2005 한국경영과학회/대한산업공학회 춘계공동학술대회
2005 년 5 월 13 일~14 일, 충북대학교

# 자원제약을 고려한 분해 일정계획 문제에 대한 2 단계 발견적 기법

# A Two-Stage Heuristic for Capacitated Disassembly Scheduling

Hyong-Bae Jeon[1], Jun-Gyu Kim[1], Hwa-Joong Kim[2], and Dong-Ho Lee[1]

[1]Department of Industrial Engineering
Hanyang University
Sungdong-gu, Seoul 133-791
KOREA

[2]Institute of Production and Robotics (STI-IPR-LICP)
Swiss Federal Institute of Technology (EPFL)
Lausanne, CH-1015
SWITZERLAND

## Abstract

Disassembly scheduling is the problem of determining the quantity and timing of disassembling used products while satisfying the demand of their parts or components over a planning horizon. The case of single product type with assembly structure is considered for the objective of minimizing the sum of disassembly operation and inventory holding costs. In particular, the resource capacity constraint is explicitly considered. The problem is formulated as an integer programming model, and a two-stage heuristic with construction and improvement algorithms is suggested in this paper. To show the performance of the heuristic, computational experiments are done on a number of randomly generated problems, and the test results show that the algorithm can give near optimal solutions within a very short amount of computation time.

## 1. Introduction

For the last decades, the growth of environmental concerns has stimulated the industry to develop various material and product recovery processes. Disassembly, one of the essential recovery processes, is the process of separating used and/or end-of-life products into their constituent parts, subassemblies, or other groupings. Due to its importance in material and product recovery, previous research has been done in a wide range of disassembly problems such as disassembly sequencing, design for disassembly, disassembly scheduling, etc. For literature reviews on the problems, see Boothroyed and Alting. [1], Jovane *et al*. [3], O'shea *et al*. [12], Lee *et al*. [8], Santochi *et al*. [13], and Lambert [7].

This paper considers disassembly scheduling, which is the problem of determining the quantity and timing of disassembling (used or end-of-life) products so as to satisfy the demand of their parts or components over a finite planning horizon. Disassembly scheduling is one of the important mid- or short-term planning problems in disassembly systems. In other words, from its solution, we can determine which products, how many, and when to disassemble products and/or their subassemblies [8].

Most previous research on disassembly scheduling is uncapacitated ones, i.e., resource capacity constraints are not considered. Gupta and Taleb [2] consider the fundamental case, i.e., single product type without parts commonality, and suggest a simple algorithm without explicit objective function, and Lee and Xirouchakis [10] suggest a heuristic for the objective of minimizing the costs related with disassembly process. For the extended models

with parts commonality, see Kim *et al* [4], Taleb and Gupta [14], and Taleb *et al* [15]. Recently, Lee *et al* [9] present integer programming models for all uncapacitated cases together with their performances using a commercial software package. Several research articles consider the capacitated problems. Lee *et al* [11] consider the fundamental case, and suggest an integer programming model. Although the model can give optimal solutions, its application is limited only to the small sized problems due to its excessive computation time, and Kim *et al* [6] suggest an algorithm that minimizes the number of products disassembled

This paper considers the case of single product type without parts commonality while the resource capacity is explicitly considered. The objective is to minimize the sum of disassembly operation and inventory holding costs, and hence this paper extends the model of Kim *et al* [6] that minimizes the number of product disassembled. In other words, the cost-based objective considered in this paper is more general than that of Kim *et al* [6]. The problem is formulated as an integer programming model, and a two-stage heuristic with construction and improvement algorithms is suggested. Finally, computational experiments are done on a number of randomly generated problems, and the test results are reported.

## 2. Problem Description

Prior to presenting the problem, we first explain the disassembly product structure. In the structure, the root item represents the product itself to be disassembled and each leaf item is the part or component not to be disassembled further. A child item represents any item that has a parent item which has at least two child items. Notice that a child item has only one parent item in the problem considered in this paper, i.e., the case of single product type without parts commonality.

Figure 1 shows an example of disassembly product structure, obtained from Gupta and Taleb [2]. Item 1 is the root item, and items 6 to 12 are leaf items. The number in parenthesis represents the yield of the item when its parent is disassembled, e.g., disassembly of one unit of item 5 results in 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, disassembly lead time (DLT) is the time required to disassemble a certain parent item.

