Keywords: Disassembly Scheduling, Remanufacturing, Multi-Objective, Optimization, Lost Sales, Nsga-II.

Abstract: Disassembly scheduling is one of the important problems in reverse logistic decisions. This paper focuses on this problem with capacity restrictions on disassembly resources, lost sales, multiple products and without part commonality. A model with two objectives is developed and optimized by a multi-objective approach. The first objective is a sum of several costs to minimize: setup cost, inventory cost, and over capacity penalty cost. The second objective is a measure of the service level. Considering the complexity of this model, a genetic algorithm is developed (NSGA-II) to obtain a set of Pareto-optimal solutions, the results are compared with those calculated by a mixed integer programming model. Results of computational experiments on randomly generated test instances indicates that the genetic algorithm gives good quality solutions up to all problem sizes in a reasonable amount of computation time whereas linear programming solvers do not give solution in reasonable time.

1 INTRODUCTION

Nowadays, due to environmental and economic reasons, more and more companies acknowledge that reverse logistic is a part of the supply chain as important as production or distribution. Disassembly process consists in separating recovered products to generate components which can be reused or be conditioned safely for the environment. Disassembly scheduling defines how many products to disassemble given the demands for components in each period of a finite horizon planning. In this paper we consider two-level product structure disassembly scheduling problem with setup times, lost sales, multi-products types and limited capacity. Part commonality between products is not considered in this paper.

The goal of this study is to develop an optimization tool for this problem with two objective: total cost and service level. To our knowledge, the disassembly scheduling problem with lost sales has not been studying in literature. Lost sales allow selecting demand to be satisfied and minimizing inventory surplus that is inherent to disassembly scheduling problem. In the following section, we start by a literature review of disassembling problems. In section 3, a mathematical formulation of the problem is introduced. In section 4, a meta-heuristic based on genetic algorithm NSGA-II is developed for large instance when CPLEX solver do not give solutions in reasonable time. Finally, section 5 explores the performances of the meta-heuristic and compares results with solutions given by solving the mixed integer programming model. Concluding remarks and future research goals will be given in section 6.

2 LITERATURE REVIEW

In this section we present various problems in disassembly system studied in literature. Gupta and Taleb (1994) defined and characterized the basic disassembly scheduling problem for a single product type, without explicit objective function and suggested an algorithm that is a reversed Material Requirement Planning (MRP). This problem was further extended to include commonality parts by Gupta and Taleb (1997) for multiple product case. Disassembly scheduling can be classified into deterministic and non-deterministic problems which incorporate random factors in the models, Inderfurth and Langella (2006) developed two heuristics which take into consideration stochastic disassembly yields, with multiple product types, parts commonality, two-level product structure. Here we interested on deterministic problems. When set up costs is considered in the objective function, lost sizing decision have to be made. We note that methodologies for lot sizing in production and...
assembly scheduling cannot be applied to disassembly due to their divergence characteristic, see Kim et al. (2007) for more details of the divergence characteristic. Resource capacity restriction also complicate the problem. Lee et al. (2002) considered the capacitated problem, and proposed an integer programming model for the case of single product type. Lee and Xirouchakis (2004) and Kim et al. (2003) proposed integer programming models to determine the disassembly scheduling of used products in order to satisfy the demand of their parts over a planning horizon, considering various situations involving costs and capacity. Kim et al. (2006) developed a two-phase heuristic to minimize of set up, disassembly operation and inventory-holding costs. Lee et al. (2006) developed an integer programming model considering capacity restriction, a two stage solution approach is proposed. Barba et al (2008) present an algorithm for reverse MRP with various lot sizing heuristics. Kim et al. (2010) consider the problem that minimizes the total cost that is sum of setup cost and inventory holding cost, they suggested a branch and bound algorithm that incorporates Lagrangian relaxation technique to obtain good lower and upper bounds. In this study we test the model with their instances. Kim and Lee (2011) proposed a heuristic for multi-period disassembly leveling and scheduling. To our knowledge, there is no study on disassembly lot sizing with lost sales.

There are several references on production lot sizing with lost sales. Xiao Liu and Freng Chu (2004) address the capacitated lot sizing problem with lost sales, they developed a dynamic programming algorithm to solve the problem. Absi et al (2013) deals with the same problem, they proposed a non-myopic heuristic based on a probing strategy and refining procedure. Their approaches can not be applied in disassembly. Indeed, there is one supply product source for several component demands and hence when a component demand is satisfied and may be cause stockout or inventory surplus for others components.

