한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003 년 5 월 16 일-17 일 한동대학교(포항)

분해 일정계획 문제에 대한 정수계획 모형

Integer Programming Models for Disassembly Scheduling

Dong-Ho Lee[1], **Hwa-Joong Kim**[2], and **Paul Xirouchakis**[2]

[1]Department of Industrial Engineering
Hanyang University
Sungdong-gu, Seoul 133-791
KOREA
e-mails:leman@hanyang.ac.kr

[2]Institute of Production and Robotics (STI-IPR-LICP)
Swiss Federal Institute of Technology (EPFL)
Lausanne, CH-1015
SWITZERLAND
e-mails:{hwa-joong.kim, paul.xirouchakis}@epfl.ch

Abstract

Disassembly scheduling is the problem of determining the ordering and disassembly schedules of used or end-of-life products while satisfying the demand of their parts or components over a certain planning horizon. To represent and optimally solve the basic case, i.e., single product type without parts commonality, an integer programming model is suggested for the objective of minimizing the sum of purchase, setup, inventory holding, and disassembly operation costs. Then, the basic model is extended to consider the cases of single and multiple product types with parts commonality. Computational experiments done on the examples obtained from the literature show that the integer programming approach suggested in this paper works well.

1. Introduction

Due to increasing legislation pressures to collect and recover used or end-of-life products in an environmentally conscious way, manufacturing firms have been paying considerable attention to material and product recovery processes. The material recovery, i.e., recycling, aims to recover the material content of used products, while the product recovery, i.e., remanufacturing, preserves product or its parts/components by performing the required disassembly, sorting, refurbishing and assembly operations (Gungor and Gupta 1999). In both types of recovery processes, disassembly is an important step since used or end-of-life products are disassembled before recycled or remanufactured.

This paper focuses on disassembly scheduling, which is the problem of determining the ordering and disassembly schedules of used or end-of-life products while satisfying the demand of their parts/components over a certain planning horizon. Disassembly scheduling is one of the important production planning problems in disassembly systems. In other words, from its solution, we can determine which products, how many, and when to disassemble products and their subassemblies.

There are several previous research articles on the disassembly scheduling problem. Gupta and Taleb (1994) consider the basic case, i.e., single product type without parts commonality, and suggest an MRP-like algorithm. In their algorithm, the demand of items at one level of the product structure is translated into an equivalent demand of the items at the next level, which is repeated from the level of parts/components to the product level. Later, Taleb *et al.* (1997) extend the basic case by considering the parts commonality, i.e., single product type with parts commonality, and suggest another MRP-like algorithm for the objective of minimizing the number of products to be disassembled. Here, the extended problem is more complicated than the original one since the parts commonality may result in two or more alternative procurement sources for each common item and hence create additional

interdependencies between parts/components. For this extended problem, Neuendorf *et al.* (2001) suggest an algorithm based on the Petri-net, and show using the example of Taleb *et al.* (1997) that their algorithm gives a better solution than the existing MRP-like algorithm. Taleb and Gupta (1997) consider the case of multiple product types with parts commonality, and suggest a two-phase heuristic algorithm for two independent objectives of minimizing the number of products to be disassembled and minimizing the disassembly costs. Recently, Lee *et al.* (2002) suggest an integer programming model for the problem with the resource capacity constraint.

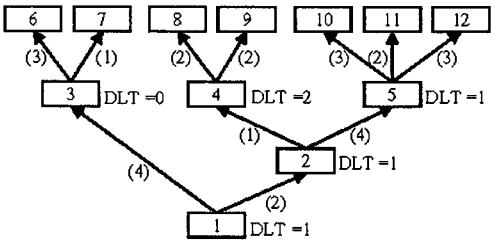
Three cases are considered in this paper. The most basic one is *single product type without parts commonality*, which is the same as that of Gupta and Taleb (1994) except for the objective function. Unlike the simple objective of minimizing the number of products to be disassembled, we consider various costs, such as purchase, setup, inventory holding, and disassembly operation costs, which are commonly used in ordinary production planning problems. The next one is *single product type with parts commonality*, which is the same as that of Taleb *et al.* (1997) and Neuendorf *et al.* (2001) except for the objective function. In general, the parts commonality is a prevalent characteristic of products since it gives a lower production cost and a more flexible production system, and hence it should not be overlooked in disassembly scheduling (Taleb *et al.* 1997). Finally, we consider *the case of multiple product types with parts commonality*, which is the same as that of Taleb and Gupta (1997) except for the objective function.