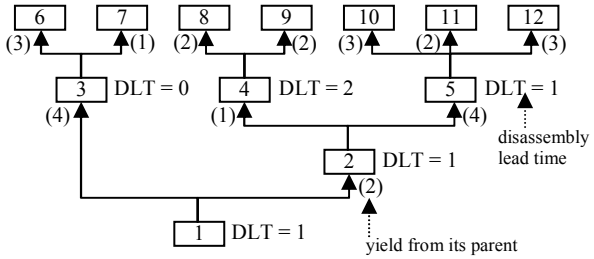


**Figure 1.** Disassembly structure: an example

The capacitated disassembly scheduling problem considered in this paper can be described as follows: *for a given disassembly structure, the problem is to determine the quantity and timing of disassembling each parent item* (*including the root item*) *to meet the demands of leaf items over a planning horizon while satisfying the capacity restriction in each period of the planning horizon.* The capacity restriction in each period is considered as available time at that period, and each disassembly operation consumes a portion of the available time. The objective is to minimize the sum of disassembly operation and inventory holding costs. Here, the disassembly operation cost is proportional to the required labor or machine processing time. It is assumed that cost values are the same over the planning horizon. Also, the inventory holding cost occurs when items are stored to satisfy future demand, and they are computed based on the end-of-period inventory.

The disassembly product structure is assumed to be given by the corresponding disassembly process plan that specifies all disassembly operations and their processing times. Additional assumptions made in this problem are summarized as follows: (a) there is no shortage of the root items, that is, products can be delivered whenever they are needed; (b) the demand of leaf items is given in advance and deterministic; (c) backlogging

is not permitted and hence demands are satisfied on time; (d) parts and/or components resulting from the disassembly are perfect in quality, i.e., no defectives are considered; (e) each disassembly operation is done in only one period and cannot be done over two or more periods; and (f) disassembly lead time is constant and known in advance.

The problem can be formulated as an integer programming model. In the formulation, all items are numbered with integer 1,2, … $i_l$, … $I$. Here, $i_l$ denotes the index for the first leaf item, and therefore the indices that are larger than or equal to $i_l$ represent leaf items. The notations used in this paper are summarized below.

*Parameters*

- $g_i$ disassembly processing time of parent item $i$
- $C_t$ available capacity in period $t$
- $D_{it}$ demand requirement of leaf item $i$ in period $t$
- $a_{ij}$ number of item $j$ obtained by disassembling one unit of item $i$ ($i < j$)
- $s_{it}$ external scheduled receipt of item $i$ in period $t$
- $\phi(i)$ parent of item $i$
- $l_i$ disassembly lead time (DLT) of item $i$
- $I_{i0}$ initial inventory of item $i$
- $p_i$ disassembly operation cost of item $i$
- $h_i$ inventory holding cost of item $i$

*Decision variables*

- $X_{it}$ amount of item $i$ disassembled in period $t$
- $I_{it}$ inventory level of item $i$ at the end of period $t$

Now, the integer program is given below.

[P] Minimize $\sum_{i=1}^{i_l-1}\sum_{t=1}^{T} p_i \cdot X_{it} + \sum_{i=2}^{I}\sum_{t=1}^{T} h_i \cdot I_{it}$

subject to

$$I_{it} = I_{i,t-1} + s_{it} + a_{\varphi(i),i} \cdot X_{\varphi(i),t-l_{\varphi(i)}} - X_{it}$$
for $i$ = 2, 3, … $i_l$ −1 and $t$ = 1, 2, … $T$ (1)

$$I_{it} = I_{i,t-1} + s_{it} + a_{\varphi(i),i} \cdot X_{\varphi(i),t-l_{\varphi(i)}} - D_{it}$$
for $i$ = $i_l$ , $i_l$ + 1… $I$ and $t$ = 1, 2, … $T$ (2)

$$\sum_{i=1}^{i_l-1} g_i \cdot X_{it} \le C_t \quad \text{for } t = 1, 2, \cdots T \quad (3)$$