Various objectives can be considered in lot sizing problems. Jafar and Mansoor (2011) addressed the lot-sizing problem with supplier selection, they developed two multi-objective mixed integer non-linear models for multi-period lot-sizing problems with multiple products and multiple suppliers, three objectives are considered cost, quality and service level. Ayyuce et al. (2013) deals with multi-objective optimization of a stochastic disassembly line balancing problem, they proposed a genetic algorithm which generates Pareto-optimal solutions considering two different fitness evaluation approaches.

To the best of the authors’ knowledge, no one has addressed the optimization of capacitated disassembly scheduling with lost sales and multi-objective approach. In this paper we compare an exact method for mono-objective and a meta-heuristic for multi-objective.

3 MODEL FORMULATION

In this section we present the mixed integer programming model of the problem. Before formulating the mathematical model, the disassembly process is described first.

A parent (root) item can be disassembled to produce a specific number of child (leaf) items. Given a set of root items, the demand of each leaf items of all roots is given over a time horizon. Each period has a normal production capacity, exceeding this capacity will result a penalty cost. If the demand of a leaf item is not met in a period it will be considered as lost sales. The problem is to determine the quantity and timing of disassembling all root items to satisfying demand of their leaf items over the planning horizon subject to capacity restrictions in each period, respecting a particular service level.

In this paper we consider two objectives: total cost and service level. The first objective is to minimize the sum of purchase, inventory holding, and disassembly costs. The second one is to maximize the service level. The cost of not satisfied demand is difficult to assess and we cannot combine cost and quantity in the same objective function, thus we consider in this model one objective (Total cost) and we include the second (Service level) as a constraint.

A. Model parameters and decision variables

The notations used are summarized below.

Indices:

\[ r \] Index for root items, \( r = 1, 2, \ldots, R \)

\[ i \] Index for leaf items, \( i = 1, 2, \ldots, N \)

\[ t \] Index for periods, \( t = 1, 2, \ldots, T \)

Parameters

\[ \delta_r \] Setup cost of parent item \( r \).

\[ C_t \] Capacity available, in time, in period \( t \).

\[ \varphi_i \] Parent of leaf item \( i \).

\[ g_r \] Disassembly operation time of root item \( r \).

\[ h_i \] Inventory holding cost of item \( i \).

\[ p_{it} \] Penalty cost disassembly time in period \( t \).

\[ d_{it} \] Demand of item \( i \) in period \( t \).
Number of unit of items i obtained by disassembly of one unit of its parent item r.

\( \text{Initial inventory of item i.} \)

\( M \) Large Number.

\( \text{Maximal lost sales level (\%)} \)

\( \text{Decision variables} \)

\( Y_{rt} = 1 \) if there is a setup for root item r in period t, 0 otherwise.

\( X_{rt} \) Disassembly quantity of root item r through period t.

\( OT_t \) Disassembly over-time in period t.

\( L_{it} \) Inventory level of leaf item i at the end of period t.

\( L_{it} \) Lost sales for each leaf item i in period t.

\( B. \text{Model assumptions} \)

Assumptions made in this model are summarized as follows:

(a) Demands for leaf items are given and deterministic;

(b) Lost sales is allowed, hence demand can be not satisfied;

(c) The disassembly process is perfect, all parts are in perfect quality, no defective are considered;

(d) Disassembly operation times are given and deterministic;

\( C. \text{Mathematical formulation} \)

In this study we solve the problem using two approaches.

The first case is a mixed integer program (MIP) where the first objective is the objective function and the second objective is a constraint. The constraint level is varying it to obtain different solutions for the same instance.

We note that in this model we consider the total loss sales level which can be calculated as:

\[ \text{TotalLostSalesLevel} = \sum_{t=1}^{T} \sum_{i=1}^{N} L_{it}/\sum_{t=1}^{T} \sum_{i=1}^{N} d_{it} \]

and then total service level can be deducted:

\[ \text{TotalServiceLevel} = 1 - \text{TotalLostSalesLevel} \]

With above parameters and decision variables, the MIP is given bellow.

