대한산업공학회/한국경영과학회 2002 춘계공동학술대회
한국과학기술원(KAIST) 2002 년 5 월 3 일-4 일

조립구조 형태 제품의 분해 일정계획 문제에 대한 발견적 기법

A Heuristic Approach to Disassembly Scheduling with Assembly Product Structure

Dong-Ho Lee and **Paul Xirouchakis**

Department of Mechanical Engineering (STI-IPR-LICP)
Swiss Federal Institute of Technology – Lausanne (EPFL)
Lausanne, CH-1015
SWITZERLAND

e-mails: {dong-ho.lee, paul.xirouchakis}@epfl.ch

Abstract

Disassembly scheduling is the problem of determining the ordering and disassembly schedules of used products while satisfying the demand of their parts or components over a certain planning horizon. The objective is to minimize the sum of purchase, setup, disassembly operation, and inventory holding costs. This paper considers products with assembly structure, i.e., products without parts commonality, and suggests a heuristic in which an initial solution is obtained in the form of the minimal latest disassembly schedule, and then improved considering tradeoffs among different cost factors. To show the performance of the heuristic suggested in this paper, computational experiments were done on the modified existing examples and the results show that the heuristic can give optimal or very near optimal solutions within very short computation times.

1. Introduction

Disassembly is one of the important remanufacturing or recycling processes, since most products should be disassembled before they are remanufactured or recycled. See O'Shea *et al.* (1998), Gungor and Gupta (1999) and Lee *et al.* (2001) for literature reviews on various disassembly problems.

Among the various disassembly problems, this paper focuses on disassembly scheduling that determines the ordering and disassembly schedules of used products while satisfying the demand of their parts/components over a certain planning horizon. In general, disassembly scheduling is an important short-term (or mid-term) production planning problem in disassembly systems. In other words, from the solution of the problem, we can determine which items (products/subassemblies), how much (amount of disassembly operations), and when (planning period) to disassemble used products in order to satisfy the demand of their parts or components.

Not much work has been done on the disassembly scheduling problem. Gupta and Taleb (1994) define the basic form of the disassembly scheduling problem, i.e., a single product with assembly structure. Since the problem is a reversed form of material requirement planning (MRP), they suggest an MRP-like algorithm. Later, Taleb *et al.* (1997) extend the basic model by considering the parts commonality, and suggest another MRP-like algorithm with the objective of minimizing the number of products to be disassembled. Here, the extended problem is more complex than the basic one since the parts commonality results in one or more alternative procurement sources for each common part and hence creates dependencies among different components. Also, Taleb and Gupta (1997) consider the case with multiple products as well as the parts commonality. Neuendorf *et al.* (2001) suggest a Petri-net approach to solve the problem with parts commonality, and Lee *et al.* (2001) extend the basic problem of Gupta and Taleb (1994) by considering various cost factors, and suggest an integer programming model that can give optimal solutions for small-to-medium sized problems. Recently, Lee *et al.* (2002) suggest another integer programming model for the problem with resource capacity constraints.

The problem considered here is the same as that of Lee *et al.* (2001), i.e., the assembly product structure without parts commonality. Although the integer programming model of Lee *et al.* (2001) can be directly used to solve small-sized problems, it is not adequate for large-sized problems due to its excessive computation times. Therefore, in this paper, we suggest a fast heuristic that can give near optimal solutions within very short computation times.

2. Problem Description

This section begins with the definition of the disassembly product structure, which can be obtained from the solution of the disassembly planning problem that determines the disassembly level, the disassembly sequence, and/or the end-of-life options of the disassembled parts/components. See O'Shea *et al.* (1998) and Lee *et al.* (2001) for more details on disassembly planning.

In the disassembly product structure, the *root item* represents the product itself 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 one child. Note that each item has at most one parent in the assembly product structure without parts commonality. Fig. 1 shows an example of the disassembly product structure, called the GT example in this paper, 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 three units of item 10, two units of item 11, and three units of item 12. Here, item 5 is called parent item, while items 10, 11, and 12 are called child items. Also, in the figure, DLT means *disassembly lead-time,* i.e., the time required to disassemble a parent item.