$X_{it} \ge 0$ and integers
for $i$ = 1, 2, … $i_l$ − 1 and $t$ = 1, 2, … $T$ (4)

$I_{it} \ge 0$ and integers
for $i$ = 2, 3, … $I$ and $t$ = 1, 2, … $T$ (5)

The objective function denotes the sum of disassembly operation and inventory holding costs. Constraint (1) represents the inventory balance of each parent item. That is, at the end of each period, the inventory level of the parent item is what we had before the period, increased by the external scheduled receipt and the quantity obtained by disassembling its corresponding parent item, and decreased by the quantity of the item disassembled in that period. Here, the inventory balance constraint of the root item is not included because it is unnecessary to have surplus inventories of the root item. Also, the inventory balance of each leaf item is represented by constraint (2), which is different from (1) in that the demand requirement is used instead of the amount of items disassembled. Also, constraint (3) represents the capacity constraint in each period. That is, the sum of processing times of disassembly operations assigned to each period should be less than or equal to the given capacity of that period. Finally, the constraints (4) and (5) represent the conditions of the decision variables. Particularly, constraint (5) guarantees that backlogging is not permitted.

## 3. Solution Algorithm

This section presents a two-stage algorithm, in which an initial solution is constructed using a modification of an existing algorithm and then it is improved considering cost changes and capacity constraints. The details of each stage are explained below.

Stage 1: *Obtaining an initial solution*

To obtain the initial solution in this stage, we modify the algorithm of Gupta and Taleb [2], to be called the GT algorithm hereafter, so that the capacity constraints are satisfied. Note that the GT algorithm gives the minimal

latest disassembly schedule. That is, the schedule satisfies the demand of leaf items as latest as possible with the minimum amount of disassembly operations. (See Lee and Xirouchakis [10] for more details on its proof and the basic of the GT algorithm will be explained in Procedure 1 (Step 2).) Therefore, if the solution obtained from the GT algorithm satisfies the capacity constraints, it minimizes the sum of disassembly operation and inventory holding costs, and hence optimal solutions can be obtained. Otherwise, the solution should be modified while increasing the inventory holding costs. Note that although the solution is modified, disassembly operation costs are not increased because we design the modification method in which the total amount of disassembly is not changed.

The basic idea of the solution modification is as follows: First, the minimal latest disassembly schedule of a parent item is determined using the same method as in the GT algorithm. Then, if the disassembly schedule obtained does not satisfy the capacity restriction in a period, the overloaded disassembly quantity in that period is moved to one earlier period. This is done recursively from parent item $i_1 - 1$ to root item.

Now, the procedure of the first stage is summarized below. In the procedure, $H(i)$ denotes the set of child items of parent $i$, $S_i$ denotes the sum of disassembly lead times of items on a path from the root item to item $i$, e.g., $S_5 = l_1 + l_2 + 1 = 3$ in the disassembly structure in Figure 1, and $R_t$ denotes the remaining capacity at period $t$.

**Procedure 1**. (C*onstruction of initial solution*)

Step 1. Set $i = i_1 - 1$, i.e., the parent item with the largest index, and $R_t = C_t$ for all $t$.

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

(a) Set $t = 1$.

(b) For each child item $j \in H(i)$, calculate the Net Requirement($NR_{jt}$) in period $t$ as

$$NR_{jt} = \max\{0, Q_{jt} - I_{j,t-1} - s_{jt}\}$$

where $Q_{jt} = D_{jt}$ if $j = i_1, i_1 + 1, \cdots I$ and $Q_{jt} = X_{jt}$

(c) Calculate the disassembly quantity 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 $\lceil \bullet \rceil$ is the smallest integer that is greater than or equal to $\bullet$.

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

Step 3. Set $t = T$, and for item $i$, do the following steps:

(a) If $R_t \geq g_i \cdot X_{it}$, go to (b). If $t - 1 < S_i$, then stop. Otherwise, calculate the overloaded quantity ($E$) as $E = \lceil (g_i \cdot X_{it} - R_t) / g_i \rceil$ and modify the solution as

$$X_{it} = X_{it} - E \text{ and } X_{i,t-1} = X_{i,t-1} + E .$$

Also, update remaining capacity ($R_t$) using

$$R_t = R_t - g_i \cdot E.$$

(b) For all child items $j \in H(i)$, calculate the inventory level in period $t$ using

$$I_{jt} = \max\{0, I_{j,t-1} + s_{jt} + a_{ij} \cdot X_{i,t-li} - NR_{jt}\}$$

(c) Set $t = t - 1$. If $t = 0$, go to Step 4. Otherwise, go to (a).

