EMITTER International Journal of Engineering Technology
Vol. 6, No. 2, December 2018
ISSN: 2443-1168

Assembly Sequence Planning by Probabilistic Tree Transformation

Takeshi Murayama, Yuichi Mine, Hiroshi Fujinaka, Toru Eguchi

Hiroshima University, Hiroshima, Japan
E-mail: murayatk@hiroshima-u.ac.jp, mine@hiroshima-u.ac.jp,
eguchi@hiroshima-u.ac.jp

Abstract

Various types of computer systems including CAD/CAM systems have been introduced in machine industry. Some of the systems can handle assembly sequence planning, however it requires long time for planning. This paper proposes a method of generating assembly sequences efficiently. This method extracts some parts and/or subassemblies whose possibilities of being removed from a product are strong, and tests whether they can be removed without any geometric interference. By performing these operations repeatedly, the method generates a disassembly sequence of the product, and obtains an assembly sequence by reversing it. The extraction of some parts and/or subassemblies is performed, based on probabilistic tree transformation. The authors present a calculation example by using a software tool integrated with a CAD system.

Keywords: Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Assembly Sequence Planning, Graph Theory.

1. INTRODUCTION

CAD (Computer - aided Design) and CAM (Computer - aided manufacturing) systems have been widely introduced in machine industry. These systems are now essential for designing and manufacturing machines. Most of CAM systems aim at planning machining operations and generating so-called NC data for machine tools. Some of the systems can handle assembly sequence planning, however it requires long time for planning.

To reduce the time and cost, many algorithms and tools have been developed for generating assembly sequences automatically[1-13]. Almost all of them are based on a disassembly approach. This approach generates a disassembly sequence by identifying a part or subassembly to be removed repeatedly, and then generates an assembly sequence by reversing the disassembly sequence. In order to identify the part or subassembly to be removed, this approach tests which parts and/or subassemblies can be removed from the product without any geometric interference. The tests for

all the parts and/or subassemblies are computationally very expensive, especially in the case that the paths to remove them are searched for at the tests[10]. Therefore some of the works focus on reducing the number of the tests. In this paper, we propose a method of reducing the number by using the heuristics and probabilistic tree transformation.

2. RELATED WORKS

To reduce the number of the tests, some approaches have been proposed. Simplification rules(e.g., superset and subset rules[4]) can avoid the unnecessary tests, however the number of the remainder(i.e., the necessary tests) is still large especially for the products composed of many parts. Subassembly extraction[5-8] and heuristics[11] are effective to reduce the number of the tests further. Reusing the assembly plans also considerably reduces the number of the tests at redesign or modification design stages[9].

3. ORIGINALITY

To reduce the number of the tests, we propose probabilistic tree transformation, in which a probabilistic technique is introduced into tree transformation in graph theory[14], and we present a method of assembly sequence generation using the probabilistic tree transformation as well as the heuristics. This method extracts some parts and/or subassemblies whose possibilities of being removed are strong, and then performs the tests for only them using CAD data. The extraction of some parts and/or subassemblies is performed, based on the probabilistic tree transformation and heuristics.

4. SYSTEM DESIGN

4.1 Generation of Assembly Sequences

In our method, a product is represented as a part-connectivity graph, G, defined by:

$$G=(V, A) \quad (1)$$

where V is a set of the nodes expressing the parts included in the product, and A is a set of the arcs expressing the connective relations among the parts. Figure 1 shows an example of the part-connectivity graph representation.

The separation or disassembly of the product can be represented by the partition of the part-connectivity graph into subgraphs, each of which represents a part or subassembly, as shown in Fig. 1. The set of the arcs cut by the partition is called cut set. In Fig. 1, cut set {*b, c, d*} corresponds to the separation of the product into two subassemblies {*1, 2*} and {*3, 4, 5*}. By calculating such cut sets recursively, we can partition the part-connectivity graph recursively, and as a result a disassembly sequence can be obtained. We can also obtain an assembly sequence by reversing it, assuming that an assembly operation is the reverse of the disassembly operation. However, at the assembly operation, it is difficult to join three or more subassemblies and/or parts together simultaneously. Therefore, to partition the part-connectivity graph, we use elementary cut sets, each of which separates the

product into two of the subassemblies and parts. Of the two, one that doesn't include a base part can be regarded as a part or subassembly to be removed from the other.

