easily transformed to an algorithm *build_AO* that generates *AO* from initial configuration and joint library by recursively decomposing a configuration into two sub-configurations [1], whose details are omitted due to space limitation. Using stop_de and de as defined earlier, one can run *build_AO* with any 3D configurations to enumerate all possible assemblies (decompositions and joint assignments), fixture schemes and

§ The result has been reduced to the Row Reduced Echelon Form for easy interpretation.

accompanying assembly sequences that satisfy the in-process dimensional adjustability and proper part constraint.

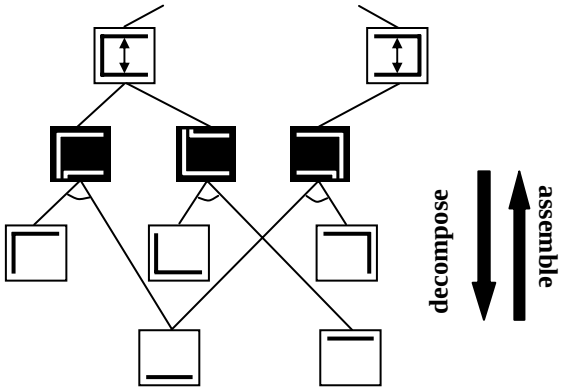

Figure 14. A part of the AND/OR graph for the 2-D rectangular box in Figure 1.

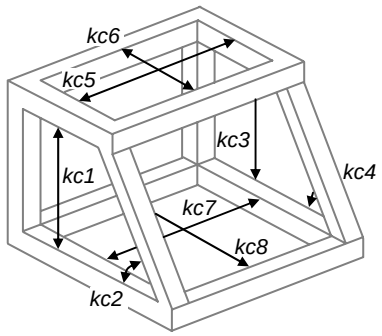

Figure 15. A frame structure with eight KCs.

CASE STUDY

A frame structure in Figure 15 is decomposed based on Equation (25). Only joint types in Figure 16 are assigned to broken connections as required by the decomposition rule. In order to reduce the size of the AND/OR graph, when several joint assignments satisfy decomposition rule for a given decomposition, one with minimum under-constraint is included in the AND/OR graph. Still, the decomposition rule produced a large AND/OR graph with 19962 nodes representing configurations and 143269 hyper-edges, which contains about 8.4 billion trees. However, using brute search starting from the terminal nodes (either part or fixture that satisfies stop_de), non-dominated solution trees for multi-objectives can be identified. Based on four objectives – the number of parts, the number of fixtures, the depth of the tree and the total under-constraints, only 90 trees are found to be non-dominated. Associated cost vectors for these non-dominated solution trees are listed in Table 1. The number of fixtures and under-constraints shows a strong correlation, because the more fixtures are used, the more DOFs should be constrained when initially placing a part on each fixture. From Figure 17 to 23, some of non-dominated solution trees and their corresponding assembly designs are presented. In solution trees, a node with a capital letter represents a part (marked with the same letter in the following assembly design), and a node marked with "fx" with a number represents a fixture. A black node represents a joint assignment and the number within the node represents, $uc(\gamma_d)$, the number of under-constraints for the joint assignment.

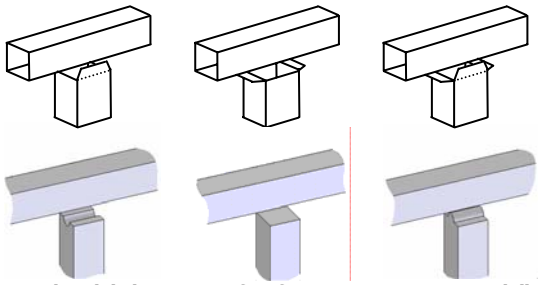

Figure 16. (top) joint types for frame sturcture, and (bottom) their graphical representation used in results.

Table 1. Non-dominated cost vectors and the number of corresponding non-dominated solution trees for the frame structure shown in Figure 15.

