University Paris-Saclay,
M2 – Optimization, Academic year 2015-2016
LRI – Laboratoire de Recherche en Informatique

REPORT OF INTERNSHIP

"STOCHASTIC PROGRAMMING APPROACHES FOR PLANNING RE-MANUFACTURING ACTIVITIES UNDER UNCERTAIN DEMAND AND RETURNS FORECASTS"

Author:
Quan Dong VU

Supervisors:
Dr. Céline Gicquel
Dr. Safia Kedad-Sidhoum

11 September 2016

Declaration

I, Quan Dong VU, declare that this report of internship titled, "Stochastic Programming Approaches for Planning Re-Manufacturing Activities under Uncertain Demand and Returns Forecasts" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for Master 2 degree - Optimisation program of University Paris Saclay, as the internship project done at LRI-Laboratoire de Recherche en Informatique, funded as a part of a one-year project supported by the Gaspard Monge program.
- Where any part of this report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this report is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

Acknowledgement

I would like to express my profound gratitude to my supervisors – Dr. Céline Gicquel and Dr. Safia Kedad-Sidhoum for their precious time, guidances and devoted support. Under their supervision, not only I did learn many precious knowledge and experiences, but also I received the encouragement and motivation to overcome challenges and difficulties encountered during the project. I also would like to give my thanks to the researchers and other intern-students in team ROCS, as well as all people and staffs at LRI-Laboratoire de Recherche en Informatique and Gaspard Monge program for their help throughout my internship.

I would like to dedicate my sincere appreciation toward all members of the Board of the M2-Optimization program, especially professor Filippo Santambrogio, for their help throughout this academic year. Special thanks are given to people in PGMO and FMJH for the financial aids and the enrolment opportunity.

Last but not least, I would like to thank my mother, my family and my friends who always support me through thick and thin.

Abstract

Re-manufacturing systems are defined as sets of processes transforming end-of-life products (returned products) into like-new products, once again usable by customers, mainly by rehabilitating damaged components. Within this project, we consider a re-manufacturing system involving three production echelons: (1) *disassembling* the used products brought back by customers, (2) *refurbishing* the recovered parts and (3) *reassembling* them into like-new finished products. We seek to plan the production activities on this system over a multi-period horizon. This leads to the formulation of a multi-echelon lot-sizing problem with lost sales, mathematically expressed as a Mixed-Integer Linear Programming. We first consider a deterministic variant of the problem and focus on its resolution through Cut-and-Branch and Branch-and-Cut algorithms. We then investigate a stochastic variant of the problem, in which the uncertainties on the quantity of returned products and the customers' demand are explicitly taken into account. We propose a multi-stage stochastic programming approach relying on scenario trees to represent the uncertain information structure. Extending the results obtained from the deterministic problem, we solve the stochastic problem by a Cut-and-Branch algorithm. Computational experiments are implemented and show that the proposed solution approach is efficient at solving medium-size instances of the problem

Keywords: *lot-sizing, re-manufacturing system, echelon stock reformulation,* (k, U) *valid inequalities class, polyhedral approach, Branch-and-Cut algorithm, Cut-and-Branch algorithm, multi-stage stochastic programming, scenario tree*

Contents

Chapter 0

Introduction

0.1 Re-manufacturing System

Traditionally, industrial production considers *forward supply chains*, which mainly deal with material flows going from the suppliers, up to the production activities transforming these raw materials into the finished products which are then distributed and sold to customers. However, nowadays, industrial companies around the world take increasing concern on the use of *reverse supply chains*, due to the general growth in environmental protection awareness and the pressure to reduce their pollutants emissions and natural resources consumption. One of the activities found in reverse supply chains is using *re-manufacturing*. It is defined as a set of processes transforming end-of-life products (used products/ returns) into like-new finished products, once again usable by customers, mainly by rehabilitating damaged components.