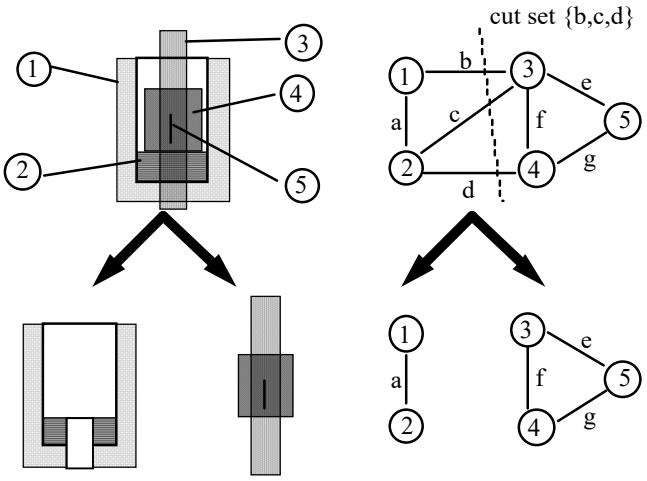


Figure 1. An example of the part-connectivity graph and its separation

Baldwin, et al.[3] calculated all the elementary cut sets(they call them assembly cut sets) to generate all feasible assembly sequences. This brings about considerable computational time if the separability corresponding to every elementary cut set is tested automatically. Therefore, we calculate some elementary cut sets that have strong possibilities of separating the product without any geometric interference, and then performs the tests for only them. This means the reduction of the search space for disassembly sequences.