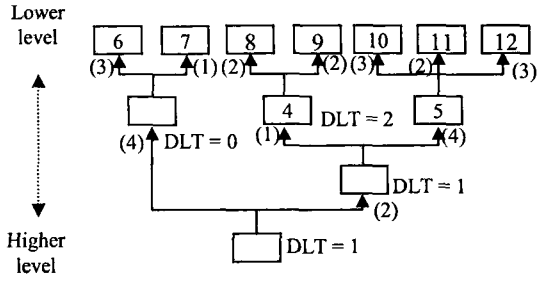

Figure 1. Disassembly product structure of assembly type: an example

The disassembly scheduling problem considered in this paper can be defined as *the problem of determining the ordering and disassembly schedules of the root item and the disassembly schedule of all parent items in the given disassembly structure of assembly type, while satisfying the demand of leaf items over a certain planning horizon with the objective of minimizing the sum of purchase, setup, disassembly operation, and inventory holding costs.* To explain the problem more specifically, additional data of the GT example are summarized in Table 1. The planning horizon consists of 10 discrete periods. Table 1(a) shows the initial inventory for product, subassemblies, and parts at the beginning of the planning horizon, which remains at the end of the previous planning horizon. Table 1(b) shows the scheduled receipts from external sources over the planning horizon, i.e., items that are expected to arrive from outside sources and not from disassembly. Finally, the demand of each leaf item over the planning horizon is summarized in Table 1(c).

Table 1. Data of the GT example

(a) Initial inventory

Item	1	2	3	4	5	6	7	8	9	10	11	12
Initial Inventory	1	6	2	45	12	10	30	55	20	120	90	80

(b) Scheduled receipts from external source

Item	Period									
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	9	0	2
2	5	7	1	9	0	0	0	0	0	0
3	2	4	8	0	0	0	6	0	2	0
4	0	0	15	2	4	8	0	0	0	0
5	1	9	0	2	3	0	0	0	0	0
6	0	15	0	0	2	4	0	0	0	0
7	0	8	0	0	0	0	0	0	0	0
8	1	6	0	2	1	2	0	0	0	0
9	7	1	9	0	0	0	7	2	8	0
10	3	15	2	4	8	0	0	2	5	1
11	0	0	5	7	1	9	0	2	3	0
12	4	7	2	8	1	6	0	2	1	2

(c) Demand

leaf item	Period									
	1	2	3	4	5	6	7	8	9	10
6	0	0	35	0	55	0	45	20	0	188
7	0	0	0	0	15	65	0	36	120	44
8	0	0	0	0	25	0	31	44	96	320
9	0	0	0	0	35	15	0	66	44	96
10	0	0	65	0	35	50	180	720	264	576
11	0	0	50	0	55	70	0	110	480	176
12	0	0	0	0	80	65	0	220	720	264

We assume that the disassembly product structure is given from the corresponding disassembly plan with the description of all parts/subassemblies and disassembly operations. It is assumed that there is no shortage of the root item (product). That is, the ordered products can be obtained whenever they are ordered. The other assumptions made in this problem, which are similar to those of MRP, are as follows: (a) the planning horizon is divided into discrete planning periods; (b) demand of leaf items is given and deterministic; (c) backlogging is not allowed and hence demand should be satisfied in time; (d) the disassembly process is perfect, and hence any defective parts resulted from disassembly are not considered; (e) disassembly lead times with discrete time scale are given and deterministic; and

(f) inventory holding costs are computed based on the end-of-period inventory.

The problem considered here can be formulated as the following integer program, which is a reversed form of the uncapacitated multi-level lot sizing problem (Lee *et al.* 2001). In the formulation, without loss of generality, all items are numbered by the integers 1, 2,...$i_l - 1$, i_l, $i_l + 1$,...I, where the index i_l represents the first leaf item in the disassembly product structure and hence the indices that are equal to or larger than i_l represent leaf items. In the formulation, the following notations are used.

Parameters

- c_t purchase cost of the root item (product) in period t
- s_i setup cost of item i
- p_i disassembly operation cost of item i
- h_i inventory holding cost of item i
- D_{it} demand of item i in period t
- a_{ij} number of units of item j obtained by disassembly of one unit of its parent item i
- r_{it} scheduled receipt of item i in period t
- $\varphi(i)$ parent of item i (Only one parent exists for each item in the assembly structure.)
- l_i disassembly lead time of parent item i
- I_{i0} initial inventory of item i

Decision variables

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

Now, the integer program is given below.