The recent development and constant growth in practical use of re-manufacturing systems can be explained by the following three mains reasons:

- Legislation: Over the last twenty years, a series of laws have been issued in Europe (and many other places) to regulate waste collection and management. For example, the WEEE Directive became European Union official law in 2003. It obliged electrical equipment manufacturers to take full responsibility for the entire life-cycle of their products, especially for collection, recovery and final disposal (European Standard EN 50419). Re-manufacturing comes into this setting as one of the possible alternatives for recovery.

- Economic values of the components embedded in the end-of-life products: From a economic point of view, re-manufacturing is a way of cutting down materials supply expenditure since it repairs and reuses recoverable components of the products. This contrasts to recycling systems which completely destroy the products and only seek to recover the basic raw materials.

- "Green image" of the company: As a friendly environmental choice, re-manufacturing helps a company reduce the environmental impact of its industrial products and preserve natural resources. It thus contributes in creating the "green image" of the company allowing marketing projects to improve customers' sensitivity.

In practice, re-manufacturing systems can be found in a variety of industrial sectors: car production, electrical goods, aircraft engines, computers, smart-phones, chemical products, etc. (see [17]). Re-manufacturing are most often complex systems encompassing a variety of activities such as used products collecting, testing and sorting, disassembling, refurbishing, reassembling and redistributing, etc.

In the present work, we consider a re-manufacturing system involving three production echelons: Disassembling, Refurbishing, and Reassembling. The system is illustrated in Figure 1 and will be explained with more details in Section 0.4. We take our focus on optimizing the production planning for such a system over a multi-period horizon. This implies solving a lot-sizing problem.

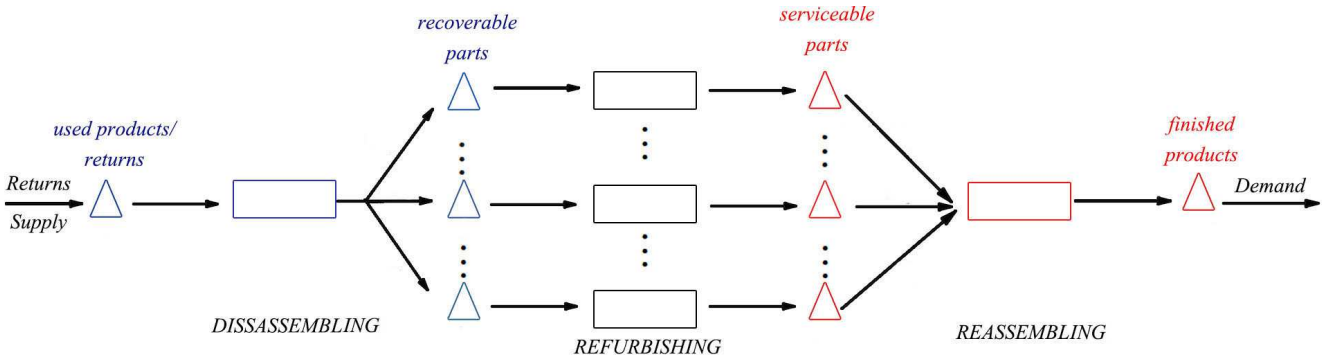

Figure 1: Configuration of the re-manufacturing system

0.2 Production Planning and Lot-sizing Problems

Industrial production planning requires making decisions on the products to be made, the timing and level of production (lot-size) as well as on the resources to be used. The main objective is to meet the customers' demand for the company products in the most cost-effective way.

Lot-sizing is defined by Kuik et al., (see [5]) as *"the clustering of items for transportation or manufacturing at the same time"*. Lot-sizing problems arise in production situations which involve set-up operations (tool changes, machine calibration, machine installation, etc.) incurring fixed costs. As a naive perception, to reduce the *set-up costs*, production should be run using large lot sizes. However, this generates de-synchronized patterns between the customers' demand and the production plan leading to costly high levels of inventory. Lot-sizing models thus aim at reaching the best possible trade-off between minimizing set-up costs and minimizing inventory holding costs, under constraints on customer demand satisfaction and technical limitations of the system.

Moreover, to simulate real-life situations, lost-sales are sometimes allowed in lot-sizing models. It corresponds to the situation where a company does not obligatorily satisfy all customer demand and accepts to pay a high penalty cost on each unit of lost-sales product.