Step 4. Set $i = i - 1$, If $i = 0$, stop. Otherwise, go to Step 2.

Notice that the total amount of disassembly is equal to the one obtained by the GT algorithm, and hence the disassembly operation cost is minimized.

Stage 2: *Improvement*

In this stage, the initial solution is improved without violating the capacity and inventory balance constraints. The improvement method suggested in this paper is based on the pairwise move between disassembly quantities assigned to two different periods, while checking the changes of objective value. The pairwise move consists of forward and backward moves. The forward move is done from one later period to one earlier period, while the backward move is reversed. Note that the forward and backward moves are done at the same time.

Before explaining the pairwise move, we make the following assumption:

$$h_i \leq \sum_{k \in H(i)} h_k \cdot a_{ik} \quad \text{for} \quad i = 2, 3, \cdots i_1 - 1$$

This assumption is not very restrictive since the inventory holding cost is directly related to the item value. That is, the item value may increase as the disassembly operations process.

First, we define the forward and backward moves.

*Forward move*

Suppose that $n$ disassembly operations of parent item $i$ in period $t$ are moved period $t+1$. Then, the new schedule of item $i$ in periods $t$ and $t+1$ becomes

$$X'_{it} = X_{it} - n \text{ and } X'_{i,t+1} = X_{i,t+1} + n,$$

and the new inventory levels of item $i$ and its child items are changed as

$$I'_{it} = I_{it} + n \text{ and}$$
$$I'_{k,t+l_i} = I_{k,t+l_i} - n \cdot a_{ik} \quad \text{for } k \in H(i).$$

From these, the movable disassembly quantity $n$ from period $t$ to $t+1$ is selected in the range defined as

$$0 \le n \le \min\{\min_{k \in H(i)} \left\lfloor \frac{I_{k,t+l_i}}{a_{ik}} \right\rfloor, X_{it}\}, \quad (6)$$

which results from the nonnegativity constraints $X_{it} \ge 0$ and $I_{it} \ge 0$. The forward move can decrease the inventory holding cost, which can be calculated as

$$F_i^n = n \cdot (\sum_{k \in H(i)} a_{ik} \cdot h_k - h_i). \quad (7)$$

*Backward move*

Suppose that $m$ disassembly operations of item $j$ are moved from period $t+1$ to $t$. Then, the new schedule of item $i$ in periods $t$ and $t+1$ becomes

$$X'_{jt} = X_{jt} + m \text{ and, } X'_{j,t+1} = X_{j,t+1} - m$$

and the new inventory levels of item $i$ and its child items are changed as

$$I'_{jt} = I_{jt} - m \text{ and}$$
$$I'_{k,t+l_j} = I_{k,t+l_j} + m \cdot a_{jk} \quad \text{for } k \in H(j).$$

From the nonnegativity constraints of the disassembly quantity and the inventory level, the movable disassembly quantity from period $t+1$ to $t$ is selected in the range defined as

$$0 \le m \le \min\{X_{i,t+1}, I_{jt}\}. \quad (8)$$

Unlike the forward move, the backward move increases the inventory holding cost, which can be calculated as

$$B_j^m = m \cdot (h_j - \sum_{k \in H(j)} a_{jk} \cdot h_k) \quad (9)$$

As stated earlier, the improvement in the second stage is done by the pairwise move, i.e., forward and backward moves at the same time. That is, an improved solution can be obtained if the cost decrease of the forward move is greater than the cost increase of the backward move while the capacity and inventory balance constraints are satisfied. The improvement method requires the methods to check the feasibility on the capacity constraints and to determine the amount of moves.