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

subject to

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

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

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

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

$$Z_t \ge 0 \text{ and integers, for all } t = 1,...T \qquad (5)$$

$$X_{it} \ge 0 \text{ and integers,}$$
$$\text{for all } i = 1,... i_l - 1 \text{ and } t = 1,...T \qquad (6)$$

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

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

The objective function denotes the sum of purchase, setup, disassembly operation, and inventory holding costs. Constraints (1), (2), and (3) represent the inventory flow conservation that defines the inventory level of item i at the end of period t. Constraint (1) represents the inventory balance of the root item. That is, at the end of each planning period, we have inventory what we had the period before, increased by the scheduled receipt and the purchased quantity, and decreased by the number of root items disassembled. Constraint (2) represents the inventory balance of parent items (except for the root item) that should be disassembled further. Here, this constraint is the same as (1) except that for each item, the quantity resulting from disassembling its parent, multiplied by the yield from its parent, is used instead of the purchase quantity. Also, the inventory balance of each leaf item is represented by constraint (3), which is different from (2) in that the demand requirement (D_{it}) is used instead of the amount of items to be disassembled. Constraint (4), where M is an arbitrary large number, guarantees that a setup cost in a period is incurred when there is at least one disassembly operation at that period. Finally, the other constraints (5), (6), (7), and (8) represent the conditions of the decision variables. In particular, constraint (7) ensures that backlogging is not allowed. See Lee *et al.* (2001) for more details of the integer programming model.

3. Heuristic Algorithm

The heuristic suggested in this paper consists of two stages. In the first stage, an initial solution is obtained in the form of the minimal latest disassembly schedule (using the algorithm of Gupta and Taleb (1994)). Then, it is improved by iteratively changing the current disassembly schedule, while considering the trade-offs among different cost factors.

Obtaining an initial solution

To obtain an initial ordering and disassembly schedules, the GT algorithm is used in the paper. In the algorithm, the demand of items at one level of the disassembly product structure is translated into an equivalent demand of the items at the next higher level, and this is repeated from leaf items to the root item. See the following procedure for more details. In the procedure, H_i denotes the set of child items of parent i, and the other notations are the same as those of the integer programming formulation.

Procedure 1. (Obtaining Initial Solution)

Step 1. Set $i = i_l - 1$, i.e., parent item with the largest index.

Step 2. For parent item i and its child items, do the following steps:

1) Set $t = 1$.

2) For each child item $j \in H_i$, calculate the Net Requirement (NR_{jt}) in period t:

$$NR_{jt} = \max\{0, R_{jt} - I_{j,t-1} - r_{jt}\},$$

where $R_{jt} = D_{jt}$ if $j = i_l, \ldots I$ (leaf item) and $R_{jt} = X_{jt}$, otherwise.

3) Calculate the disassembly schedule of parent i in period $t - l_i$ using

$$X_{i,t-l_i} = \max_{j \in H_i}\left\{\left\lceil \frac{NR_{jt}}{a_{ij}} \right\rceil\right\},$$

where $\lceil \cdot \rceil$ is the smallest integer value that is greater than or equal to $\cdot$. Here, if $X_{i,t-l_i} > 0$ and $t - l_i \le 0$, stop. (The problem is infeasible.)

4) For each child item j, calculate the inventory level in period t (I_{jt}) as

$$\max\{0, I_{j,t-1} + r_{jt} + a_{ij} \cdot X_{i,t-\varphi(i)} - NR_{jt}\}$$

5) Set $t = t + 1$. If $t > T$, go to Step 3. Otherwise, go to (2).

Step 3. Set $i = i - 1$. If $i = 0$, calculate the ordering schedule (Z_t) and the inventory level (I_{1t}) of the root item using the following procedure and stop. Otherwise, go to Step 2.

1) Set $t = 1$.

2) Calculate the ordering schedule and the inventory level in period t using

$Z_t = \max\{0, X_{1t} - I_{1,t-1} - n_t\}$ and

$I_{1t} = \max\{0, I_{1,t-1} + n_t - X_{1t}\}$

3) Set $t = t + 1$. If $t > T$, stop. Otherwise, go to (2).