Lot-sizing problems can be mathematically formulated as Mixed-Integer Linear Programming (MILP) problems (please see Section 1.1 for more details). A small illustrative example on lot-sizing modelled as MILP with trading-off between set-up and inventory holding costs is provided in Appendix A. The question of how to apply and extend lot-sizing models to plan production in

re-manufacturing systems has emerged as a new research field over the last 10 years. By using mathematical programming tools, this work aims at contributing to this new research area.

0.3 Polyhedral Approaches for Mixed-Integer Linear Programming

The Branch-and-Bound algorithm is one of the most widely-used algorithms to solve MILP. It carries out an implicit enumeration of all candidate solutions. This enumeration relies heavily on a search tree (Branch-and-Bound tree); furthermore, a lower bound of the optimal solution value is computed (at each node of the tree) and exploited to discard uninteresting subsets of feasible solutions. In practice, the Branch-and-Bound algorithm performance intensely depends on the quality of this lower bound. In the worst cases, the number of nodes in the Branch-and-Bound tree may be an exponential term of the number of integer variables, leading to long computation time. In the present work, we focus on a variant of the Branch-and-Bound algorithm where the lower bounds are obtained from solving the Linear Relaxation of the problem. This Linear Relaxation is the problem acquired when we remove all integrality constraints from the MILP formulation.

In order to reduce the computation time of the Branch-and-Bound algorithm, we need to improve the lower bounds. This can be done through the use of valid inequalities, which will improve the description of the feasible set of the problem, and thus strengthen the lower bound provided by its Linear Relaxation. Such solution approaches are called polyhedral approaches.

0.3.1 Valid Inequalities and Separation Algorithm

We first notice that, given a MILP and its feasible set X, by considering the formulation as the convex hull of X (called tight formulation), simply solving it as a pure linear programming problem provides us a solution of the MILP. However, finding the exact description of $conv(X)$ is as least as hard as solving the MILP; moreover, most of the time, the description of $conv(X)$ contains an amount of inequalities which is exponential in the number of variables (see [19]).

Therefore we can only hope to attain some "partial reformulation" where the tightened formulation is "closer to" (but not necessary is) $conv(X)$ in order to improve the lower bounds quality and to reduce the number of nodes in the Branch-and-Bound tree. This purpose can be accomplished by adding into the initial formulation of the problem some *Valid Inequalities, simply defined as inequalities which are satisfied by all points of the feasible set X*. Of course, we only need those valid inequalities which are effective, i.e. indeed reduce the feasible set of its Linear Relaxation closer to $conv(X)$ (see [19]). In the literature, there are various finite sets or "families" of valid inequalities, which we call *classes of valid inequalities*.

Notice that, adding a class C of valid inequalities to the MILP formulation and solving its Linear Relaxation, we only need a limited number of these inequalities (ideally, not exceeding the number of variables). As a consequence, if we add all valid inequalities of a class $\mathcal{C}$, usually, there would be many useless inactive valid inequalities. This would increase the size of the linear programming problems to be solved and thus, slow down the resolution.

To avoid this case, we should solve the Linear Relaxation beforehand (without adding any valid

inequalities); and then only iteratively add those valid inequalities from $\mathcal{C}$ which are needed. Thus, the next challenge encountered is to seek for an algorithm determining whether a valid inequality is needed, i.e. whether the solution of the considering Linear Relaxation is satisfied by that inequality. We call such an algorithm a *separation algorithm*. In the book by *Pochet and Wolsey [19]* on lot-sizing problems, a separation algorithm is defined as follows: given the solution $(x^*, y^*) \in \mathbb{R}^n \times \mathbb{Z}^m$ of the Linear Relaxation and a class $\mathcal{C}$ of valid inequalities for the corresponding MILP; after running algorithm, either we prove that (x^*, y^*) satisfies all valid inequalities in $\mathcal{C}$, or we find a valid inequality in $\mathcal{C}$ that is violated at (x^*, y^*).