Suppose that $n$ disassembly operations of parent item $i$ are moved from period $t$ to $t+1$, i.e., forward move, and $m$ disassembly operations of item $j$ are moved from period $t+1$ to $t$ at the same time, i.e., backward move. Then, this pairwise move results in the remaining capacities

$$R'_t = R_t + n \cdot g_i - m \cdot g_j, \text{ and} \quad (10)$$
$$R'_{t+1} = R_{t+1} - n \cdot g_i + m \cdot g_j \quad (11)$$

where $R_t = C_t - \sum_{i=1}^{i_l - 1} g_i \cdot X_{it}$. From this, we can see that the pairwise move is feasible if both remaining capacities are nonnegative.

Now, we explain the method to determine the amount of moves. In the move, we determine the best ($n^*$, $m^*$) using

$$(n^*, m^*) = \arg\max_{(n,m)} \{F_i^n + B_j^m\}.$$

In the method, we set initially the forward move amount $n$ as

$$n = \min\{\min_{k \in H(i)} \left\lfloor \frac{I_{k,t+l_i}}{a_{ik}} \right\rfloor, X_{it}\},$$

which is the maximum value in the range (6). Then, the amount $n$ of forward move is decreased one by one until $n = 0$, which is the minimum value in the range (6). Here, given amount $n$, the backward move amount $m$ is determined as

$$m = \max\{\left\lceil \frac{(n \cdot g_i - R_{t+1})}{g_j} \right\rceil, 0\}$$

since the backward move amount $m$ should have the minimum value in the range (8) since

the backward move results in the increase of costs as in (7) and also remaining capacities after the move should not be nonnegative, which are defined in (10) and (11).

The procedure of the second stage is summarized below.

**Procedure 2.** (*Improvement*)

Step 1. Set $i = 1$

Step 2. For parent $i$, do the following steps:

(a) Set $t = 1$ and $j = i + 1$

(b) Find the best backward and forward moves ($n^*$, $m^*$) using the method explained earlier. If the best move results in cost reduction, update the solution by performing the pairwise move.

(c) Set $j = j + 1$. If $j > i_l - 1$, set $t = t + 1$. If $t > T$, go to Step 3 and otherwise, go to (b).

Step 3. Set $i = i + 1$, if $i > i_l - 1$, stop. Otherwise, go to Step 2.

## 4. Computational Experiments

To show the performance of the two-stage heuristic suggested in this paper, computational experiments were done on a number of randomly generated test problems. Also, the two-stage heuristic is compared with the algorithm of Kim *et al.* [6] (without and with the improvement method) although it is developed to minimize the number of products disassembled. Two performance measures were used in this test: percentage deviation from the optimal solution value and CPU seconds. Here, optimal solution values were obtained by solving problem [P] directly using CPLEX 9.0. The algorithm and the program to generate integer programs were coded in C and tests were done on a personal computer with a Pentium processor operating at 2.0 GHz clock speed.

For the test, 25 problems were generated for each combination of two levels of capacity tightness (loose and tight), five levels of the number of items (10, 20, 30, 40 and 50) and three levels of the number of periods (10, 20and 30). For each level of the number of items, 5 disassembly products structures (and hence totally 15) were randomly generated. In the disassembly structures, the number of child items for each parent and its yield was generated from $DU(2,5)$ and $DU(1,3)$ respectively. Here, $DU(a, b)$ is the discrete uniform distribution with range $[a, b]$. Disassembly lead time was set to 0, 1, and 2 with probabilities, 0.2, 0.7, and 0.1, respectively. External scheduled receipt and initial inventory levels were set to 0 without loss of generality.

For each disassembly structure, 5 problems with different data were generated for each level of the number of periods. Disassembly operation costs were generated from $DU(50,100)$, inventory holding costs were generated from $DU(5,10)$. Capacity per period was set to 400, 480, and 540 with probabilities, 0.2, 0.5, and 0.3, respectively. Disassembly time was generated from $DU(1,4)$. The demand is generated using the following procedure:

(1) Demand is initially generated from 0 or $DU(50,200)$ with probabilities 0.1 and 0.9, respectively;

(2) With the demand, the problem is solved using the GT algorithm

(3) The overall capacity usage($CU$) of its solution is calculated using

$$CU = \sum_{i=1}^{i_l-1} \sum_{t=1}^{t=T} g_i \cdot X_{it},$$

where $X_{it}$ is the solution of the GT algorithm.