Objectives				No. of solution trees
No. of parts	No. of fixture	Depth of tree	No. of under-constraints	
7	2	7	6	16
8	4	5	24	24
8	1	8	3	46
6	3	6	11	4
Total				90

Figure 17 shows a non-dominated solution tree, and corresponding assembly design and sequence, which has 7 parts, 2 fixtures, the depth of 7, and 6 under-constraints. The coordinate frame shown by each joint shows DOFs constrained by the joint in black and un-constrained DOFs in gray. In the figure, fx1 controls *kc3*, *kc4*, *kc7* and *kc8*, and fx2 controls *kc1*, *kc2*, *kc5* and *kc6*. The assembly sequence is as follows:

1. Locate G on fx2. Three KCs related to G; *kc1*, *kc2* and *kc5* are constrained by fx2. In order to uniquely locate G, fx2 should constrain the other three DOFs (the number of under-constraints) in addition to those required by the KCs.
2. Assemble F on G-fx2. Only one KC, *kc6* is required for F, which is fixed by fx2. The other five DOFs are constrained by the lap-butt joints with G, thus there is no under-constraint to be controlled additionally.
3. Locate G on fx2. Three KCs related to G; *kc1*, *kc2* and *kc5* are constrained by fx2. In order to uniquely locate G, fx2 should constrain the other three DOFs in addition to those required by the three KCs.
4. Assemble F on G-fx2. Only one KC, *kc6* is required for F, which is fixed by fx2. The other five DOFs are constrained by the lap-butt joints with G, thus there is no under-constraint to be controlled additionally.
5. In parallel with step 4, place C on fx1. All four KCs related to C are fixed on fx1. The other two DOFs should be constrained additionally.