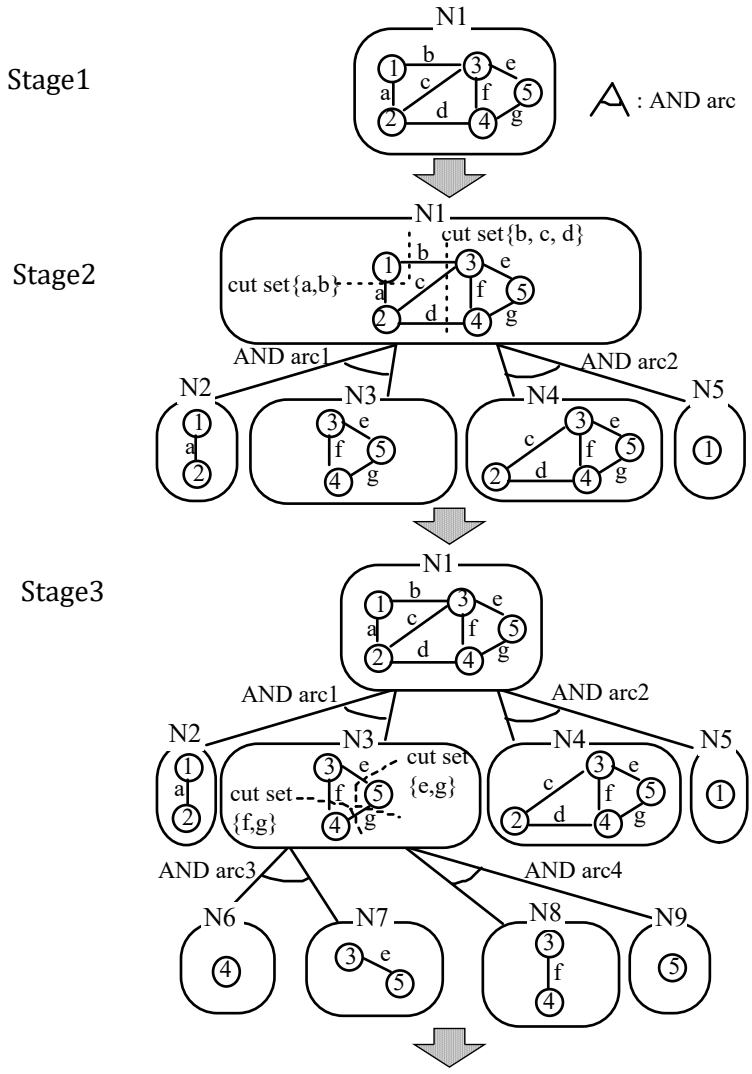


Figure 2. Dissasembly sequence search by calculating cut sets

Figure 2 shows an example of disassembly sequence generation by calculating some elementary cut sets. As shown in this figure, to generate a disassembly sequence, we search an AND/OR graph[15], in which each node corresponds to an object(i.e., product, subassembly, or part) and each AND arc corresponds to the separation of a product or subassembly(i.e., it corresponds to an elementary cut set). We perform the search, based on a modified AO* algorithm. Murayama, et al.[9] described this algorithm in detail. The summary of this is as follows. This algorithm first generates a node, *N1*, representing a product, as shown in Fig. 2. Then, this algorithm calculates some elementary cut sets whose number is given by the operator in advance of searching. In the example shown in Fig. 2, two elementary cut sets {*a, b*} and {*b, c, d*} are calculated. According to the calculated cut sets, this algorithm generates the child nodes and arcs. In Fig. 2, four child nodes(*N2, N3, N4*, and *N5*) and two AND arcs(*AND arc1* and *AND arc2*) are generated at Stage2. Next, it is tested

whether the separation corresponding to each of the newly generated AND arcs can be done without any geometric interference. We perform this test automatically using CAD data by a method we proposed before[10]. Next, this algorithm selects a node and calculates some elementary cut sets again to generate its child nodes. In Fig. 2, node *N3* is selected and its child nodes(*N6, N7, N8*, and *N9*) are generated at Stage3. These operations are repeated until a disassembly sequence is obtained. At selecting a node, this algorithm takes account of the costs for separating the product and subassemblies. The costs are calculated through the tests mentioned above. By considering the costs, this algorithm can generate a good disassembly sequence whose cost is the smallest in the reduced search space.

This algorithm and AO* algorithm[15] are identical in the way of handling the costs and selecting a node in searching. However, this algorithm is different from AO* algorithm in that this algorithm searches the reduced search space(i.e., this algorithm generates not all but some child nodes and AND arcs), as shown in Fig. 3(a). Therefore, this algorithm doesn't always find an optimal solution because the reduced search space may not include the optimal solution. However, this algorithm can find the best solution in the reduced search space. Figure 3(b) shows a case that the reduced search space doesn't include any feasible region(i.e., a case that all the separations corresponding to the generated elementary cut sets are infeasible), however this is very rare. In such a case, this algorithm expands the search space by adding some child nodes and AND arcs to the AND/OR graph. Figure 3(c) shows a case that the reduced search space includes small feasible region(i.e., a case that almost all the separations corresponding to the generated elementary cut sets are infeasible). In such a case, we may not obtain a good solution. To avoid such a case(i.e., to let the reduced search space include large feasible region), our method calculates some good elementary cut sets that have strong possibilities of separating the product or subassembly by using the heuristics and probabilistic tree transformation. The method for utilizing the heuristics and probabilistic tree transformation is described in the following sections.

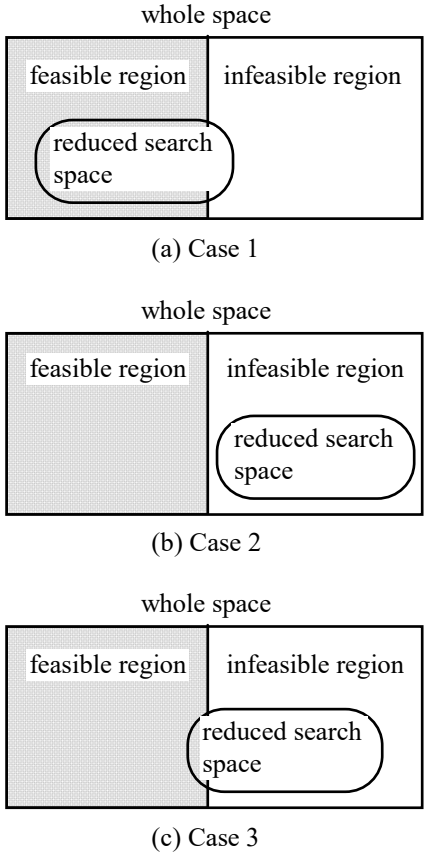


Figure 3. Relationship among the reduced search space and feasible/ifeasible regions

4.2 Generation of Cut Sets using Heuristics and Probabilistic Tree Transformation

Figure 4 shows the procedure for utilizing the heuristics. Each of the operations in the procedure is described as follows.