\[ \begin{align*}
\text{Minimize} & \quad z = \sum_{t=1}^{T} \sum_{r=1}^{R} Y_{rt} \times s[r] + \sum_{t=1}^{T} \sum_{i=1}^{N} l_{it} \times h_i + \sum_{t=1}^{T} OT_t \times p_t \\
\text{Subject to} & \quad l_{it} = l_{it+1} + a_{r,i} + X_{r,0} + L_{it} - d_{it} \quad \text{for all} \quad t=2,\ldots,T \quad \text{and} \quad i=1,\ldots,N \\
& \quad X_{rt} \leq M \times Y_{rt} \quad \text{for all} \quad t=1,\ldots,T \quad \text{and} \quad r=1,\ldots,R \\
& \quad (\sum_{r=1}^{R} g_{r} + X_{rt}) - C_t \leq OT_t \quad \text{for all} \quad t=1,\ldots,T \\
& \quad L_{it} \leq d_{it} \quad \text{for all} \quad t=1,\ldots,T \quad \text{and} \quad i=1,\ldots,N \\
& \quad \sum_{t=1}^{T} \sum_{i=1}^{N} d_{it} \leq L_{Max} \\
& \quad X_{rt}, l_{it} \geq 0
\end{align*} \]

Objective function (1) is the Total cost which is the sum of setup cost, expected inventory holding and penalty costs, production costs are not considered in this study.

The constraint are the following:

- (2) Define the inventory flow conservation of leaf items at the end of each period (\( l_{t,0} \)) is an input data.
- (4) Ensure that a setup is performed in a period when disassembly operation is performed.
- (5) Enforces the capacity feasibility.
- (7) State the upper bound available of lost sales level; we note that maximizing the total service level equivalent minimizing the total lost sales level.
- (8) Defines the domain of variables.

The second case we solve the problem by using the NSGA-II algorithm that considers multiple objectives:

\[ \begin{align*}
\text{Minimize} & \quad z_1 = \sum_{t=1}^{T} \sum_{r=1}^{R} Y_{rt} \times s[r] + \sum_{t=1}^{T} \sum_{i=1}^{N} l_{it} \times h_i + \sum_{t=1}^{T} OT_t \times p_t \\
& \quad h_i + \sum_{t=1}^{T} OT_t + p_t \\
\text{Minimize} & \quad z_2 = \sum_{t=1}^{T} \sum_{i=1}^{N} L_{it}
\end{align*} \]

Subject to : Constraints (1) to (6) and (8).

4 MULTI-OBJECTIVE GENETIC ALGORITHM

Generally, based on a population search Multi-Objective Evolutionary Algorithm (MOEA) can present a set of non-dominated or Pareto optimal solutions. In this study we consider two objectives, total cost and service level. To solve the model in this paper we use Non-dominated Sorting Genetic Algorithm II (NSGA-II), one of the MOEAs frequently used in many optimization problems as the best technique to generate Pareto frontiers, which has been proposed by Deb et al. (2000). Moreover, the NSGA-II has been consistently uses in several research articles which deals with supply chain problems see Godichaud et al., D. Sanchez et al. and Li et al.

D. NSGA-II Principle

This algorithm uses a fixed-sized population. We start by initializing the population then the population is sorted based on non-domination criteria into several fronts. The first front is a completely non-dominated set in the current population and the
second front being dominated by the individuals in
the first front only and so on. Each individual in each
front is assigned fitness value. We said that solution
\( p \) is dominated by solution \( q \) if only \( q \) is better than \( p \)
with regard to all objectives, or \( q \) is better than \( p \)
with regard to other objectives. This process is
continues until all fronts are identified. In addition to
fitness value we calculate the crowding distance
which is a measure of how close an individual is to
its neighbors, we used it in order to maintain
diversity in the population.

E. NSGA-II algorithm
Create initial population \( P_0 \), of size \( n \); Create child population \( Q_0 \) using binary tournament
selection, recombination and mutation;

While (stopping criterion)
We create a new population \( R_t \) which
combine \( P_t \) (parent) and \( Q_t \) (child)
Sort \( R_t \) by non-domination
Assign a fitness equal to its non-domination level
for each solution, identify levels \( F_i, i = 1,2, \ldots \)
Computed the crowding distance of each solution
Set new population \( P_{t+1} \)
Set \( i = 1 \)
While \( |P_{t+1}| + |F_i| \leq n \) do
Add \( F_i \) to \( P_{t+1} \)
Set \( i = i + 1 \)
end while
Set \( \text{Diff} = n - |P_{t+1}| \)
If \( \text{Diff} \neq 0 \) Sort solutions by descending crowding
distance
for \( j = 1 \) to \( \text{Diff} \) do
Add Solution \( F_i \) of \( F_i \) to \( P_{t+1} \)
end for
end if
end while

F. Encoding
In this study, the decisions variables are
\( Y_{rt}, X_{rt}, OT_t, I_t \) and
\( L_{it} \), among which \( Y_{rt} \) is a binary variable (0-1), and
the others are positive variables of integer numbers.
Generally, in literature, setup variables are used to
code solutions and integer variables are deduced
based of the properties of the model. These properties
are not sufficient for the problem with with lost sales
and integer variables can not be computed from \( Y_{rt} \),
and thus, we encode \( X_{rt} \) as chromosomes.