Now, it is shown that the initial disassembly schedule obtained by the GT algorithm described in the above procedure is the minimal latest schedule. More formally, for a given disassembly schedule X_{it} for all $i = 1, \ldots i_l - 1$ and $t = 1, \ldots T$, let us assume that a new one X'_{it} is obtained by changing the disassembly schedule of an arbitrary parent item k in two periods u and v, $u < v$ (while the others remain the same) as follows:

$$X'_{ku} = X_{ku} - \alpha$$
$$X'_{kv} = X_{kv} + \alpha, \text{ and}$$
$$X'_{kt} = X_{kt}, \text{ for all } t \neq u, v,$$

where $0 < \alpha \le X_{ku}$. Then, the disassembly schedule X_{it} is called the *latest* if the new disassembly schedule X'_{it} is infeasible (due to backlogging). Also, the disassembly schedule X_{it} is called to be *minimal* if a decrease in the amount of disassembly operations results in backlogging. The following proposition specifies the characteristic of the initial disassembly schedule.

Proposition 1. *The initial disassembly schedule obtained by Procedure* 1 *is the minimal latest one.*

Proof. Among the steps of the procedure, the disassembly schedule is determined using the following formula (Step 2(3)).

$$X_{i,t-l_i} = \max_{j \in H_i}\left\{\left\lceil \frac{NR_{jt}}{a_{ij}} \right\rceil\right\},$$

i.e., the maximum rounded-up ratio of the net requirement for each child item divided by its yield from the parent item i. This implies that the net requirement for item i in period t is satisfied at $t - l_i$ (*as late as possible*) with the *minimum amount* of disassembly operations. Here, l_i is the disassembly lead time of parent i. Therefore, Procedure 1 gives the minimal latest disassembly schedule. ■

Improvement

In this stage, the initial solution is improved using trade-offs among different cost factors. That is, the current disassembly schedule is changed and then the resulting new ordering and disassembly schedules and inventory levels are evaluated. To do this, required are the methods to change the current disassembly schedule, to check the feasibility of the new disassembly schedule, and to calculate new ordering and disassembly schedules and inventory levels.

To change the current disassembly schedule, a backward and extreme move is defined first. Note that the backward and extreme move is a pairwise move in that the change is done between amounts of disassembly operations in two different periods.

Definition 1. For a given disassembly schedule X_{it}, a new one X'_{it} obtained from a *backward and extreme move* of parent item k, between periods u and v ($u < v$ and $X_{kv} > 0$), is defined as

$$X'_{ku} = X_{ku} + X_{kv},$$
$$X'_{kv} = 0, \text{ and}$$
$$X'_{kt} = X_{kt} \text{ for all } t \neq u, v,$$

where the others in the given disassembly schedules (except for item k) remain the same.

The pairwise move defined above is *backward* in that the amount of disassembly operations in the later period (v) is moved to the earlier period (u), and is *extreme* in that the amount of move is the amount of disassembly operations in the later period.

If the initial disassembly schedule is given, it can be easily seen that a forward move (from u to v) always results in an infeasible disassembly schedule because the initial disassembly schedule is the latest one (Proposition 1). This is the reason that backward moves are considered in this paper. Also, we consider extreme moves since the setup cost in the later period can be reduced.

Now, it is needed to check if the new disassembly schedule after a backward and extreme move results in backlogging, i.e., checking the feasibility of the backward and extreme move. In the disassembly scheduling problem considered here, the feasibility checking is different according to different type of items, i.e., root item and non-root parent items. In the case of the root item, a new feasible disassembly schedule can always be obtained by changing the corresponding ordering schedule. On the other hand, in the case of non-root parent items, a backward and extreme move may result in backlogging and hence does not always give a new feasible disassembly schedule. The following proposition shows a simple condition to check the feasibility of a backward and extreme move (for non-root parent items).

Proposition 2. *For a given disassembly schedule X_{it}, a backward and extreme move of non-root parent item k from periods v to u ($u < v$) is feasible if*

$$X_{kv} \le I_{kt} \quad \text{for all } t = u, u+1, \ldots v-1,$$

where I_{it} is the inventory level of item i in period t under the given disassembly schedule.