0.3.2 Branch-and-Cut and Cut-and-Branch Algorithm

The *Branch-and-Cut algorithm* basically is a combination of the Branch-and-Bound algorithm and the above-mentioned idea of utilizing the valid inequalities. Given a finite class of valid inequalities, named $\mathcal{C}$; at each node of the Branch-and-Bound tree:

- Solve the current Linear Relaxation (already added branching constraints), call its solution (X^*, Y^*).
- While there is at least one valid inequality from $\mathcal{C}$ found, run the *Separation Algorithm* on (X^*, Y^*) to generate the new valid inequalities needed from $\mathcal{C}$ (violated by (X^*, Y^*)), add them to the formulation of the Linear Relaxation, resolve it, update new solution to (X^*, Y^*) (this is called the *Cutting Plane algorithm*, and the valid inequalities added would sometime be called *"cuts"*).
- We find the optimal solution of the strengthened reformulation Linear Relaxation at this node.
- Pruning and branching are done in the same way as in the Branch-and-Bound algorithm.

The *Cut-and-Branch algorithm* is a variant of Branch-and-Cut algorithm where the Cutting plane algorithm is only called at the root node of Branch-and-Bound tree. In practice, the Cut-and-Branch algorithm saves us the time in processing the nodes in the tree, but our formulation is weaker than the one obtained from applying the Branch-and-Cut algorithm.

0.4 Main Objective of The Project

Over a discrete time horizon, we seek to build an optimal production plan (in term of minimizing the total cost) for a re-manufacturing system involving three main production echelons (see Figure 1):

- *Echelon 1: Disassembling*: starting from the used products (returns) brought back by customers, we disassemble them into a series of components/parts (called recoverable parts)
- *Echelon 2: Refurbishing*: throughout rehabilitation processes, we refurbish the recoverable parts to provide the well-functioning serviceable parts
- *Echelon 3: Reassembling*: taking the serviceable parts, we reassemble them into the like-new finished products which are then sold to customers.

The main challenge in this context, which is also one of the special features of re-manufacturing systems, is the high level of uncertainty in the input data such as returns supply, market demand forecasts, and cost parameters, arising from the lack of control by companies on the quantity, quality and timing of the used products returned by customers. Even in cases when companies apply the special policies to collect the used products from customers, say, products life-cycle contracts, collecting agencies, etc., these parameters would still not be accurately predictable. Within this work, we model the quantity of collected returns and demand of customers as discrete random variables, unfolding little by little as time progresses.

The main objectives of this internship project is thus to:

(1) propose a stochastic programming based approach capable of explicitly handling the uncertainty in the input data needed for planning production on the re-manufacturing system,

(2) develop a solution algorithm based on polyhedral approaches in order to solve the resulting MILP to optimality, at least for medium-size instances.

0.5 Outline of the Report

To achieve our objectives, we first present a literature review in *Chapter 1*, where we give an overview on the current state of the art and highlight the contributions and significances of this report. We provide a general research background on lot-sizing problems (*Section 1.1*), and then focus on lot-sizing for re-manufacturing systems (*Section 1.2*). Lastly, we mention some noteworthy works related to stochastic lot-sizing problems in *Section 1.3*.

Chapter 2 is devoted to the deterministic version of the considered problem, where we temporarily assume that all input parameters are determined beforehand. Modelling it as a MILP problem (*Section 2.1*), we use the echelon stock concept to come up with a new formulation enabling us to decompose it into a series of sub-problems linked to each other by some coupling constraints (*Section 2.2*). Next, we suggest a Cut & Branch algorithm based on the use of the specific class (k,U) valid inequalities (*Section 2.3*). In *Section 2.4*, we describe the computational experiments carried out to evaluate the performance of our designed algorithm and then discuss the obtained results.