G. Initial population
In this study, population of candidate solution \( P_0 \) is
randomly generated according to an uniform
distribution. We use a random integer generator for
\( X_{rt} \) with respect to the bounding conditions.

H. Selection and Evaluation
Capacity constrains and objective functions are used
to evaluate the objectives of each chromosome, we
note that there are two objective function values for
each one. We use the constrained tournament
method because of its ability to satisfy constraints
and at the same time perform selection based on
fitness.
This operator involves running several tournaments
among a few individuals chosen at random from the
population and the one with best fitness (winner) is
selected for crossover.

I. Crossover and Mutation
One crossover point is used. Genes from beginning
of chromosome to the crossover point is copied from
one parent, and the rest is copied from the second
parent, at the end we obtained two children. After
this we mutate on chromosome by changing one
more variable in some way by random. Crossover
and mutation are performed with a given probability.
Values are mentioned in the next section.

5 COMPUTATION
EXPERIMENTS

The NSGA-II algorithm tested in this paper was
-coded in Java and run on a personal computer with a
five processors operating at 2.60 GHz clock speed.

J. Test instances
For the test, instances are generated as in (H.-J.
Kim and P. Xirouchakis, 2010). U(a,b) is the discrete
uniform distribution with a range of \([a,b]\).

- We generated 10 instances for each number of
  root items (10,20,30), three number of
  children generated from a discrete uniform
distribution with a rang U(1,10), U(10,100),
  and U(100,1000) for low, medium, and large
  respectively ,and three number of periods
  (10,20,30);
- \( s_r \) : Setup cost for each root was generated
  from U(1000,5000);
- \( a_i \) : For each root the number of child were
  generated from U(1,5);
- \( d_{it} \) :Demand was generated from U(50,200);
- \( h_i \) : Inventory holding costs were generated
  from U(1,10);
- \( p_t \) : Penalty costs for overtime were
  generated from U(5,15);
Disassembly time was generated from \( U(1,3) \);

Initial inventory was generated from \( (20,100) \);

Available aggregate capacity in each period is set to 540,480 and 400 with probabilities, 0.3,0.5,0.2

**K. Parameters setting**

Different tests with different parameters were made to choose the efficient parameters for the algorithm. The following control parameters for genetic algorithm are the ones we used in our case study:

- Maximum generation \( Nbr\text{-}Ittr = 1500 \).
- Population size \( NbPop = 100 \).
- Mutation probability \( Coef_m = 0.2 \).
- Crossover probability \( Coef_c = 0.9 \).

**L. Computational Results**

In this section, we apply the GA discussed earlier to solve the model proposed and to show the effectiveness of our GA meta-heuristic firstly we compare the NSGA-II performances with those of Cplex 12.5 software, in terms of computation time and solution quality to solve the small-sized problem and after we present the strength of the NSGA-II to solve all sizes instances.

The figure 1 present Pareto front obtained with NSGA-II for the first instance (10 periods, 10 roots, low number of children), the Pareto contains 100 solutions. We present also 11 exact algorithm solutions solved with mono-objective for 11 different percentages of lost sales level. We note that each solution obtained is a point of the optimal Pareto front.

To show the quality of our results we will compare the two fronts: the first obtained by Cplex (optimal pareto solution) and the second obtained by NSGA-II.

![Figure 1: Example of Pareto front obtained with GA (100 solutions) and some exact algorithm solutions (11 solutions).](image)

To compare our curves we use the Hyper Volume indicator. Readers wishing more detailed description of the algorithm can be referred to Deb.

Table 1 present the hyper volume values:

<table>
<thead>
<tr>
<th>Hyper Volume</th>
<th>Cplex</th>
<th>NSGA-II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.829</td>
<td>0.826</td>
</tr>
</tbody>
</table>

The gap indicates that the front solutions of NSGA-II is 99.64% close to the optimal front obtained by Cplex. Moreover the decision maker has several choices in terms of solutions (100 by NSGA-II against 11 by Cplex).