To represent and optimally solve the basic case of the problem, an integer programming model is suggested for the cost-based objective function. The basic idea is that the disassembly scheduling problem considered here can be regarded as a reverse form of the multi-level lot-sizing problem. Note that multi-level lot-sizing is the problem of determining the number of parts/components to be assembled in order to satisfy the demand of the final product. See Bahl *et al.* (1987) for a review on various multi-level lot-sizing problems. Also, based on the model for the basic case, two more integer programming models are suggested for the cases with parts commonality, i.e., single and multiple product types with parts commonality. As stated earlier, the objective is to minimize the sum of purchase, setup, inventory holding, and disassembly operation costs. Note that each of the three integer programs can be easily modified to consider the existing objective of minimizing the number of products to be disassembled or minimizing product disassembly costs. To show the performance of the integer programming approach suggested in this paper, computational experiments are done on the examples obtained from the literature and the results are reported.

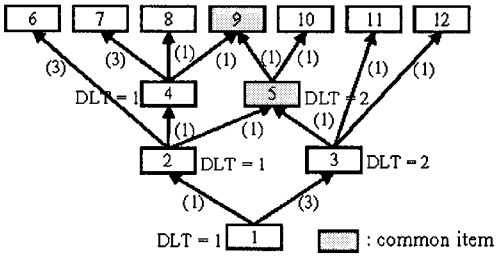
2. Problem Description

This section begins with the definition of the disassembly product structure. In this structure, the root item represents the product itself to be disassembled and each leaf item represents any item that cannot be further disassembled. Also, a child item represents any item that has a parent at the next higher level and a parent item is any item that has at least two child items at the next lower level. Three examples of the disassembly product structure are shown in Fig. 1. Fig. 1(a) shows an example of single product type without parts commonality, to be denoted as the GT example in this paper, which is obtained from Gupta and Taleb (1994). In this figure, item 1 represents the root item, and items 6, 7, 8, 9, 10, 11, and 12 represent the leaf items. The number in parenthesis represents the yield of the item when its parent is disassembled, e.g., item 5 is disassembled into in three units of item 10, two units of item 11, and three units of item 12. Also, the disassembly lead-time (DLT) is the time required to disassemble a certain parent item. Note that each item has at most one parent in the case without parts commonality. Fig. 1(b) shows an example of single product type with parts commonality, to be denoted as the TGB example in this paper, which is obtained from Taleb *et al.* (1997). In this figure, the shaded boxes represent common items. For example, item 5 is a common item since it has more than one parent, i.e., item 5 can be obtained from either item 2 or 3. Finally, an example of multiple product types with parts commonality, to be denoted as the TG example in this paper, is shown in Fig. 1(c), which is obtained from Taleb and Gupta (1997). Note that the TG example has three root items.

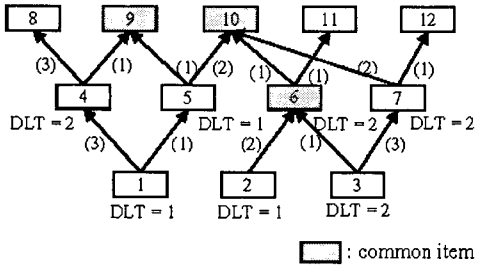
Now, the problem considered in this paper can be described as follows: *for a given disassembly product structure, we want to determine the ordering schedule for the root item and the disassembly schedules for all parent items while satisfying the demand of leaf items over a certain planning horizon with the objective of minimizing the sum of purchase, setup, inventory holding, and disassembly operation costs*. Here, the disassembly product structure can be obtained from the disassembly process plan that specifies all parts/subassemblies of a given product and necessary disassembly operations. For reviews on disassembly process planning, see O'Shea *et al.* (1998), Lee *et al.* (2001), and Santochi *et al.* (2002).