Proof. The new inventory level, I'_{kt} for $t = u, u+1, \ldots v-1$ of item k, after the backward and extreme move from periods v to u can be calculated as follows.

$$\begin{aligned} I'_{ku} &= I_{k,u-1} + r_{ku} + a_{\varphi(k),k} \cdot X_{\varphi(k),u-l_k} - (X_{ku} + X_{kv}) \\ &= I_{ku} - X_{kv} \end{aligned}$$

$$\begin{aligned} I'_{kt} &= I'_{k,t-1} + r_{kt} + a_{\varphi(k),k} \cdot X_{\varphi(k),t-l_k} - X_{kt} \\ &= (I_{k,t-1} - X_{kv}) + r_{kt} + a_{\varphi(k),k} \cdot X_{\varphi(k),t-l_k} - X_{kt} \\ &= (I_{kt} + r_{kt} + a_{\varphi(k),k} \cdot X_{\varphi(k),t-l_k} - X_{kt}) - X_{kv} \\ &= I_{kt} - X_{kv}, \quad \text{for } t = u+1, \ldots v-1 \end{aligned}$$

$$\begin{aligned} I'_{kv} &= I'_{k,v-1} + r_{kv} + a_{\varphi(k),k} \cdot X_{\varphi(k),v-l_k} \\ &= (I_{k,v-1} - X_{kv}) + r_{kv} + a_{\varphi(k),k} \cdot X_{\varphi(k),v-l_k} \\ &= I_{k,v-1} + r_{kv} + a_{\varphi(k),k} \cdot X_{\varphi(k),v-l_k} - X_{kv} = I_{kv} \end{aligned}$$

Then, the feasibility condition follows from the constraint that $I'_{kt} \ge 0$ for $t = u, u+1, \ldots v-1$. (The others remain the same.) ■

Now, we explain the method to calculate the new ordering schedule and inventory level after the current disassembly schedule is changed using a backward and extreme move. Like the method to check the feasibility described earlier, there are two cases: root item and non-root parent items. In the case of the root item, the backward and extreme move affects the ordering schedule as well as the inventory levels of the root item and its child items, while the others remain the same. In this case, the new ordering schedule and the new inventory levels after the backward and extreme move (between periods u and v ($u < v$)) can be calculated using the following procedure.

Step 1. Set $t = 1$ and $I'_{10} = I_{10}$.

Step 2 Calculate the ordering quantity and the inventory level in period t using

$$Z'_t = \max\{0, X_{1t} - I'_{1,t-1} - r_{1t}\} \text{ and}$$
$$I'_{1t} = \max\{0, I'_{1,t-1} + r_{1t} - X_{1t}\}$$

Step 3. Set $t = t+1$. If $t > T$, stop. Otherwise, go to Step 2.

On the other hand, in the case of a non-root parent item, a backward and extreme move affects the inventory levels of the parent item and its child items, while the ordering schedule and the inventory levels of the other items remain the same. In this case, the new inventory levels can be calculated using the following method. Let the backward and extreme move be done for item k between periods u and v ($u < v$). First, the new inventory level of non-root parent item k after a backward and extreme move between periods u and v is as follows.

$$I'_{kt} = I_{kt} \quad \text{for all } t = 1,2,\ldots u-1, v, v+1, \ldots T, \text{ and}$$
$$I'_{kt} = I_{kt} - X_{kv} \quad \text{for all } t = u, u+1, \ldots v-1.$$

The above formulas result from the fact that the amount of disassembly operations in the later period v is done in the earlier period u and hence the inventory levels in periods $u, u+1, \ldots v-1$ decrease by that amount. Second, the following proposition gives the method to calculate the new inventory levels of child items. Note that this method is worth because it can reduce much computation time to evaluate new disassembly schedules without recalculating the inventory levels of all items

Proposition 3. *New inventory level I'_{jt} of each child item j, $j \in H_i$ in period t, after a backward and extreme move of non-root parent item i from periods v to u ($u < v$) is as follows.*

$$I'_{jt} = I_{jt} + a_{ij} \cdot X_{iv} \quad \text{for } t = u + l_i, \ldots, v + l_i - 1, \text{ and}$$
$$I'_{jt} = I_{jt} \quad \text{for } t = 1,2,\ldots u + l_i - 1, v + l_i, \ldots, T,$$

where a_{ij} is the number of units of child j obtained by disassembly of one unit of its parent i

Proof. For each child item j of the non-root parent item i, $j \in H_i$, the new inventory level (I'_{jt}) in each

period, after the backward and extreme move, can be calculated as follows.

For $t = 1,2,..u + l_i - 1$, $I'_{jt} = I_{jt}$.

