'92 KACC 1992. 10. 19～21

A New Method for Generating Assembly Sequences

Jong Hun Park and Myung Jin Chung

Dept. of Electrical Engineering,
Korea Advanced Institute of Science and Technology,
373-1, Kusong-dong, Yusong-gu, Taejon, 305-701, Korea

Abstract – Current available methods for generating assembly sequences have a large undesirable search-space. This paper presents a method for reducing the search-space. The method acquires explicitly assembly constraints caused by not only the geometry of parts but also the connectivity between the parts, in simplified form. Then the method generates assembly sequences without searching undesirable tasks using the assembly constraints. If these undesirable tasks are excluded, assembly sequences can be generated by searching only a fraction of all assembly tasks for a product and its subassemblies.

I. INTRODUCTION

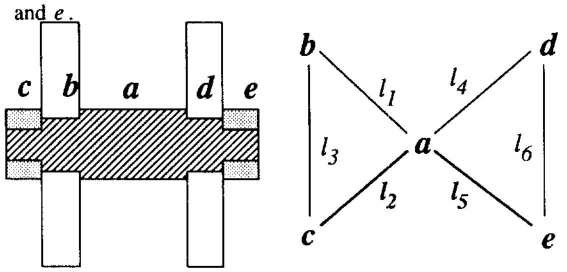
In manufacturing industries, assembly is an important application area in robotics since assembly accounts for major portion of not only the manufacturing cost of a product but also the labor force[1]. Many important things in assembly are determined by the sequence of assembly tasks[2]. Hence, the assembly sequence must be carefully selected so that it produces the best output. For optimal assembly sequencing, a planner must have the ability to search all assembly sequences.

To generate assembly sequences, current available methods[2-6] search either all assembly tasks[2] or disassembly tasks[3-6], and then they exclude undesirable ones. However, the number of tasks searched by these methods explosively increases as the number of parts rises. Moreover, considerable parts of the tasks searched by these methods are not desirable. The ratio of undesirable tasks to desirable ones greatly rises, as the complexity of a product and the number of parts increase.

The objective of this paper is to present an efficient method for generating assembly sequences. Assembly sequencies are generated by recursively disassembling a target product and its subassemblies without violating assembly constraints. To increase the efficiency, the proposed method 1) excludes redundancies in assembly constraints, and 2) excludes undesirable tasks without searching them. Since the search-space of the proposed method is greatly reduced, application ranges of the method are expanded notably.

II. BACKGROUND

A product can be modeled as a connected graph $G(P,L)$, where P is a set of all parts of the product, and L is a set of all liaisons of the product in which the liaisons indicate contact relationships between the parts[2]. Fig. 1 shows a graph model of an electric grinder. Blades b and d of the grinder are fixed to the motor shaft a by bolts c and e.



(a) Shape of a grinder. (b) Model of a grinder.
Fig. 1. The graph model of a grinder.

A *subassembly* is defined as a vertex-induced connected subgraph of the graph model G. Since a subassembly can be expressed uniquely by a set of parts comprising the subassembly, the graph and set representations of a subassembly are interchangeably used. A subassembly is said to be *reachable* if it can successfully become a target product by sequentially adding the missing parts. For instance, a subassembly $\{a,c,d\}$ of the grinder is not reachable because part b cannot be added to the subassembly.

An *assembly task* is defined as an action which joins a subassembly to another subassembly to produce a large subassembly. A task joining more than two disconnected subassemblies is not considered since the task causes assembly sequences to be not *contact coherent*[7]. There must be at least one liaison between the pair of subassemblies joined by an assembly task in model G. Assembly task which joins a subassembly M to another subassembly B is represented as $M @ B$.

The subassembly which is manipulated by such assembly equipment as a robot and a human worker is called a *mating subassembly*. The subassembly which is held by such assembly equipment as the jig, the fixture, and the work table is called a *base subassembly*. Since a mating subassembly is manipulated by assembly equipment it requires special properties. Parts of a mating subassembly should be very stably joined each other so that they cannot be separated by little force in any orientation. A subassembly whose parts are not very stably joined to each other is not suitable for a mating subassembly because it needs extra-costs for stablizing the parts during manipulation. Also, a mating subassembly should be reachable because unreachable subassembly cannot become a target product. Hence, we define reachable and very stable subassemblies as mating subassemblies.