Chapter 3 is devoted to research on the lot-sizing problem for our re-manufacturing system under stochastic forecasts on customers' demand and quantity of returns, named, *stochastic problem*. *Section 3.1* introduces the use of a *scenario tree* for representing the stochastic information structure, which allows us to model the stochastic problem as a MILP problem (*Section 3.2*).The obtained MILP has a structure similar to the deterministic problem investigated in Chapter 2. We thus seek to extend the techniques applied in Chapter 2 in order to solve the stochastic problem. We reformulate it using echelon inventory concept (*Section 3.3*). We then try to modify and apply the (k,U) valid inequalities for strengthening the linear relaxation, and then solve it by Cut-and-Branch algorithm (and Branch-and-Cut algorithm) (*Section 3.4*). We implement some computational experiments and then present the results in *Section 3.5*. Finally, we attempt to construct new (and theoretically stronger) valid inequalities, based on the procedure named *pairing procedure* (*Section 3.6*).

Chapter 1

Literature Review

In this chapter, we present an overview of the literature related to our considered problem. Our aim is to clarify the contributions and significances of this report in comparison with works and research done in the literature. Firstly, *Section 1.1* is devoted to a brief summary of some noteworthy results on lot-sizing problems and their application. *Section 1.2* proposes a review of recent papers on lot-sizing for re-manufacturing systems. Finally, *Section 1.3* mentions research related to stochastic lot-sizing problems, with a focus on stochastic programming with scenario tree.

1.1 Lot-sizing Problems

Considering a simple single-item single-echelon uncapacitated system, in 1958, Wagner and Whitin proposed in their classic paper ([29]) a dynamic lot-sizing model where the basic trade-off between set-up and inventory costs was first thoughtfully considered. As a review for various papers arisen from their model since then, the paper [3] offers an overview on modelling deterministic single-echelon dynamic lot sizing problems. Due to the fact that most lot-sizing problems are hard to solve (in term of complexity), they have been studied with a variety of solution approaches and techniques, including dynamic programming, Dantzig–Wolfe decomposition, Lagrange relaxation, cutting planes, heuristics and meta-heuristic, etc. For comparison and general guidelines on these techniques, the readers are referred to [4].

As explained in Chapter 0, our re-manufacturing system involves three production echelons, thus, we take interest in multi-echelon lot-sizing problems. A method for solving a multi-echelon lot-sizing problem can be found in [30]; however, the authors focus on capacitated-resources cases and they allow back-orders which differs from our considered problem. To solve an uncapacitated multi-echelon lot-sizing problem, M. Zhang et al. present the use of valid inequalities for alternative formulations (see [32]), but they assume a simple in-series-production structure whereas we have a more general disassembly-reassembly product structure.

Remarking that we also allow lost-sales in our model, we consult some papers in the literature, including, *M. Loparic et al. [16]* on the uncapacitated problem with sales in each period (equals lost-sales subtracted from demand); this is also one of the key-authors for our research as it introduces the specific valid inequalities class named (k, U) that we will use in this project. Lost-sales concept is

also considered in [1] and [12], simultaneously with some other assumptions.

Finally, the source for preliminaries and basic concepts regarding polyhedral approaches for lot-sizing, especially Branch-and-Cut algorithm (and its variant as Cut-and-Branch algorithm) is the book by *Pochet and Wolsey ([19])* on lot-sizing problems and MILP.

1.2 Lot-sizing for Planning Re-manufacturing Activities

The use of lot-sizing in re-manufacturing system is relatively new in the literature. Research done in this field can be categorized according to three main types of re-manufacturing systems:

- Single-echelon re-manufacturing system (often coupled with a manufacturing system): An example is illustrated by Figure 1.1 a. This hybrid use of re-manufacturing and manufacturing systems can be composed in different variants: either customers' demand on re-manufactured products and newly manufactured products can be mutually substituted (see [27]), or either in one-way substitution case (see [21]). Some solution approaches can be found in [22] and [11].

- Multi-echelon disassembly system: disassembling processes can be considered as an alternative components supplying source (see [14]) or as a standing alone operation (see [24]). An example is illustrated in Figure 1.1 b.