(a) Single product type without parts commonality



(b) Single product type with parts commonality



(c) Multiple product types with parts commonality

Figure 1. Disassembly product structure: examples

We consider time-varying purchase costs depending on the market situation, which means that the purchase costs may vary according to different planning periods. The setup cost, which is specific for each parent item, is the total sum of costs required to ready to disassembly a parent item. It is assumed that a setup cost for an item in a period occurs if at least one operation for that item should be performed in that period. Also, inventory holding costs occur when items are held in order to satisfy future demand. Finally, the disassembly operation cost, apart from the setup cost mentioned above, is proportional to the labor time or machine processing time required for the corresponding disassembly operation.

It is assumed that there is no shortage of the root items, i.e., the used or end-of-life products can be supplied whenever they are ordered. Also, as assumed in previously published research articles, we consider scheduled receipts from external sources, i.e., the items that are expected to arrive from outside sources and not from disassembly. Note that scheduled receipts from external sources are zero if all items are derived strictly from disassembly. Other assumptions made in this problem can be summarized as follows: (a) the planning horizon is divided into discrete planning periods; (b) demands for leaf items are given and deterministic; (c) backlogging is not allowed and hence demand should be satisfied on time; (d) disassembly processes are perfect, and hence no defective parts and components are obtained from disassembly; (e) disassembly lead times are given and deterministic; and (f) inventory holding costs are computed based on the end-of-period inventory.

3. The Integer Programming Approach

This section presents the integer programming models that represent and optimally solve the three cases of the disassembly scheduling problem. An integer program is suggested for the basic case of single product type without parts commonality, and then it is extended to consider the more general cases of single and multiple product types with parts commonality.

3.1 The case of single product type without parts commonality

To formulate this basic case, all items in the disassembly structure are numbered in the topological order from bottom to top and from left to right, starting with number one for the root item. That is, without loss of generality, all items are numbered by integers 1, 2,..., i_l, $i_l + 1$,..., I, where $i_l + 1$ is the index for the first leaf item and hence the indices that are larger than i_l represent leaf items.

The notations used in the formulation are summarized below.

Parameters

- c_t purchase cost of the product in period t
- s_i set up cost of item i
- h_i inventory holding cost of item i
- p_i disassembly operation cost of item i
- D_{it} demand of item i in period t
- a_{ij} number of units (yield) of item j obtained from disassembling one unit of item i
- r_{it} scheduled receipt of item i in period t
- $\varphi(i)$ parent of item i

l_i disassembly lead time (DLT) of item i
I_{i0} initial inventory of item i

Decision variables

Z_t purchase quantity of the root item in period t
X_{it} disassembly quantity of item i in period t
Y_{it} = 1 if there is a setup for item i in period t, and 0 otherwise
I_{it} inventory level of item i at the end of period t

Now, the integer program [P1] for the case of single product type without parts commonality is given below.

[P1]

$$\text{Minimize } \sum_{t=1}^{T} c_t \cdot Z_t + \sum_{i=1}^{q} \sum_{t=1}^{T} s_i \cdot Y_{it} + \sum_{i=1}^{I} \sum_{t=1}^{T} h_i \cdot I_{it} + \sum_{i=1}^{q} \sum_{t=1}^{T} p_i \cdot X_{it}$$

subject to

$$I_{1t} = I_{1,t-1} + r_{1t} + Z_t - X_{1t} \quad \text{for } t = 1,\dots T \qquad (1)$$

$$I_{it} = I_{i,t-1} + r_{it} + a_{(i),i} \cdot X_{(i),t-l_{(i)}} - X_{it}$$
$$\text{for } i = 2,\dots i_l \text{ and } t = 1,\dots T \qquad (2)$$

$$I_{it} = I_{i,t-1} + r_{it} + a_{(i),i} \cdot X_{(i),t-l_{(i)}} - D_{it}$$
$$\text{for } i = i_l + 1,\dots, I \text{ and } t = 1,\dots T \qquad (3)$$

$$X_{it} \le M \cdot Y_{it}$$
$$\text{for } i = 1,\dots i_l \text{ and } t = 1,\dots T \qquad (4)$$

$$Z_t \ge 0 \text{ and integer} \quad \text{for } t = 1,\dots T \qquad (5)$$

$$X_{it} \ge 0 \text{ and integer}$$
$$\text{for } i = 1,\dots i_l \text{ and } t = 1,\dots T \qquad (6)$$

$$I_{it} \ge 0 \text{ and integer}$$
$$\text{for } i = 1,\dots, I \text{ and } t = 1,\dots T \qquad (7)$$

$$Y_{it} \in \{0,1\} \text{ for } i = 1,\dots i_l \text{ and } t = 1,\dots T \qquad (8)$$