An assembly task is called physically *infeasible* if it cannot be executed successfully because of the topology, geometry, and dimension of parts. Also, an assembly task is called logically infeasible if the pair of subassemblies joined by the task are not connected in model G because it is not contact coherent. The assembly constraints caused by the geometry of parts are called *G-constraints*. The assembly constraints caused by the contact coherence are called *C-constraints*.

G-constraints have been acquired by either systematic question-and-answers with a designer[2,3] or computer algorithms[5,6]. G-constraints have been represented either as the precedence relationships[2,3,5] or the infeasibility conditions[6] as shown in Table 1. However, C-constraints have been acquired by exhaustive methods such as the connectivity tests[2,3] or the cut-sets[4]. De Fazio and Whitney[2] excluded incoherent assembly tasks by checking the connectivity between the subassemblies joined by each task. Homem de Mello and Sanderson[4] excluded incoherent assembly tasks by using the cut-set technique.

const.	exhaustive rep.	simplified rep.
G-const.	feasibility of tasks	precedence rel. (>, >=) infeasibility cond.(@)
C-const.	cut-sets connectivity tests	infeasibility cond. (@)

Table 1. Representations of G-constraints and C-constraints. Operators ">" and ">=" indicate "must precede" and "must precede or concurrent with", respectively. Operator "@" indicates "cannot be joined to". The infeasibility representation of C-constraints will be presented in Section III.

Assembly constraints may have many redundancies because G-constraints and C-constraints are independent of each other. Also, assembly constraints may have many redundancies due to their inherent properties. For instance, an infeasibility condition $M @ Q$, which indicates that a mating subassembly M cannot be joined to a subassembly with Q, has the following properties:

[**Property 1**] An infeasibility condition $M @ Q$ can cover an infeasibility condition $M @ Q^*$, where Q^* is a superset of Q.

[**Property 2**] If a part $p \in P$ cannot be joined to a set of parts $Q \subset P$ due to G-constraints, subassemblies which contain Q except part p are not reachable.

III. SIMPLIFIED ASSEMBLY CONSTRAINTS

An assembly task is executed by joining a mating subassembly to a base subassembly. Hence, only the infeasibility conditions of every mating subassembly are needed as assembly constraints. Infeasibility conditions of all subassemblies are not necessarily required as the assembly constraints. Since only a fraction of subassemblies are suitable for mating subassemblies, the introduction of mating subassemblies may greatly reduce the complexity in acquiring assembly constraints as well as in generating assembly sequences. The rest of this section will summarize methods for acquiring the simplified assembly constraints by removing redundancies in the assembly constraints.

A. THE SIMPLIFIED G-CONSTRAINTS

A mating subassembly cannot be joined to a base subassembly if the base subassembly blocks all the paths of the mating subassembly. We assume that a mating subassembly M has a finite number of paths. A mating subassembly M cannot be joined to a subassembly which contains at least one part for every path that blocks the path. Hence, one obvious method for acquiring G-constraints of a mating subassembly M is to enumerate all subassemblies that block all the paths of M. However, this exhaustive method is so inefficient that it cannot be used as the number of parts increases.

To block all the paths of a mating subassembly M, subassemblies must contain a set Q which contains one part for every path of M to block the path. G-constraints of a mating subassembly M can be simplified by searching minimal sets, in which a set Q satisfies the following three conditions:

1) A set Q contains only one part for every path of M to block the path;
2) A set Q must be contained by a subassembly $S \cap M = \varnothing$. If there is no subassembly which contains Q, then infeasibility condition $M @ Q$ is useless due to C-constraints.
3) Subassemblies containing a set Q except the parts of M do not contain more than one part which blocks any path of M. This is due to **Property 1**.

To show the procedure for acquiring the simplified G-constraints, we adopt the grinder of Fig. 1 as a simple example. The meaningful paths of part a are the left and right directions since part a is contacted to parts b, c, d, and e in a 'fits' condition. Part a is blocked by parts b and c when it is removed to the left. Part a is blocked by parts d and e when it is removed to the right. Hence, part a cannot be joined to subassemblies with $\{b,d\}$, $\{b,e\}$, $\{c,d\}$, or $\{c,e\}$. However, there is no subassembly that contains any of these sets without containing part a. That is, G-constraints of part a cannot contribute as assembly constraints because the G-constraints of part a can be covered by C-constraints.