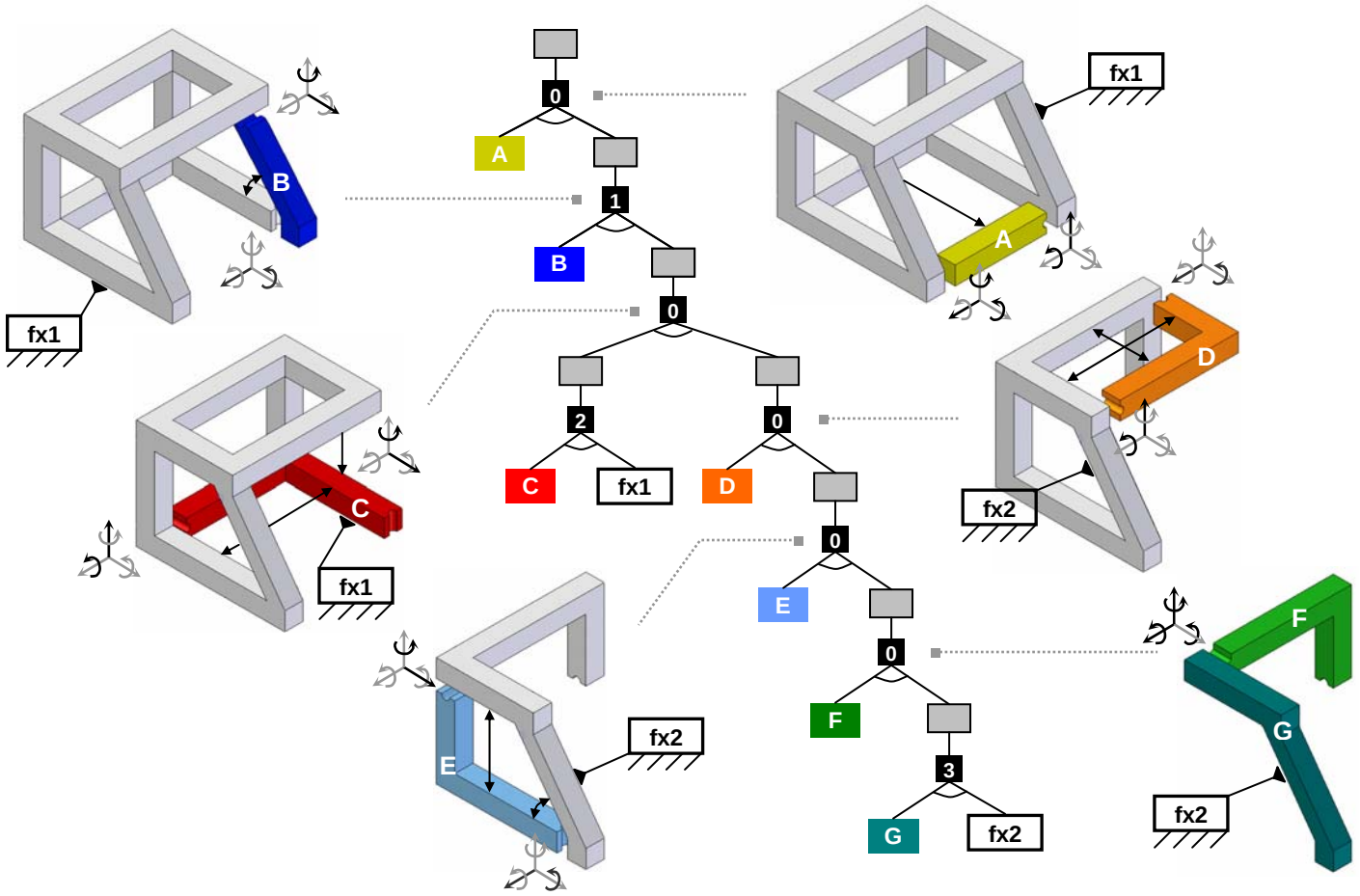

Figure 17. One of 24 non-dominated solution trees whose cost vector is (7, 2, 7, 6).

6. Assemble D-E-F-G on C-fx1, where *kc3* and *kc7* are realized by fx1. The other four DOFs are constrained by the lap joint from F to C and another lap joint from C to E.
7. Assemble B on C-D-E-F-G-fx1, where *kc4* is realized. The lap joint from B to D and the lap joint from C to B constrains four DOFs, thus one DOF should be constrained additionally.
8. Assemble A on B-C-D-E-F-G-fx1, realizing *kc8*. The other five DOFs are fully constrained by one lap joint and one butt joint of A. All assembly steps are now completed.

Figure 18 shows a non-dominated solution tree, which has 8 parts, 4 fixtures, the depth of 5, and 24 under-constraints. This tree has the minimum depth and has a few parallel steps. For this reason, the tree is suitable for parallelized and short cycle time production. The price it pays is the many fixtures required to realize KCs in parallel. On the other hand, the tree shown in Figure 20 is completely serial, using only one fixture to control all the KCs. Because there are less fixtures, many DOFs are constrained by joints between parts, thus yielding mere three under-constraints throughout the assembly. Instead, the production would require a longer cycle time.

There are only four trees that have the non-dominated cost vector of (6, 3, 6, 11). Two of these are shown as a AND/OR graph in Figure 22, which contains two different assembly sequences to build the assembly design shown in Figure 24 (note the OR relation between the two hyper-edges from the top node). The other two trees for the same cost vector are mirror images of ones shown in Figure 22, which has corresponding assembly design that is also mirror image of one shown in Figure 23. Whereas other non-dominated solution trees have one or more decomposition(s) solely to remove an un-planar part, a closed loop or a T-joint required by stop_de, without breaking a KC, all the decompositions in these trees have been made to remove KCs. In other words, these trees show the most efficient way to remove KCs in terms of the number of decompositions. As a result, these solutions have minimum number of parts, 6.