(4) Demand is regenerated using

$$d_{it} = \lfloor \alpha \cdot TC / CU \cdot d'_{it} \rfloor,$$

where $\alpha$ is a parameter that represents capacity tightness ($\alpha$ is set to 0.7 and 0.9 for the cases of loose and tight capacity tightness), $TC$ is the sum of capacities over the planning horizon and $d_{it}'$ is the demand generated initially in (1).

The test results are summarized in Table 1(a) and (b) that shows the average percentage deviations from the optimal solution values and average CPU seconds. It can be seen from the tables that the two-stage heuristic gives very near optimal solutions: within 0.7% in overall average for the case of tight capacity and 0.1% for the case of loose capacity. Also, significant improvements can be obtained in the second stage, which shows the effectiveness of the improvement procedure. However, the two-stage heuristic could not give feasible solutions for a few problems in the case of tight capacity. In this

case, the algorithm of Kim *et al* [6] with the improvement stage can be used although its performance is not better than the two-stage heuristic. Finally, the CPU seconds of the two-stage heuristic were much smaller than CPLEX. In fact, in the case of tight capacity, CPLEX requires long CPU seconds and they are not consistent. Therefore, we can argue that the two-stage heuristic can be used to solve the practical sized problems.

**Table 1.** Test results for the suggested algorithm

(a) Case of tight capacity

| Number of items | Number of periods | Percentage deviations | | CPU seconds | |
|---|---|---|---|---|---|
| | | Two-stage | Kim *et al.*[1] | Two-stage | CPLEX |
| 10 | 10 | 4.5 (1.3)[2] | 7.7 (6.8)[3] | 0.000* | 0.8 |
| 10 | 20 | 2.7 (0.6) | 13.9 (13.2) | 0.000 | 164.0 |
| 10 | 30 | 3.5 (0.6) | 14.6 (14.0) | 0.000 | 4.3 |
| 20 | 10 | 2.0 (0.3) | 12.5 (09.7) | 0.000 | 6.7 |
| 20 | 20 | 3.1 (0.9) | 17.8 (15.0) | 0.000 | 226.9 |
| 20 | 30 | 2.2 (0.5) | 22.4 (19.5) | 0.000 | 532.8 |
| 30 | 10 | 3.7 (0.6) | 10.2 (6.6) | 0.000 | 0.4 |
| 30 | 20 | 2.7 (0.7) | 14.9 (11.7) | 0.000 | 21.6 |
| 30 | 30 | 2.0 (0.3) | 17.2 (15.0) | 0.000 | 0.7 |
| 40 | 10 | 2.6 (0.7) | 8.4 ( 5.3) | 0.001 | 3.5 |
| 40 | 20 | 3.4 (0.8) | 16.9 (13.1) | 0.001 | 436.1 |
| 40 | 30 | 1.7 (0.5) | 20.6 (16.9) | 0.000 | 443.1 |
| 50 | 10 | 3.6 (1.3) | 8.5 (5.0) | 0.002 | 0.2 |
| 50 | 20 | 3.3 (0.8) | 18.0 (13.2) | 0.000 | 20.5 |
| 50 | 30 | 1.8 (0.2) | 24.2 (19.9) | 0.001 | 2.9 |

[1] The algorithm of Kim *et al* [6]
[2,3] average percentage deviation out of 25 problems without and with (in parenthesis) the improvement stage procedure
* Average CPU seconds is less than 0.0005s