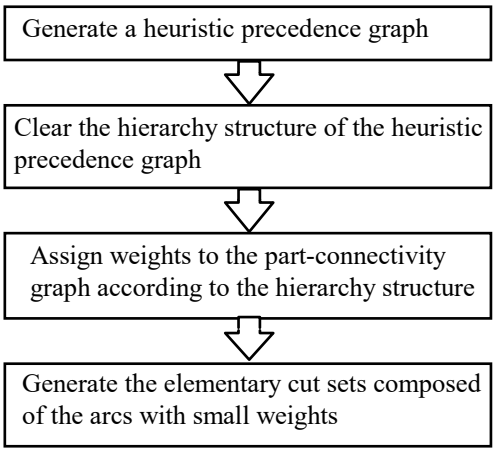


Figure 4. Procedure for generating good elementary cut sets

4.2.1 Heuristic Precedence Graph

First, we generate a heuristic precedence graph by using the heuristics. Each node of this graph expresses a connective relation between two parts, and each arc expresses a heuristic precedence relation between two connective relations. This heuristic precedence relation means that the connective relation represented by its terminal node very probably emerges earlier than that represented by its starting node when the product is assembled(conversely, the connective relation represented by the starting node is very probably released earlier than that represented by the terminal node when the product is disassembled).

We can generate such a heuristic precedence graph by applying heuristic rules to the CAD data of a product. Figure 5 shows an example of applying one of the heuristic rules. Such heuristic rules can be induced automatically by a machine-learning technique[11].

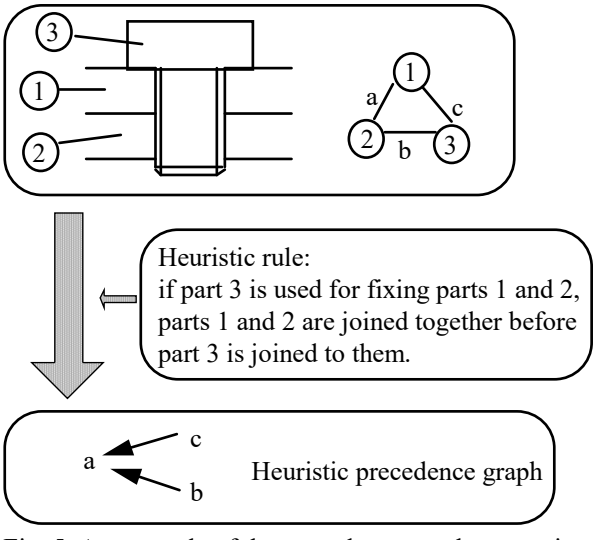


Figure 5. An example of the precedence graph generation by applying a heurustic rule

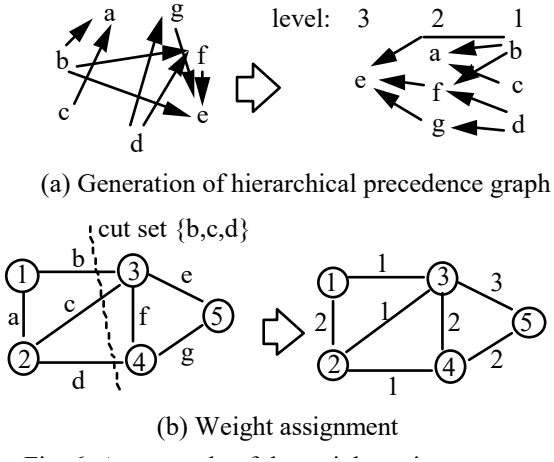


Figure 6. An example of weight assignment using predence graph

4.2.2 Weight Assignment

We combine the heuristic precedence graphs, each of which is made by applying one of the heuristic rules, and change it to a hierarchical graph, as shown in Fig. 6 (a). This change is executed by using a technique in Interpretive Structural Modeling[16]. By using the hierarchical graph, we assign the weights to the connective relations in the part-connectivity graph according to the levels in which the connective relations are included, as shown in Fig. 6. For example, connective relation *e* is in level 3 of the hierarchical graph, therefore 3 is assigned to it in the part-connectivity graph. The larger weight the connective relation has, the earlier it very probably emerges in the assembly stage(conversely, the smaller weight the connective relation has, the earlier it is very probably released in the disassembly stage).