SUMMARY AND DISCUSSION

This paper presents an integrated synthesis approach of assembly and fixture scheme. Starting with initial geometry, the decomposition rule is applied recursively, to obtain all feasible "designs" (assembly designs, fixture scheme, and assembly sequence), such that every assembly step can be free of over-constraints. An example with a simple space frame is presented to demonstrate the method. Considering the number of parts, the number of fixtures, the depth of assembly tree, and the

number of under-constrains as objectives to minimize, the Pareto optimal solutions are obtained by searching the enumerated AND/OR graph of assembly synthesis that contains all feasible designs. Non-dominated solution trees show trade-offs among the assembly design, the number of fixtures and the assembly sequence. An assembly with many connections among parts and many KCs is likely to have less parallel assembly sequence, because it is likely that the assembling two large subassemblies at the later stage would have many joints between the two subassemblies and realize many KCs at one step, thus more likely to have over-constraints. It has been also shown that a more parallelized assembly sequence would require more fixtures.

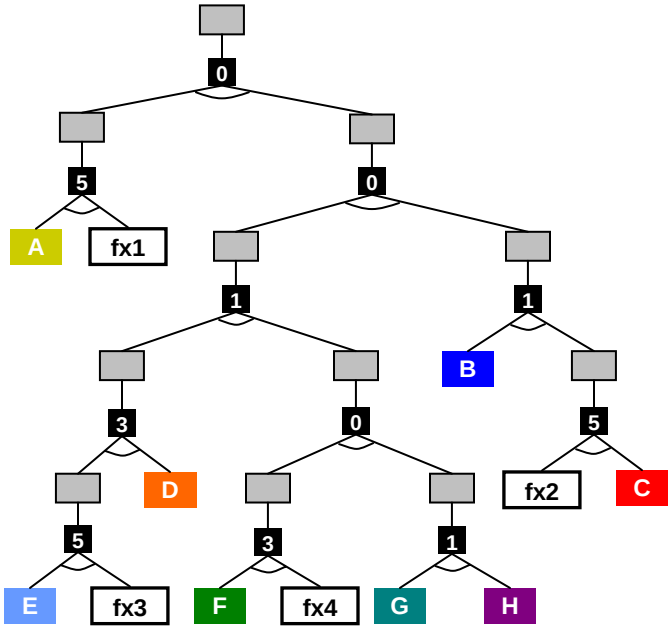

Figure 18. One of 24 non-dominated solution trees whose cost vector is (8, 4, 5, 24).

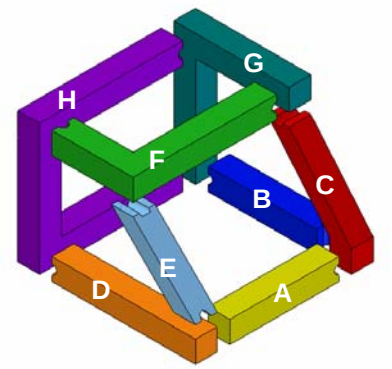

Figure 19. The assembly design matching the non-dominated solution tree shown in Figure 18.