(b) Case of loose capacity

| Number of items | Number of periods | Percentage deviations | | CPU seconds | |
|---|---|---|---|---|---|
| | | Two-stage | Kim *et al.* | Two-stage | CPLEX |
| 10 | 10 | 0.8 (0.1) | 26.0 (24.3) | 0.000 | 0.02 |
| 10 | 20 | 0.6 (0.1) | 43.5 (42.6) | 0.000 | 0.03 |
| 10 | 30 | 0.1 (0.06) | 40.9 (40.5) | 0.000 | 0.04 |
| 20 | 10 | 1.1 (0.4) | 39.9 (35.3) | 0.000 | 0.04 |
| 20 | 20 | 0.13 (0.11) | 55.1 (51.8) | 0.000 | 0.04 |
| 20 | 30 | 0.13 (0.01) | 59.7 (57.1) | 0.000 | 0.10 |
| 30 | 10 | 0.4 (0.1) | 33.2 (29.5) | 0.000 | 0.03 |
| 30 | 20 | 0.2 (0.05) | 49.0 (44.7) | 0.000 | 0.06 |
| 30 | 30 | 0.1 (0.01) | 54.5 (51.0) | 0.001 | 0.11 |
| 40 | 10 | 1.0 (0.2) | 30.8 (23.2) | 0.000 | 6.96 |
| 40 | 20 | 0.1 (0.02) | 46.1 (40.1) | 0.001 | 0.11 |
| 40 | 30 | 0.1 (0.01) | 54.2 (49.0) | 0.000 | 37.73 |
| 50 | 10 | 1.4 (0.3) | 28.4 (22.3) | 0.000 | 0.11 |
| 50 | 20 | 0.2 (0.03) | 47.7 (40.8) | 0.001 | 0.12 |
| 50 | 30 | 0.03 (0.01) | 58.0 (52.5) | 0.001 | 0.21 |

## 5. Concluding Remarks

This paper considered the capacitated disassembly scheduling problem with single product type without parts commonality for the objective of minimizing the sum of disassembly operation and inventory holding costs. To solve the problem, a two-stage heuristic, which consists of construction and improvement algorithms, was suggested. Computational experiments on a number of randomly generated test problems show that the two-stage heuristic can give optimal or very near optimal solutions within very short amount of computation time.

This research can be extended in several ways. First, it is needed to consider more general cases such as multiple product types with part commonality. Here, the part commonality introduces one or more procurement sources for each common part and hence makes the problem difficult to solve. Second, like other disassembly problems, uncertainties such as stochastic demand, defective parts/components, and stochastic disassembly operation times are important further considerations.

## References

1. Boothroyd, G. and Alting, L., 1992, Design for assembly and disassembly, *Annals of the CIRP* 41, 625-636
2. Gupta, S. M. and Taleb, K. N., 1994, Scheduling disassembly, *International Journal of Production Research* 32, 1857-1886.
3. Jovane, F,. Alting, L., Armoillotta, A., Eversheirm, W., Feldmann, K., Seliger, G., and Roth, N., 1993, A key issue in product life cycle: disassembly, *Annals of the CIRP* .42, 651-658
4. Kim, H.-J., Lee, D.-H., Xirouchakis, P., and Züst, R., 2003, Disassembly scheduling with multiple product types, *Annals of the CIRP* 52, 403-406.

5. Kim, H.-J., Lee, D.-H., and Xirouchakis, P., 2004, A Lagrangean heuristic algorithm for disassembly scheduling with capacity constraints, Technical Report, Department of Mechanical Engineering, Swiss Federal Institute of Technology – Lausanne (EPFL).
6. Kim, J.-G., Jeon, H.-B., Kim, H.-J., Lee, D.-H., and Xirouchakis, P., 2005, Capacitated disassembly scheduling: minimizing the number of products disassembled, to appear in *Lecture Note in Computer Science*.
7. Lambert, A. J. D., 2003, Disassembly sequencing: a survey," *International Journal of Production Research* 41, 3721-3759.
8. 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 Manu-facture-Part B* 215, 695-710.
9. Lee, D.-H., Kim, H.-J., Choi, G., and Xirouchakis, P., 2004, Disassembly scheduling: integer programming models, *Proceedings of the Institution of Mechanical Engineers: Journal of Engineering Manufacture-Part B* 218, 1357-1372.
10. Lee, D.-H. and Xirouchakis, P., 2004, A two-stage heuristic for disassembly scheduling with assembly product structure, *Journal of the Operational Research Society* 55, 287-297.
11. Lee, D.-H., Xirouchakis, P., and Züst, R., 2002, Disassembly scheduling with capacity constraints", *Annals of the CIRP*.51, 387-390.
12. 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.
13. Santochi, M., Dini, G., and Failli, F., 2002, Computer aided disassembly planning: state of the art and perspectives, *Annals of the CIRP*.51, 1-23.
14. Taleb, K. N. and Gupta, S. M., 1997, Disassembly of multiple product structures, *Computers and Industrial Engineering* 32, 949-961
15. 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.