By using the weights, we can evaluate the cut sets from the viewpoint of the possibility of separating the product or subassembly. Namely, we can regard an elementary cut set composed of the arcs with small weights as one that has strong possibility of separating the product or subassembly. In our method, each elementary cut set *Ci* is evaluated by:

$$Ei=max\{w_1,w_2,...,w_j,...,w_n\} \quad (2)$$

where w_j is the weight of arc *j* included in cut set *Ci*. The smaller *Ei* is, the better the cut set is. For example, *Ei* of elementary cut set {*b, c, d*} shown in Fig. 6 is 1. As this value is the smallest, this cut set can be considered as the best one.

4.2.3 Probabilistic Generation of Cut Sets

The simplest method for generating the good elementary cut sets(i.e., the elementary cut sets composed of the arcs with small weights) is to generate all the cut sets and select the good ones out of them. However it takes very long time to generate all the cut sets especially for the products composed of many parts. Therefore our method generates some good elementary cut sets efficiently without generating all the cut sets.

4.2.3.1 Tree and Fundamental Cut Sets

To generate the elementary cut sets, first, we generate a tree *T* in the part-connectivity graph. A tree is defined by a set of the arcs that connect all the nodes but don't include any circuits. In Fig. 7, the set of the arcs drawn in bold lines, {*b, c, e, g*}, is an example of the tree. Using tree *T*, we can generate fundamental cut sets, each of which contains one arc included in *T*. In the example shown in Fig. 7, *C1, C2, C3,* and *C4* are the fundamental cut sets for tree *T*. Such fundamental cut sets are also necessarily elementary cut sets. Therefore, if they are good ones(i.e., the elementary cut sets composed of the arcs with small weights), they can be used as the candidates for partitioning the part-connectivity graph in searching the AND/OR graph.

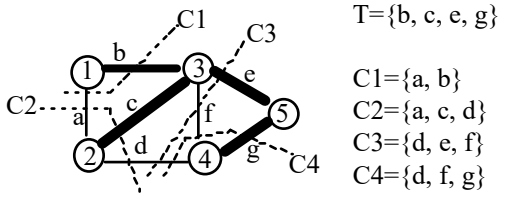


Figure 7. An example of the tree and fundamental cut sets

4.2.3.2 Probabilistic Tree Transformation

By using tree *T*, we can generate some elementary cut sets whose number is *N(V)-1*, where *N(V)* is the number of the nodes included in the part-connectivity graph *G(V, A)*. We can generate the other elementary cut sets by performing the tree transformation, in which a member of tree is replaced with another arc. The procedure is as follows:

Step 1: Select a cut set *Ci* with small *Ei* out of the fundamental cut sets.
Step 2: If the arc with the maximum weight in *Ci* is a member of cotree *CT*(=*A-T*), call the arc *ARC* and do the following procedure. Otherwise(i.e., if the arc is a member of tree *T*), go back to Step 1.
Step 3: Add *ARC* to *T*. By adding this, a circuit becomes included in *T*. Select and delete the arc with the small weight in the circuit and add it to the cotree. This arc may become a new member of *Ci*. As a result of this operation, a new tree is generated.
Step 4: Generate new fundamental cut sets corresponding to the new tree.

By executing the procedure, we can improve the good cut set further. Figure 8 shows an example of applying the procedure to the case of the weight assignment shown in Fig. 6(b). In this example, first, cut set *C1* whose *E1* is *2* is selected in Step 1. Next, the maximum-weight arc *a* in *C1* is selected and added to the tree in Step 2 and 3. As a result, circuit{*a, b, c*} is generated, as shown in Fig. 8(b). Then arc *c* whose weight is small is deleted from the circuit and added to the cotree. Consequently the new tree {*a, b, e, g*}and the new

fundamental cut set *C5*, which is better than *C1* selected in Step 1, are generated, as shown in Fig. 8(c).