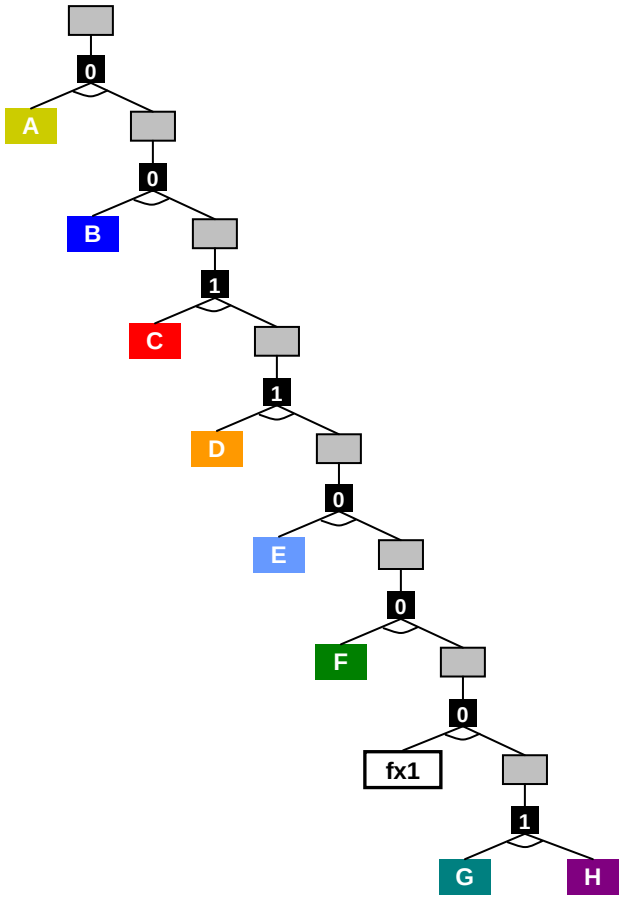

Figure 20. One of 46 non-dominated solution trees whose cost vector is (8, 1, 8, 3).

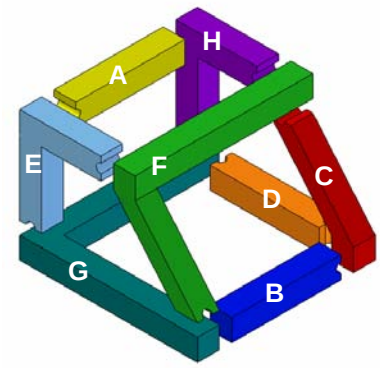

Figure 21. The assembly design matching the non-dominated solution tree shown in Figure 20.

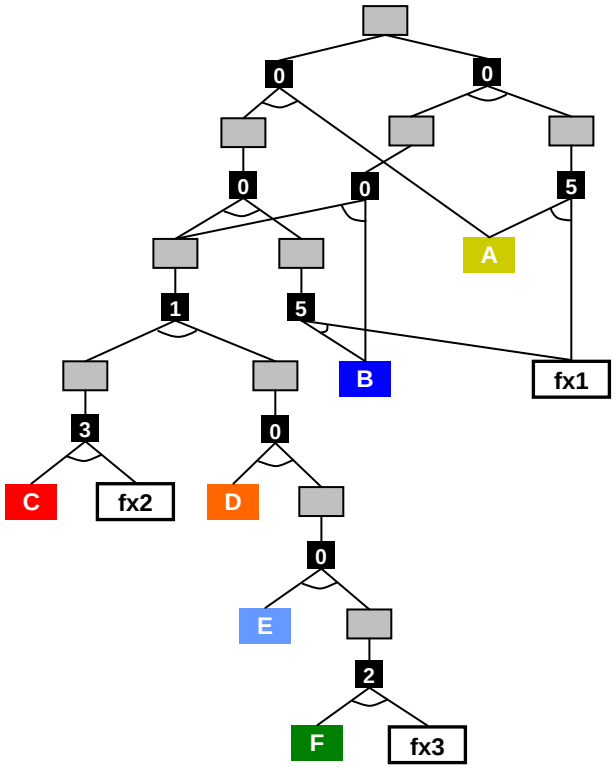

Figure 22. Two of four non-dominated solution trees whose cost vector is (6, 3, 6, 11).

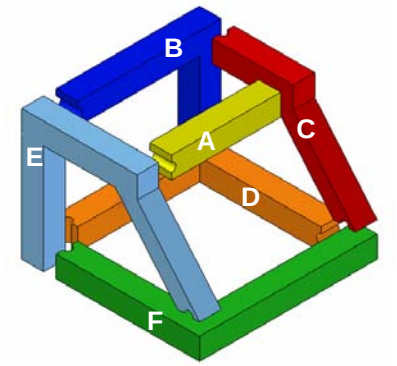

Figure 23. The assembly design matching the trees in Figure 22, which contains two assembly sequences.