Part b is blocked by part c when it is removed to the left. Part b is blocked by parts a, d, and e when it is removed to the right. Hence, part b cannot be joined to subassemblies with $\{a,c\}$, $\{c,d\}$, or $\{c,e\}$. However, subassemblies which contain set $\{c,d\}$ or $\{c,e\}$ without containing part b always contain set $\{a,c\}$. So, the G-constraints of part b can be simplified into one infeasibility condition $\{b\} @ \{a,c\}$. Similarly, the simplified G-constraint of part d is acquired. The G-constraint of part d is $\{d\} @ \{a,e\}$.

G-constraints of parts c and e do not exist because they have unblocked paths. Once G-constraints of every part have been acquired, the reachability of all subassemblies can be determined due to **Property 2**. Hence, the mating subassemblies of the grinder can be acquired from liaison graph model using the G-constraints of every part. Among subassemblies of the grinder, reachable and very stable subassemblies with more than one part are $\{a,b,c\}$ and $\{a,d,e\}$. G-constraints of these mating subassemblies do not exist because they have unblocked paths. Hence, the simplified G-constraints of the grinder are $\{b\} @ \{a,c\}$ and $\{d\} @ \{a,e\}$.

B. THE SIMPLIFIED C-CONSTRAINTS

One obvious method for acquiring C-constraints of a mating subassembly M is to enumerate all subassemblies that cause the separation of M from the subassemblies to be not contact coherent. However, this exhaustive method is so complex that it is not practical as the number of parts increases, even though currently the norm.

If a mating subassembly M cannot be separated from a subassembly S due to the contact coherence, then there is at least a pair of parts (a,b) which are disconnected by separating M from subassembly $S \supset (\{a,b\} \cup M)$. Hence, the C-constraint between a maing subassembly M and its neighbor parts a and b can be expressed by a sentence: "a mating subassembly M cannot be separated from a subassembly $S \supset (\{a,b\} \cup M)$, if S does not contain an elementary chain which connects parts a and b without passing any part of M".

The C-constraint between a mating subassembly M and its neighbor parts a and b can be simplified using the following fact: "a chain λ between parts a and b is always broken if another chain λ', which is a subset of λ, is broken". But, some chains between parts a and b cannot be broken because parts of the chains, except parts a and b, cannot be removed due to G-constraints. Hence, only chains that can be broken without violating G-constraints, and which are not superset of any other chain are meaningful in acquiring C-constraints. If a family of these meaningful chains is represented as Λ, then the C-constraint between a mating subassembly M and its neighbor parts a and b can be represented as the infeasibility condition:

$$M \,@\, (\{a,b\} \,\&\, \bar{\lambda})\ , \quad \text{for all } \lambda \in \Lambda\ .$$

In this expression, a chain is expressed by a set of parts comprising the chain since the order of parts in a chain is not critical in determining the existence of the chain. The operators "&" and "$^{-}$" indicate "and" and "not exist", respectively.

As a simple example, the procedure for acquiring C-constraints of the grinder in Fig. 1 will be shown. The acquisition of the simplified C-constraints of part a begins by searching its neighbor parts from a liaison graph model of the grinder. Parts b, c, d, and e are neighbors of part a. There are six pairs of a's neighbor parts: (b,c), (b,d), (b,e), (c,d), (c,e), and (d,e). Among them, pairs (b,c) and (d,e) are discarded because parts of these pairs cannot be disconnected by removing part a. However, parts of pairs (b,d), (b,e), (c,d), and (c,e) are always disconnected by removing part a because all the chains between the parts of these pairs pass part a. Hence, C-constraints of part a are $\{a\}@\{b,d\}$, $\{a\}@\{b,e\}$, $\{a\}@\{c,d\}$, and $\{a\}@\{c,e\}$. These infeasibility conditions can be compressed into one infeasibility condition using "or" operator:

$$\{a\} \,@\, \left[\, \{b,d\} \,|\, \{b,e\} \,|\, \{c,d\} \,|\, \{c,e\} \,\right].$$

This infeasibility condition means that part a cannot be joined to a subassembly which contains any of sets $\{b,d\}$, $\{b,e\}$, $\{c,d\}$, and $\{c,e\}$.