For $t = u + l_i$,

$$\begin{aligned} I'_{j,u+l_i} &= I_{j,u+l_i-1} + r_{j,u+l_i} + a_{ij} \cdot X'_{iu} - X_{j,u+l_i} \\ &= I_{j,u+l_i-1} + r_{j,u+l_i} + a_{ij} \cdot (X_{iu} + X_{iv}) - X_{j,u+l_i} \\ &= (I_{j,u+l_i-1} + r_{j,u+l_i} + a_{ij} \cdot X_{iu} - X_{j,u+l_i}) + a_{ij} \cdot X_{iv} \\ &= I_{j,u+l_i} + a_{ij} \cdot X_{iv} \end{aligned}$$

For $t = u + l_i + 1,...,v + l_i - 1$,

$$\begin{aligned} I'_{jt} &= I'_{j,t-1} + r_{jt} + a_{ij} \cdot X'_{i,t-l_i} - X_{jt} \\ &= (I_{j,t-1} + a_{ij} \cdot X_{iv}) + r_{jt} + a_{ij} \cdot X_{i,t-l_i} - X_{jt} \\ &= (I_{j,t-1} + r_{jt} + a_{ij} \cdot X_{i,t-l_i} - X_{jt}) + a_{ij} \cdot X_{iv} \\ &= I_{jt} + a_{ij} \cdot X_{iv} \end{aligned}$$

For $t = v + l_i$,

$$\begin{aligned} I'_{j,v+l_i} &= I'_{j,v+l_i-1} + r_{j,v+l_i} + a_{ij} \cdot X'_{iv} - X_{j,v+l_i} \\ &= (I_{j,v+l_i-1} + a_{ij} \cdot X_{iv}) + r_{j,v+l_i} - X_{j,v+l_i} \\ &= I_{j,v+l_i-1} + r_{j,v+l_i} + a_{ij} \cdot X_{iv} - X_{j,v+l_i} = I_{j,v+l_i} \end{aligned}$$

Then, $I'_{jt} = I_{jt}$ for $t = v + l_i + 1,...T$. ■

Then, based on the method to calculate the new ordering schedule and the new inventory levels, the amount of cost change after a backward and extreme move can be obtained as follows. In the case of the root item, the cost change can be calculated only after we obtain the new ordering schedule and the new inventory levels. Also, in the case of non-root parent items, the backward and extreme move results in the following cost change. First, there is decrease (improvement) in inventory holding and setup costs of the non-root parent item in the later period. More formally,

$$A_i(u,v) = h_i \cdot X_{iv} \cdot (v-u) + s_i,$$

where $A_i(u,v)$ is the amount of cost decrease when there is a backward and extreme move of non-root parent item i between periods u and v. Second, the backward and extreme move incurs cost increase in the inventory holding costs of the child items. In this case, a formal description of the cost increase is as follows.

$$B_i(u,v) = \sum_{j \in H_i} h_j \cdot a_{ij} \cdot X_{iv} \cdot (v-u),$$

where $B_i(u,v)$ is the amount of cost increase of the backward and extreme move (of a non-root parent item).

Now, the improvement procedure using backward and extreme moves is as follows.

Procedure 2. (Improvement)

Step 1. Let the current solution be the initial solution obtained from Procedure 1.

Step 2. Let N denote the set of pairs (i, t) such that $X_{it} > 0$ in the current disassembly schedule. For each pair $(i, t) \in N$, find the best backward and extreme move using

$$u^*(i,t) = \arg\min_{u=1,2,...t-1} \{B_i(u,t) - A_i(u,t)\}$$

If the best move is feasible (proposition 2), set the best move to be the candidate move for the pair (i, t). Otherwise, no candidate move is set to the pair (i, t).

Step 3. If no candidate move exists for each pair (i, t), stop and save the current solution. Otherwise, select the best candidate move.

Step 4. If there is no improvement in the best candidate move, stop and save the current solution. Otherwise, perform the best candidate move and update the current solution. Then, go to Step 2.

4. Computational Experiments

To show the performance of the heuristic suggested in this paper, this section reports the test results of computational experiments on the modified version of the GT example described in the problem description section. Two cases, time-invariant and time-variant purchase costs, are considered in the computational experiments. The time-invariant purchase cost implies that purchase costs are the same in different planning periods, while the time-variant costs imply that purchase costs are different according to different planning periods over the planning horizon. In this test, time-variant purchase costs are considered since the purchase costs of used products depend highly on the market situation.