Due to the enumerative nature of the presented approach, the amount of computation for complex assemblies would be inevitably large. Taking the number of prospective decompositions that must be analyzed as a measure of the amount of computation, Homem de Mello and Sanderson [20] showed that computational complexity is generally $O(3^n)$ in generating AND/OR graph of assembly sequence for given assembly designs, where n is the number of parts. Although the amount of computation for assembly synthesis would largely dependent on the number of KCs, available joint types and the manufacturability criteria, the worst case (when every part consists of only one member) would be comparable to that of assembly sequence generation. Some of the actual computation times are shown in Table 2[**], which exhibits a rapid growth in the computation time with the increased number of members and KCs. The present method, therefore, would most effectively be integrated in the design process if it is applied to subassemblies of a product decomposed by other means. Alternatively, the repeated applications of the method to subassemblies with incremental refinement of members (from coarse to fine) would also be effective to manage the complexity.

Table 2. Amount of computation for (1) the example in Figure 15, (2) without KCs and (3) with 5 members and 4 KCs removed.

	1	2	3
No. of members	12	12	7
No. of KCs	8	0	4
No. of decompositions	143269	32240	2621
No. of solution trees	8.38×10^9	1.03×10^8	9.92×10^3
computation time [sec]	2482	293.0	8.125

ACKNOWLEDGMENTS

This work has been supported by the National Science Foundation with a CAREER Award (DMI-9984606) and Toyota Motor Company. Any opinions, 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.

REFERENCES

[1] Lee, B. and Saitou, K., 2003, "Decomposition-based assembly synthesis for in-process dimensional adjustability," *ASME Journal of Mechanical Design*, vol. 125, no. 3, pp. 464-473.

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

[3] Blanding, D. L., 1999, *Exact Constraint: Machine Design Using Kinematic Principles*, ASME Press, NY.

[4] Kriegel, J. M., 1995, "Exact constraint design," *Mechanical Engineering*, vol. 117, no. 5, pp. 88-90.

[5] Lee, B. and Saitou, K., 2004, "Three-dimensional synthesis for robust dimensional integrity based on screw theory," Proceedings of the *Fifth International Symposium on Tools and Methods of Competitive Engineering*, Lausanne, Switzerland, April 13-17, vol. 2, pp. 585-596.

[6] Ball, R. S., 1900, *A Treatise on the Theory of Screws*, Cambridge University Press.

[7] Whitehead, T. N., 1954, *The Design and Use of Instruments and Accurate Mechanism*, Dover Publications, New York, NY.

[8] Kamm, L. J., 1990, *Designing Cost-Effective Mechanisms*, McGraw-Hill.

[9] Downey, K, Parkinson, A. R and Chase, K. W., 2003, "An introduction to smart assemblies for robust design,"

[**] Test runs were conducted on a PC with 3.2 GHz Intel® Pentium 4 ® processor with 1GB RAM.

Research in Engineering Design, vol. 14, no. 4, pp. 236-246.
[10] Waldron, K. J., 1966, "The constraint analysis of mechanisms," *Journal of Mechanisms*, vol. 1, no. 2, pp. 101-114.
[11] Adams, J. D. and Whitney, D. E., 2001, "Application of screw theory to constraint analysis of assemblies joined by features", *ASME Journal of Mechanical Design*, vol. 123, no. 1, pp. 26-32.
[12] Asada, H. and By, A. B, 1985, "Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures," *IEEE Journal of Robotics and Automation*, vol. RA-1, no. 2, pp. 86-94.
[13] 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 Conferences*, Irvine, CA, Paper no. 96-DETC/DTM-1506.
[14] Woo, L. and Freudenstein, F., 1970, "Application of line geometry to theoretical kinematics and the kinematic analysis of mechanical systems," *Journal of Mechanisms*, vol. 5, pp. 417-460.
[15] Hunt, K. H., 1978, *Kinematic Geometry of Mechanisms*, Oxford University Press.
[16] Roth, B., 1983, "Screws, motors, and wrenches that can not be bought in a hardware store," *Robotics Research, The first symposium*, MIT Press, Cambridge, MA, pp. 679-735.
[17] Foulds, L. R., 1991, *Graph Theory Applications*, Springer-Verlag, New York, NY.
[18] Nilsson, N. J., 1980, *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA.
[19] 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.
[20] 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.