Parts a and c, which are neighbors of part b, cannot be disconnected by removing part b since these parts are connected by liaison l_2. Also, c's neighbor parts a and b cannot be disconnected by removing part c since these parts are connected by liaison l_1. Similarly, d's neighbor parts a and e, e's neighbor parts a and d cannot be disconnected by removing parts d and e, respectively. Hence, C-constraints of parts b, c, d, and e do not exist.

C-constraints of mating subassemblies $\{a,b,c\}$ and $\{a,d,e\}$ do not exist since neighbor parts of these mating subassemblies cannot be disconnected by removing these subassemblies. That is, parts b and c, which are neighbors of a mating subassembly $\{a,d,e\}$, cannot be disconnected by removing $\{a,d,e\}$. Parts d and e, which are neighbors of a mating subassembly $\{a,b,c\}$, cannot be disconnected by removing $\{a,b,c\}$. Hence, the simplified C-constraints of the grinder are

$$\{a\} \,@\, \left[\, \{b,d\} \,|\, \{b,e\} \,|\, \{c,d\} \,|\, \{c,e\} \,\right].$$

IV. GENERATION OF ASSEMBLY SEQUENCES

Once the mating subassemblies and their simplified assembly constraints have been acquired, assembly sequences are generated by recording all disassembly tasks that separate unconstrained mating subassemblies from a target product and its subassemblies. Since the number of assembly sequences for a product is very large, the assembly sequences are compactly represented as a labeled, directed graph. In a directed graph, a task which joins a mating subassembly M to a base subassembly B is represented as a labeled link:

$$M \cup B \;\xrightarrow{\;M\;}\; B\,.$$

A procedure for generating assembly sequences without searching undesirable tasks using assembly constraints is as follows:

Given :

(1) The mating subassemblies of a target product P. Let R be a set of the mating subassemblies.

(2) Simplified assembly constrints of every mating subassembly.

Procedure :

```
(1) Initialize global variables.
    1.1) Create and clear a labeled, directed graph, LDG.
    1.2) Create a family of subassemblies, $FOS$. Let a target
         product $P$ be an initial element of $FOS$.
(2) Among the non-disassembled elements of $FOS$, choose a
    subassembly $S$ which has the largest number of parts. If
    subassembly $S$ is an empty set, then Stop.
(3) Create a node of $S$ if the node does not exist in LDG.
(4) Select mating subassemblies of $S$. A subassembly $M \in R$
    cannot be a mating subassembly of $S$ if $M$ is not a subset
    of $S$. Hence, the mating subassemblies of $S$ are acquired
    by selecting subassemblies $M \in R$ which are subsets of $S$.
(5) Select unconstrained mating subassemblies of $S$. That is,
    mating subassemblies of $S$ whose infeasibility conditions of
    both G-constraints and C-constraints are null are selected.
(6) For each mating subassembly $M$ which has been selected in
    Step (6),
    6.1) Record an assembly task $M$ @ $(S - M)$ in LDG. The
```

two steps for recording the assembly task are as follows:

a) Create a node of $S-M$ if the node does not exist in LDG.

b) Connect the node of $S-M$ to the node of S by a link whose label is M.

6.2) Insert M and $S-M$ into FOS.

(7) Go to Step (2).

As a simple example, consider the grinder in Fig. 1. Once the mating subassemblies of the grinder and their simplified assembly constraints have been acquired as shown in Section III. The procedure for generating assembly sequences begins by creating a node of the grinder. Among the mating subassemblies of the grinder, only subassemblies $\{c\}$, $\{e\}$, $\{a,b,c\}$, and $\{a,d,e\}$ are unconstrained. The other mating subassemblies $\{b\}$ and $\{d\}$ cannot be separated from the grinder due to the assembly constraints. Hence, desirable assembly tasks for the grinder are $\{c\}@\{a,b,d,e\}$, $\{e\}@\{a,b,c,d\}$, $\{a,b,c\}@\{d,e\}$, and $\{a,d,e\}@\{b,c\}$. These tasks are represented graphically as the first level in Fig. 2. From these tasks, subassemblies $\{c\}$, $\{a,b,d,e\}$, $\{e\}$, $\{a,b,c,d\}$, $\{a,b,c\}$, $\{d,e\}$, $\{a,d,e\}$, and $\{b,c\}$, are newly enrolled in FOS.