However, if we select the cut set with the minimum *Ei* in Step 1 and the arc with the minimum weight in Step 3, we can generate only a limited number of cut sets and therefore we may come to rest in local minimum of the space for searching the cut sets. To avoid it, we propose the probabilistic tree transformation that selects a cut set in Step 1 and an arc in Step 3 probabilistically. Namely, in Step 1 we select cut set *Ci* with the following probability *PCi*:

$$PCi = \frac{1/Ei}{\sum(1/Ei)} \tag{3}$$

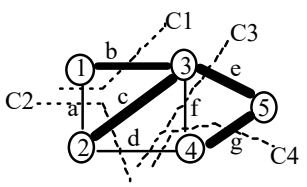
The smaller *Ei* is, the larger *PCi* is(i.e., the cut set with small *Ei* is very probably selected). In the same way, we select arc *j* in Step 3 with the following probability *PAj*:

$$PAj = \frac{1/wj}{\sum(1/wj)} \tag{4}$$

Roulette Wheel Selection Algorithm[17] is used for the selections taking account of the probabilities.

The above-mentioned procedure is repeated and all the generated cut sets are memorized. Then, some good cut sets are selected out of them to partition the part-connectivity graph.

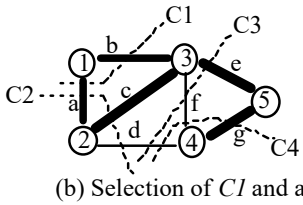
As we mentioned above, the tree transformation, in which the cut set with the minimum *Ei* in Step 1 and the arc with the minimum weight in Step 3 are selected, generates a limited number of cut sets, whereas the probabilistic tree transformation can generate a variety of cut sets, and this brings about a better assembly sequence search.



T={b, c, e, g}

C1={a, b} E1=2
C2={a, c, d} E2=2
C3={d, e, f} E3=3
C4={d, f, g} E4=2

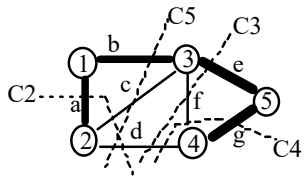
(a) Initial tree and fundamental cut sets



T={a, b, c, e, g}

C1={a, b} E1=2
C2={a, c, d} E2=2
C3={d, e, f} E3=3
C4={d, f, g} E4=2

(b) Selection of *C1* and addition of *a* to *T*



T={a, b, e, g}

C2={a, c, d} E2=2
C3={d, e, f} E3=3
C4={d, f, g} E4=2
C5={b, c, d} E5=1

(c) Addition of *c* to the cotree and the consequent generation of the new tree and fundamental cut sets

Figure 8. An example of the tree transformation

5. EXPERIMENT AND ANALYSIS

To demonstrate the effectiveness of our method, we developed a software tool which is integrated with a CAD system. By using the tool, we present a calculation example of a gear pump. Figure 9 shows the 3d-model of the gear-pump, which we built by using the CAD system. Figure 10 shows its graph representation and Table 1 shows its parts list. By using the model, this software tool generated an assembly sequence, with using four good cut sets at a time to partition the part-connectivity graph. Figure 11 shows the generated assembly sequence. In this figure, SAi(i=0,1,...,14) is the product or subassembly. Table 2 shows the parts included in the product and subassemblies. This assembly sequence is an optimal one, however our approach doesn't always find optimal ones because of the reduction of the search space. We could not compare our computational time with those of the other approaches because it is a very hard work to implement the other approaches on our computer. However, as the number of the tests for separability is about 10 percent of that in the case of generating all the elementary cut sets, the computational time of our method may be reduced to about 10 percent.