The objective function denotes the sum of purchase, setup, inventory holding, and disassembly operation costs. Note that the objective of the GT example obtained from Gupta and Taleb (1994) is to minimize the number of products to be disassembled and can be represented as,

$$\text{Minimize } \sum_{t=1}^{T} Z_t \qquad (9)$$

Constraint (1) represents the inventory balance for the root item (product). That is, at the end of each planning period, we have the inventory that we had a period before, increased by the amount of scheduled receipt and purchased quantity, and decreased by the disassembly quantity of the product. Constraint (2) represents the inventory balance for the parent items that should be disassembled further. This is the same as (1) except that for each parent item, the disassembly quantity of its parent, multiplied by its yield from its parent, is used instead of the purchase quantity. Also, the inventory balance for each leaf item is represented by constraint (3), which is different from (2) in that the demand requirement is used instead of the disassembly quantity. Constraint (4), where M is an arbitrary large number, guarantees that a setup cost in each period is incurred whenever at least one disassembly operation is performed in that period. Finally, the other constraints (5), (6), (7), and (8) represent conditions on the decision variables. In particular, constraint (7) ensures that backlogging is not allowed. Note that constraints (4) and (8) are not needed when the GT example is formulated because its objective of minimizing the number of products to be disassembled does not include setup.

3.2 The case of single product type with parts commonality

This case can be formulated by modifying the inventory balance constraints (2) and (3) of [P1]. To do this, we first define an additional notation $\Phi(i)$ to represent the parts commonality. Unlike $\varphi(i)$ in [P1], $\Phi(i)$ denotes the set of parents of item i. Recall that there may be two or more parents for each item in the case with parts commonality.

The integer program [P2] for the case of single product type with parts commonality is given below.

[P2]

$$\text{Minimize } \sum_{t=1}^{T} c_t \cdot Z_t + \sum_{i=1}^{q} \sum_{t=1}^{T} s_i \cdot Y_{it} + \sum_{i=1}^{I} \sum_{t=1}^{T} h_i \cdot I_{it} + \sum_{i=1}^{q} \sum_{t=1}^{T} p_i \cdot X_{it}$$

subject to

$$I_{it} = I_{i,t-1} + r_{it} + \sum_{k \in \Phi(i)} a_{ki} \cdot X_{k,t-l_k} - X_{it}$$
$$\text{for } i = 2,\dots i_l \text{ and } t = 1,\dots T \qquad (2')$$

$$I_{it} = I_{i,t-1} + r_{it} + \sum_{k \in \Phi(i)} a_{ki} \cdot X_{k,t-l_k} - D_{it}$$
$$\text{for } i = i_l + 1,\dots I \text{ and } t = 1,\dots T \qquad (3')$$
$$\text{and (1), (4), (5), (6), (7) and (8)}$$

The objective function is the same as that of [P1]. Note that the objective of the TGB example obtained from Taleb *et al.*(1997), which is to minimize the number of products to be disassembled, can be represented with (9). Also, like the GT example, constraints (4) and (8) related to setup are not needed in the formulation of the TGB example with the original objective. The inventory balance constraint for the root item is the same as that of [P1] since both cases consider a single product type. On the other hand, the inventory balance constraints for the parent items (except for the root item) should be modified to consider the parts commonality. The modification is done to represent the quantity resulting from disassembling the set of parents for each child item. That is, unlike considering one

parent for each child item in the constraints (2) and (3) of [P1], the modified constraints (2′) and (3′) of [P2] represent the set of parents for each child item and hence describe the quantity obtained from summing over those resulting from disassembling the set of parents related with that child item.

3.3 The case of multiple product types with parts commonality

This case can be formulated by extending [P2] in such a way that the inventory balance constraint for a single root item is expressed for each of the multiple root items. In this case, all items are numbered by integers 1, 2,..., i_r, $i_r + 1$,..., i_b $i_l + 1$,..., I, where the additional notation i_r denotes the number of root items and hence $i_r + 1$, $i_r + 2$,..., i_l represent the indices for parent items except for the multiple root items.

The integer program [P3] for the case of multiple product types is given below.