Here, the heuristic solutions are compared with Cplex solutions to assess the benefits of increasing the CPU time limit. Concerning the exact method (case 1), we solve the problem on mono-objective. This table summarizes the computation time of one instance with 10 periods, 10 roots and low number of children. As mentioned in mathematical model there is a constraint of lost sales level: \( LMax \), in this experiment we change the \( LMax \) value and we evaluate the objective function. In this test we used Cplex software to obtain solutions.

For this instance for each lost sales level value we allowed Cplex to run for maximum 3000sec to avoid excessive computation times and we fixed the absolute tolerance on the gap between the best integer objective and the objective of the best node remaining at 0.01.

<table>
<thead>
<tr>
<th>Lost Sales level (%)</th>
<th>Objective CPU(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.71</td>
</tr>
<tr>
<td>10</td>
<td>5.89</td>
</tr>
<tr>
<td>20</td>
<td>49.30</td>
</tr>
<tr>
<td>30</td>
<td>1138.43</td>
</tr>
<tr>
<td>40</td>
<td>1804.52</td>
</tr>
<tr>
<td>50</td>
<td>466.61</td>
</tr>
<tr>
<td>60</td>
<td>98.88</td>
</tr>
<tr>
<td>70</td>
<td>59.77</td>
</tr>
<tr>
<td>80</td>
<td>21.39</td>
</tr>
<tr>
<td>90</td>
<td>6.19</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
</tbody>
</table>

| Total time            | 3652.69             |

We reported CPU time in second for the instance.
example in table 2, the total time to compute solutions for each percentage of lost sales level (11 solutions) is:

\[ Total_{\text{exact}} = \sum_{\text{Percentage}} T_{\text{Objective}} \]

\[ Total_{\text{exact}} = 3652.69 \text{sec} \]

On the other side the total time to obtain the Pareto front which provides 100 solutions with the NSGA-II algorithm (considering mathematical model case 2) for the same instance is: Total_{\text{NSGA-II}} = 4\text{sec} (see the table 2 in the next section). We kept a large number of solution to analyse the behaviour of the algorithm. In practice, decision makers have to choose only one solution based on its preferences. Multi criteria decision making can be used to this end with the NSGA-II solutions as an input.

\[ Total_{\text{NSGA-II}} \ll Total_{\text{exact}} \]

(4sec \ll 3652.69sec)

Table 3: CPU time in seconds of Kim problem instances.

<table>
<thead>
<tr>
<th>Number of root items</th>
<th>Number of children</th>
<th>Number of periods</th>
<th>CPU (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Low</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>10</td>
<td>299</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>414</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>741</td>
</tr>
<tr>
<td>20</td>
<td>Low</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>10</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>10</td>
<td>618</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>776</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1271</td>
</tr>
<tr>
<td>30</td>
<td>Low</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>10</td>
<td>973</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>2278</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>3317</td>
</tr>
</tbody>
</table>

Genetic algorithm is much faster than Cplex, without taking into consideration the number of solution found. Genetic algorithm gives solutions that are very close to optimal ones within very short computational time. Hence the efficiency of the genetic algorithm provides the decision maker a huge choice in terms of solution quality and in short time. Before presenting results we note that from 30 periods with medium number of children Cplex could not give solutions. In this section the table 3 summarize the computation time of the GA for all instances.

We observe that the computation time increases quickly as the number of the periods, on the other side it does not increase apparently as the numbers of root items increase.

6 CONCLUSIONS

In this paper, we addressed the multi-products capacitated disassembly scheduling with setup times and lost sales. To our knowledge, it the first time that disassembly scheduling problem with lost sales is investigated. We formulated a multi-objective optimization model, and propose a genetic algorithm NSGA-II for solving the problem. The objectives considered are (1) Minimizing the total cost and (2) Maximizing the service level. The performance of NSGA-II is investigated by comparing its results with those obtained by exact method on mono-objective sample (270 test problems) randomly generated (Kim et al., 2009- instances). This comparison shows that the NSGA-II give solution with good quality in reasonable time while Cplex software does not. This research can be extended in several ways. New mathematical formulation approaches can be developed considering multi level product structure and parts commonality constraints. Uncertainties such as stochastic demands or stochastic disassembly times have to be considered. The method can also be improved by using other dominance criterion to reduce the number of solution and be developing hybridization. Properties of the model should also be investigated to improve encoding of solutions.

REFERENCES