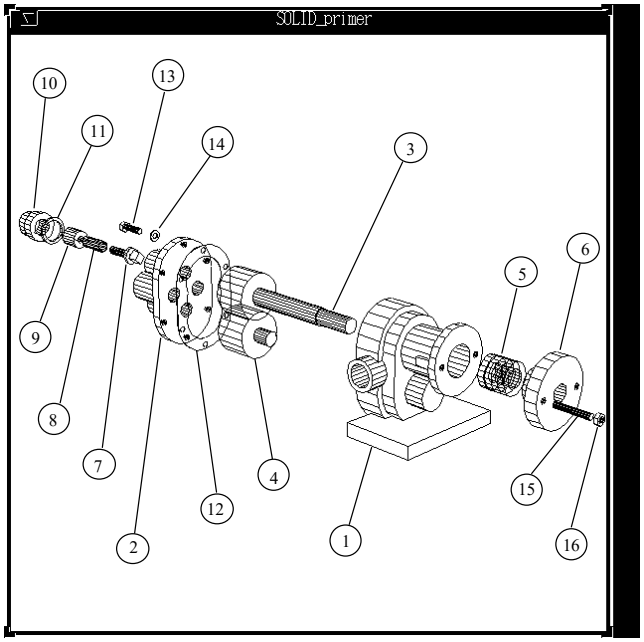


Figure 9. 3d-model of the gear pump

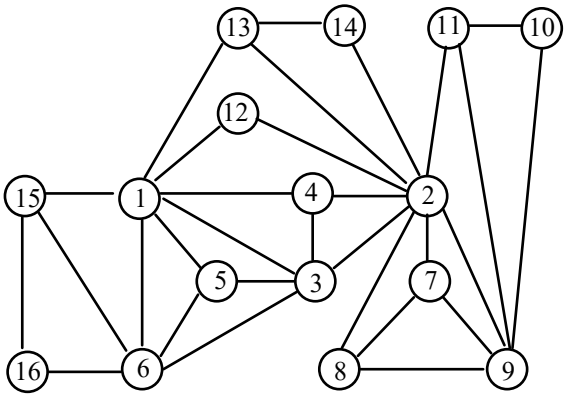


Figure 10. Part-connectivity graph of the gear pump

Table 1 Parts list of the gear pump

1	body	9	valve guide
2	cover	10	valve guide cover
3	shaft with gear	11	packing
4	shaft with gear	12	sheet packing
5	packing	13	tap bolt
6	packing gland	14	washer
7	valve	15	stud bolt
8	coil	16	nut

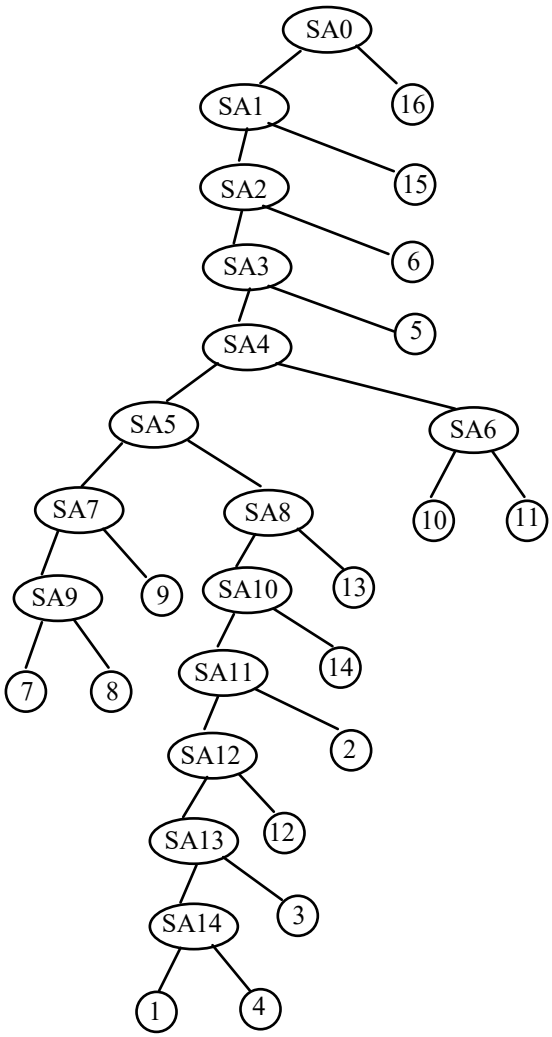


Fig. 11 Disassembly/assembly sequence generated by the software tool

Figure 11. Dissambly/assembly sequence generated by the software tool

Table 2 The parts included in product and subassemblies

product or subassembly	parts
SA0	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
SA1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
SA2	1,2,3,4,5,6,7,8,9,10,11,12,13,14
SA3	1,2,3,4,5,7,8,9,10,11,12,13,14
SA4	1,2,3,4,7,8,9,10,11,12,13,14
SA5	1,2,3,4,7,8,9,12,13,14
SA6	10,11
SA7	7,8,9
SA8	1,2,3,4,12,13,14
SA9	7,8
SA10	1,2,3,4,12,14
SA11	1,2,3,4,12
SA12	1,3,4,12
SA13	1,3,4
SA14	1,4