Among the non-disassembled elements of FOS, a subassembly $\{a,b,c,d\}$ is selected to be disassembled because it contains the largest number of parts. The mating subassemblies of a subassembly $\{a,b,c,d\}$ are acquired from the mating subassemblies of the grinder by removing the subassemblies with part e. The set of mating subassemblies of $\{a,b,c,d\}$ is

$$R' = \left\{ \{a\},\{b\},\{c\},\{d\},\{a,b,c\} \right\}.$$

Of the mating subassemblies of the grinder, subassemblies $\{e\}$ and $\{a,d,e\}$ cannot be mating subassemblies of $\{a,b,c,d\}$ because they are not subsets of $\{a,b,c,d\}$.

Among the mating subassemblies of $\{a,b,c,d\}$, subassemblies $\{c\}$, $\{d\}$, and $\{a,b,c\}$ are not constrained. The other mating subassemblies $\{a\}$ and $\{b\}$ cannot be separated from subassembly $\{a,b,c,d\}$ because their infeasibility conditions are not null in subassembly $\{a,b,c,d\}$. Hence, the desirable assembly tasks for a subassembly $\{a,b,c,d\}$ are $\{c\}@\{a,b,d\}$, $\{d\}@\{a,b,c\}$, and $\{a,b,c\}@\{d\}$. These tasks are represented graphically as the second level in Fig. 2. From these tasks, subassemblies $\{a,b,d\}$ and $\{d\}$ are enrolled as new elements of FOS.

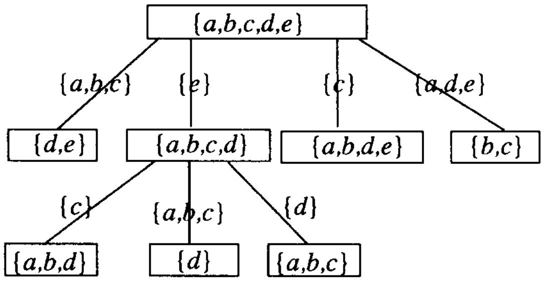


Fig. 2. Disassembly tasks for the grinder and $\{a,b,c,d\}$.

Repeating the procedure for disassembling the other subassemblies until all the elements of FOS are completely disassembled, then assembly sequences for the grinder are represented compactly as shown in Fig. 3. In this figure, the label of each directed link is omitted for the convenience of representation.

Once assembly sequences for a product have been generated, the optimal sequence may be selected

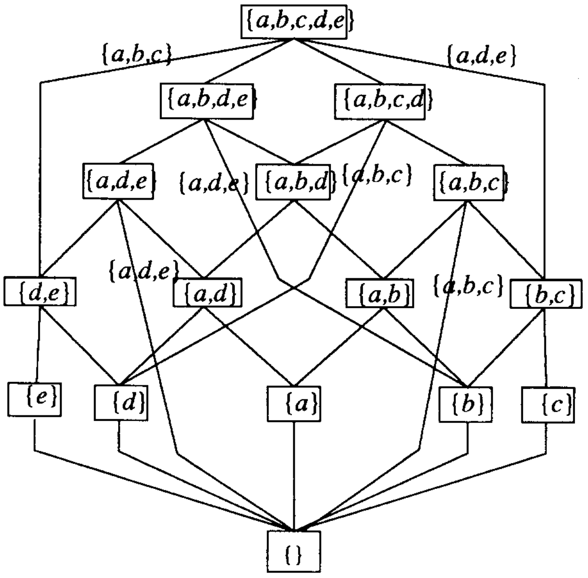


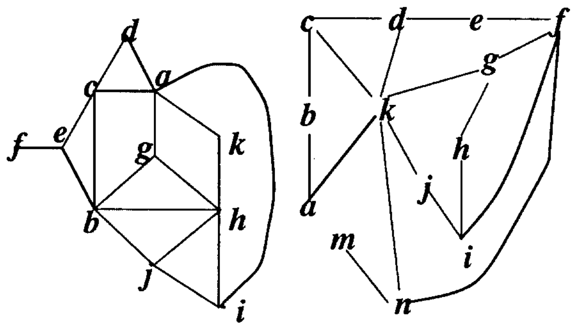
Fig. 3. Assembly sequences for the grinder.

automatically using a graph search technique[9], and appropriate evaluation criteria[8]. An assembly sequence can be selected manually by a designer after reducing the number of assembly sequences by imposing interactively additional assembly constraints as in Baldwin *et al*[3].