[P3]

$$\text{Minimize } \sum_{i=1}^{i_r}\sum_{t=1}^{T} c_{it}\cdot Z_{it} + \sum_{i=1}^{i_l}\sum_{t=1}^{T} s_i\cdot Y_{it} + \sum_{i=1}^{I}\sum_{t=1}^{T} h_i\cdot I_{it} + \sum_{i=1}^{i_l}\sum_{t=1}^{T} p_i\cdot X_{it}$$

subject to

$$I_{it} = I_{i,t-1} + r_{it} + Z_{it} - X_{it}$$
$$\text{for } i = 1, \dots i_r \text{ and } t = 1, \dots T \qquad (1')$$

$$I_{it} = I_{i,t-1} + r_{it} + \sum_{k\in\tilde{P}(i)} a_{ki}\cdot X_{k,t-l_k} - X_{it}$$
$$\text{for } i = I_r + 1, \dots I_l \text{ and } t = 1, \dots T \qquad (2'')$$

and (3′), (4), (5), (6), (7), and (8)

The objective function is the same as that of [P2] except that the purchase costs is summed over the multiple root items. Note that an additional subscript i is used for purchase costs and purchase quantities, i.e., c_{it} and Z_{it} for $i = 1, 2,\dots, i_r$ and $t = 1, 2,\dots, T$. The modified constraints (1′) and (2″) represent the inventory balances of the multiple root items, $i = 1, 2,\dots, i_r$, and the parent items, $i = i_r + 1, 2,\dots, i_b$, and the others are the same as those of [P2]. Note that the TG example considers two independent objective functions, i.e., minimizing the number of products to be disassembled and minimizing the sum of product disassembly costs, and they can be represented with (9′) and (10), respectively.

$$\text{Minimize } \sum_{i=1}^{i_r}\sum_{t=1}^{T} Z_{it} \text{ and} \qquad (9')$$

$$\text{Minimize } \sum_{i=1}^{i_r} g_i\cdot\left(\sum_{t=1}^{T} Z_{it}\right), \qquad (10)$$

where g_i is the disassembly cost of product i. Like the GT and TGB examples, constraints (4) and (8) related to setup are not required when formulating the TG example with the original objective.

4. Computational Experiments

This section presents the test results on the three integer programs suggested in this paper. In these tests, CPLEX 6.5, a commercial software package, was used to solve the integer programs. The TGB and TG examples with the original objectives are solved, and their solutions are compared with those of Taleb *et al.* (1997) and Taleb and Gupta (1997), respectively. Note that no comparison was done for the GT example (the case of single product type without parts commonality) since the MRP-like algorithm of Gupta and Taleb (1994) already gives the optimal solution. In the tests, the program to generate integer programs was coded in C and the tests were done on a personal computer with a Pentium processor operating at 800 MHz clock speed.

Test results for the TGB example are given in Table 1, which summarizes the ordering schedules obtained from Taleb *et al.* (1997) (original and corrected), Neuendorf *et al.* (2001), and the integer programming approach, respectively. Note that Neuendorf *et al.* (2001) report the round-off errors in the original solution of Taleb *et al.* (1997) and suggest a corrected solution. The optimal number of products to be disassembled, obtained from the integer programming approach, is 551, which is the same as that of the Petri-net based algorithm suggested by Neuendorf *et al.* (2001). It can be seen from the table that the optimal solution obtained from the integer programming approach is different from that of Neuendorf *et al.* (2001). This implies that the TGB example has multiple optimal solutions, and hence it can be argued that the cost-based objective suggested in this paper is needed to evaluate different solutions with the same number of products to be disassembled. Finally, the CPU second required to obtain the optimal solution was 0.02.