6. CONCLUSION

We proposed the efficient method for generating assembly sequences. This method extracts some parts and/or subassemblies whose possibilities of being removed from a product are strong, and tests whether they can be removed without any geometric interference. By performing these operations repeatedly, the method generates a disassembly sequence of the product, and obtains an assembly sequence by reversing it. The characteristics of this method are to reduce the search space and to avoid to come to rest in local optimum by using the heuristics and probabilistic tree transformation. We developed the software tool and carried out the experiment using the tool. As a result of the experiment, the tool found the optimal assembly sequence, and the number of the tests for separability is about 10 percent of that of the whole tests. This result shows that the proposed method could find a good assembly sequence and bring about a large reduction in the computation time for assembly sequence planning.

REFERENCES

[1] A. J. D. Lambert, **Disassembly sequencing: A survey**, International Journal of Production Research, Vol. 41, Issue 16, pp.3721-3759, 2003.

[2] Michał Taraska, Remigiusz Iwańkowicz, Tomasz Urbański, Tadeusz Graczyk, **Review of Assembly Sequence Planning Methods in Terms of Their Applicability in Shipbuilding Processes**, Polish Maritime Research, Vol. 25, Issue s1, pp. 124–133, 2018.

[3] D. F. Baldwin, T. E. Abell, M. M. Lui, T. L. DeFazio and D. E. Whitney, **An Integrated Computer Aid for Generating and Evaluating Assembly Sequences for Mechanical Products**, IEEE Transactions of Robotics and Automation, Vol. 7, No. 1, pp. 78-94, 1991.

[4] A. Bourjault, **Contribution a une approche methodologique de l'assemblage automatise: Elaboration automatique des sequences operatories**, Ph. D. dissertation, Universite de Franche-Comte, 1984.
[5] S. Lee, and C.Yi, **Subassembly Stability and Reorientation,** Proceedings of IEEE Robotics and Automation, pp. 521-526, 1993.
[6] S. Abe, T. Murayama, F. Oba, and A. Narutaki, **Stability Check and Reorientation of Subassemblies in Assembly Planning**, Proceedings of The 1999 IEEE System, Man and Cybernetics Conference (SMC'99) Vol.1, pp.II-486-II-491, 1999.
[7] Yong Wang, Jihong Liu, **Subassembly identification for assembly sequence planning**, The International Journal of Advanced Manufacturing Technology, Volume 68, Issue 1–4, pp 781–793, 2013.
[8] Moez Trigui, Imen Belhadj, Abdelmajid Benamara, **Disassembly Plan Approach based on Subassembly Concept**, The International Journal of Advanced Manufacturing Technology, Volume 90, Issue 1–4, pp 219–231, 2017.
[9] T. Murayama and F. Oba, **An Efficient Method for Generating Assembly Sequences in Product Design Stages**, Proceedings of IECON'93, pp. 564-569, 1993.
[10] T. Murayama and F. Oba, **Disassembly/Assembly Path Search Reusing Solutions**, Proceedings of IEEE 6th International Conference on Emerging Technologies and Factory Automation, pp. 183-188, 1997.
[11] T. Murayama, F. Oba, and B. Takemura, **Assembly Sequence Planning using Inductive Learning Techniques**, Journal of Robotics and Mechatronics, Vol.11, No.4, pp.315-320, 1999.
[12] M. Santochi, et al., **STC 'A' Cooperative Work on Assembly-Planning Software Systems**, Annals of the CIRP, Vol. 44/2, pp. 651-658, 1997.
[13] T. Murayama, F. Oba, and S. Abe: **Assembly partitioning by genetic algorithm for generating assembly sequences efficiently**, Advancement of intelligent production, pp. 695-700, 1994.
[14] Adrian Bondy and U.S.R. Murty, **Graph Theory (Graduate Texts in Mathematics),** Springer, 2011.
[15] Rich, E., **Artificial Intelligence**, McGraw-Hill, pp. 87-94, 1983.
[16] J. N. Warfield, **Toward Interpretation of Complex Structural Model**, IEEE Transactions on Systems, Man & Cybernetics, Vol.4, No.5, pp.405-417, 1974.
[17] Melvin Ballera, **Roulette Wheel Selection Algorithm and Reinforcement Learning**, LAP LAMBERT Academic Publishing, 2017.