V. CONCLUSION

The generation of all assembly sequences is so complex that it is nearly impossible to generate them without using efficient algorithms, as the number of parts as well as the complexity of a product increase. This paper has presented an efficient method for generating assembly sequences without searching undesirable tasks. Not to search undesirable tasks it has acquired assembly constraints from a graph model of a target product. Then, it has generated the assembly sequences by separating recursively unconstrained mating subassemblies from a target product and its subassemblies. To increase the efficiency in generating assembly sequences, it has simplified the assembly constraints by excluding their redundancies.

Table 2 shows both the number of cut-sets *vs* the number of simplified C-constraints and the number of physically infeasible assembly tasks *vs* the number of simplified G-constraints for three products, *i.e.* the grinder in Fig. 1, the AFI in Fig. 4-a of Reference [2], and the automobile alternator in Fig. 4-b of Reference [10]. Table 3 shows the numbers of tasks for these products searched by the method of De Fazio and Whitney[2], by the method of Homem de Mello and Sanderson[4], and by the proposed method.



(a) Model of AFI. (b) Model of alternator.

Fig. 4. Models of AFI and an alternator.

These tables show the following important things :

1) The number of cut-sets increases explosively as the number of parts rises.
2) Most of the assembly tasks that are acquired by cut-sets are infeasible due to G-constraints.
3) The numerous physically infeasible assembly tasks, which are exhaustive representation of the G-constraints, can be compactly represented as the simplified G-constraints.
4) The numerous cut-sets, which are exhaustive representation of the C-constraints, can be compactly represented as the simplified C-constraints.
4) The number of tasks searched by the proposed method is greatly less than that of any other method.

Product	Cut-Sets	*C-const.*	Infeasible T.	*G-const.*
grinder	26	*4*	6	*2*
AFI	1063	*48*	842	*27*
alternator	949	*27*	662	*37*

Table 2. The numbers of cut-sets, physically infeasible assembly tasks, and simplified assembly constraints.

Product	De Fazio's M.	Homem's M.	*Proposed M.*
grinder	26	35	*18*
AFI	963	578	*154*
alternator	959	1134	*123*

Table 3. The numbers of tasks searched by the method of De Fazio and Whitney, by the method of Homem de Mello and Sanderson, and by the proposed method.

REFERENCES

1 U. Rembold and R. Dillmann, Computer-Aided Design and Manufacturing Methods and Tools, *Springer-Verlag*, 1986.

2 T.L. De Fazio and D.E. Whitney, "Simplified Generation of All Mechanical Assembly Sequences", *IEEE Trans. on Robotics and Automation*, vol.3, no.6, pp.640-658, 1987.

3 D.F. Baldwin *et al*, "An Integrated Computer Aid for Generating and Evaluating Assembly Sequences for Mechanical Products", *IEEE Trans. on R & A*, vol.7, no.1, 1991.

4 L.S. Homem de Mello and A.C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences", *IEEE Trans. on R & A*, vol.7, no.2, pp.228-240, 1991.

5 Y.F. Huang and C.S.G. Lee, "An Automatic Assembly Planning System", *IEEE Int. Conf. on R & A*, 1990.

6 J.H. Park and M.J. Chung, "Automatic Assembly Planning for a Flexible Assembly System", *IEEE Int. Conf. on System, Man, and Cybernatics*, pp.437-442, 1991.

7 J.D. Wolter, "A Combinatorial Analysis of Enumerative Data Structures for Assembly Planning", *IEEE Int. Conf. on R & A*, pp.611-618, 1991.

8 J.D. Wolter, "On the Automatic Generation of Assembly Plans", *IEEE Int. Conf. on R & A*, pp.62-68, 1989.

9 P.H. Winston, Artificial Intelligence, *Addison Wesley*, 1984.

10 S. Y. Nof ed., Handbook of Industrial Robotics, *John Wiley & Sons*, pp.1043, 1985.

11 N. Christofides, Graph Theory and Algorithm Approach, *Academic Press*, 1975.