Two performance measures were used in the test. They are percentage deviations from optimal solution values and CPU seconds. Here, optimal solution values were obtained by solving the formulated integer programs directly using CPLEX 6.5, a commercial integer programming software. In addition, we report how much improvement was obtained in the improvement stage.

For the test, 10 problems with different cost values were generated randomly for each of the two cases on purchase costs. Time-invariant purchase costs were generated from $DU(50, 100)$, while time-variant purchase costs were generated from multiplying randomly generated coefficients, ranged from 0.8 to 1.2, by the time-invariant purchase cost values. Here, $DU(a, b)$ denotes the discrete uniform distribution with range $[a, b]$. Also, setup costs were generated from $DU(100, 1000)$, and disassembly operation costs were generated from multiplying randomly generated coefficients, ranged from 0.05

to 0.1, by the corresponding setup costs. Finally, inventory costs were generated as 5 to 10% of the corresponding purchase costs.

Results of the test on the GT examples are summarized in Table 2, which shows the percentage deviations from optimal solutions. It can be seen from the table that the heuristic suggested in this paper gives very near optimal solutions for each of the two cases on purchase costs, i.e., 0.15% in average for the case of time-invariant purchase costs and 0.49% in average for the case of time-variant purchase costs. Also, compared to the first stage (GT algorithm), more than 1% (in average) of improvement was obtained in the improvement stage, which shows that the improvement procedure is efficient. Table 2(b) compares the CPU seconds of the two-stage heuristic and CPLEX 6.5 to obtain optimal solutions. As expected, CPLEX 6.5 required much longer computation times than the heuristic. The interesting point is that the heuristic required very short computation times, i.e., less than 0.01 seconds.

Table 2. Test results for the GT examples

(a) Percentage deviations from optimal solutions

Problem	Time-invariant purchase costs		Time-variant purchase costs	
	INI†	Heur‡	INI	Heur
1	2.50	0.00*	2.97	0.72
2	2.46	0.00*	2.66	0.28
3	1.82	0.22	2.80	0.83
4	0.89	0.17	1.52	0.57
5	1.14	0.30	1.54	0.70
6	0.64	0.00*	0.89	0.17
7	0.94	0.20	1.19	0.45
8	0.99	0.42	1.02	0.44
9	1.09	0.00*	1.63	0.29
10	0.56	0.15	0.85	0.43
Average	1.30	0.15	1.71	0.49

† Initial solution by the GT algorithm
‡ the heuristic suggested in this paper (after improvement stage)
* optimal

(b) CPU seconds

Problem	Time-invariant purchase costs		Time-variant purchase costs	
	IP	Heur	IP	Heur
1	4.6†	0.01	7.0	0.00
2	7.7	0.00**	3.8	0.00
3	17.9	0.00	16.5	0.00
4	5.4	0.00	12.5	0.00
5	35.4	0.00	7.6	0.00
6	12.4	0.00	17.3	0.00
7	24.5	0.01	9.6	0.00
8	29.7	0.00	14.2	0.01
9	4.5	0.00	4.8	0.00
10	3.7	0.00	3.6	0.00

See the footnotes of (a).
† CPU second of CPLEX 6.5
** CPU second was less than 0.01 second. (personal computer with a Pentium processor operating at 800 MHz clock speed)

5. Concluding Remarks

We considered the disassembly scheduling problem for products with assembly structure, which is the problem of determining the ordering and disassembly schedules of used products, while satisfying the demands of their parts/components over a given planning horizon. To solve the problem, a heuristic, which consists of two stages, is suggested this paper. Computational experiments showed that the heuristic can give optimal or very near optimal solutions within a very short computation time.

Because the disassembly scheduling problem considered in this paper is the most basic form of the problem, this research can be extended in several ways. First, it is needed to consider the problem with general structure, i.e., parts commonality. Second, like other disassembly problems, uncertainties, such as defective parts/components, stochastic demands, are important considerations.

References

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, *Journal of Engineering Manufacture* 215, 695-710.

Lee, D.-H., Neuendorf, K.-P. and Xirouchakis, P. (2001), Disassembly Scheduling for Products with Assembly Structure: Integer Programming Approach, Technical Report, Department of Mechanical Engineering, Swiss Federal Institute of Technology – Lausanne (EPFL), Switzerland.

Lee, D.-H., Xirouchakis, P., Zust, R., (2002), Disassembly Scheduling with Capacity Constraints, to appear in *Annals of the CIRP*.

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.

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.