Table 1. Comparison of solutions: TGB example

Approaches	Period										Objective values
	1	2	3	4	5	6	7	8	9	10	
TGB[1]		77	49	416							542
TGB[2]		77	49	416	17						559
NLKX[3]		77	15	450			9				**551**
IP[4]	154			388			9				**551**

[1] original solution (with round-off errors) of Taleb *et al.* (1997)
[2] corrected solution of Taleb *et al.* (1997) (reported in Neuendorf *et al.* (2001))
[3] solution of Neuendorf *et al.* (2001)
[4] solution of the integer programming approach with the objective of minimizing the number of products to be disassembled (CPU seconds: 0.02)

Table 2 summarizes the test results for the TG example, i.e., the case of multiple product types with parts commonality. In this test, as in Taleb and Gupta (1997), the disassembly costs were set to 20, 6, and 21 for products 1, 2, and 3, respectively. The integer programming approach outperformed the two-phase heuristic of Taleb and Gupta (1997). In fact, the amounts of improvement were 33.3% and 3.75% for the cases with minimizing the number of products to be disassembled and minimizing the disassembly cost of products, respectively. Finally, CPLEX 6.5 required 0.07 seconds to obtain the optimal solution for each of the two cases.

Table 2. Comparison of solutions: TG example

(a) Taleb and Gupta (1997)

Products	Period										Total
	1	2	3	4	5	6	7	8	9	10	
1			1								1
2			1			2					3
3	1	1									2

Number of product disassembled = 6
Total disassembly cost = 20·1 + 6·3 + 21·2 = 80

(b) Integer programming approach

(Minimizing the number of products disassembled)

Products	Period										Total
	1	2	3	4	5	6	7	8	9	10	
1		1									**1**
2											**0**
3	3										**3**

Number of product disassembled = 4
CPU seconds: 0.07

(Minimizing disassembly cost of products)

Products	Period										Total
	1	2	3	4	5	6	7	8	9	10	
1	1										**1**
2	4	1	1								**6**
3	1										**1**

Total disassembly cost = 20·1 + 6·6 + 21·1 = 77
CPU seconds: 0.07

5. Concluding Remarks

This paper considered the problem of determining the ordering and disassembly schedules of used products while satisfying the demand of their parts/components over a given planning horizon. The objective is to minimize the sum of purchase, setup, inventory holding, and disassembly operation costs, which generalizes the previous simple objective of minimizing the number of products to be disassembled. Three integer programs were suggested to represent and optimally solve three cases of the problem, i.e., single product type without and with parts commonality and multiple product types with parts commonality. Computational experiments were done on the examples obtained from the literature (for the original objectives) and the test results showed that our integer programming approach can give improved (optimal) solutions for the examples.

References

Bahl, H. C., Ritzman, L. P., and Gupta, J. N. D. (1987), Determining Lot Sizes and Resource Requirements: a Review, *Operations Research* 35, 329-345.

Brennan, L., Gupta, S. M., and Taleb, K. N. (1994), Operations Planning Issues in an Assembly/Disassembly Environment, *International Journal of Operations and Production Management* 14, 57-67.

Gungor, A. and Gupta, S. M. (1999), Issues in Environmentally Conscious Manufacturing and Product Recovery: a Survey, *Computers and Industrial Engineering* 36, 811-853.

Gupta, S. M. and Taleb, K. N. (1994), Scheduling Disassembly, *International Journal of Production Research* 32, 1857-1886.

Lee, D.-H., Kang, J.-G., and Xirouchakis, P. (2001), Disassembly Planning and Scheduling: Review and Further Research, *Proceedings of the Institution of Mechanical Engineers: Journal of Engineering Manufacture – Part B* 215, 695-710.

Lee, D.-H., Xirouchakis, P., and Züst, R. (2002), Disassembly Scheduling with Capacity Constraints, *Annals of the CIRP* 51(1), 387-390.

Neuendorf, K.-P., Lee, D.-H., Kiritsis, D., and Xirouchakis P. (2001), Disassembly Scheduling with Parts Commonality using Petri-Nets with Timestamps, *Fundamenta Informaticae* 47, 295-306.

O'Shea, B., Grewal, S. S., and Kaebernick, H. (1998), State of the Art Literature Survey on Disassembly Planning, *Concurrent Engineering: Research and Application* 6, 345-357.

Santochi, M., Dini, G., and Failli, F. (2002), Computer Aided Disassembly Planning: State of the Art and Perspectives, *Annals of the CIRP* 51(2), 1-23.

Taleb, K. N. and Gupta, S. M. (1997), Disassembly of Multiple Product Structures, *Computers and Industrial Engineering* 32, 949-961.

Taleb, K. N., Gupta, S. M., and Brennan, L. (1997), Disassembly of Complex Product Structures with Parts and Materials Commonality, *Production Planning and Control* 8, 255-269.