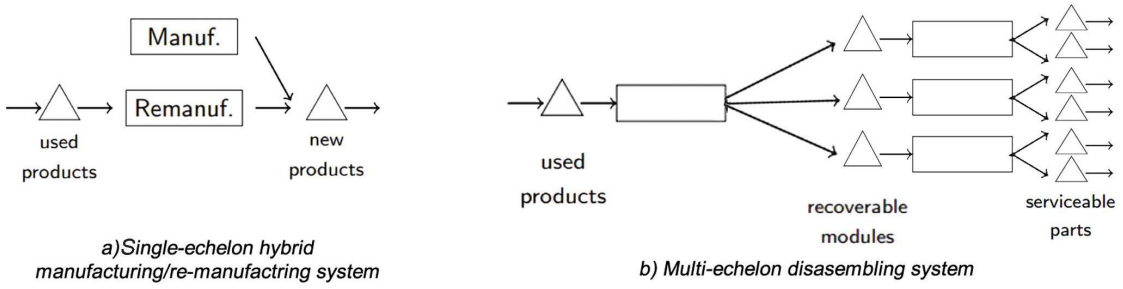

Figure 1.1: Illustration of some re-manufacturing systems

- Multi-echelon disassembly-reassembly system: The subsystems of re-manufacturing system are explicitly expressed reflecting the typical characteristics; an example of using this framework is [13]. *Our lot-sizing for re-manufacturing system can be categorized into this type*, the Figure 1 (Section 0.1) can be considered as an illustration. The most similar model we found in the literature is the paper by *Ahn et al. [2]*, which is also another key-author of this report. The authors also consider the re-manufacturing system with three echelons. However, they focus on a deterministic problem, and assume an unlimited amount of returns in each period whereas we consider a stochastic problem and take into account the limited availability of returned products. Moreover, they proposes a heuristic solution approach whereas we seek to design an exact method providing a guaranteed optimal solution.

In short, this report studies on the **Uncapacitated Multi-Item Multi-Echelon Lot-sizing Problem for Re-manufacturing System** (involving three echelons: Disassembling, Refurbishing

and Reassembling) **with Lost-Sales and Uncertain Input Data.** As far as we are concerned, there has not been any work in the literature which considers the same model and context. On the complexity of the problem, since it is proven in [2] that their problem is NP-hard, and their model could be considered as a particular variant of our model (given that the returns amount is unlimited and lost-sales cost is infinite), we can deduce that **our considered problem is also a NP-hard problem.**

1.3 Stochastic Programming Approaches for Lot-sizing Problems

Investigating the lot-sizing problem under stochastic factors is an active research area in the production planning field. We can classify the related papers according to their modelling of the decision process.

Single-stage models seek to build a production plan for the whole horizon beforehand, and do not consider the possibility to adjust it, once new information becomes available. Most commonly found option is to consider that demand is stochastic and non-stationary, e.g. [28] on stochastic version of Wagner–Whitin model, [18] with their heuristics and [26] combining with a fill-rate constraint. On the other hand, the paper [15] studies chance-constrained programs with discrete distributions.

In contrast, multi-stage models allow us to postpone some of the decisions to a later point in time, when new information on the uncertain parameters becomes available. Such an approach has been considered in a series of papers by Y. Guan et al. ([6], [7], and one of our key-papers [8]), where polyhedral approaches to solve stochastic knapsack and stochastic lot-sizing problems are discussed. The authors propose a general method for cutting plane generation and use a Branch-and-Cut algorithm based on investigating cuts for each scenario. However, their works are limited to single-echelon, single-item (non-related to re-manufacturing) and rather simple problems. For our work, we try to modify and apply their idea on our much more complicated multi-echelon multi-item lot-sizing problem. Y. Guan et al. also utilize a scenario tree for expressing the information structure and focus on a specific class of valid inequalities namely (Q, S_Q) generalized from the classic class (l, S) (this approach can also be found in [23] by M. Di Summa and L. A. Wolsey). In contrast, our work extends the ideas with (k, U) class of valid inequalities. Furthermore, the pairing operation and tree inequality concept suggested in [8] is another idea to generate stronger valid inequalities, which is one of our perspectives for future works.

Chapter 2

Deterministic Problem

In this chapter, we assume that all input parameters needed for production planning decisions are perfectly known at the time the production plan is built. This leads to a deterministic combinatorial optimization problem, which we call the Deterministic Problem. The *Section 2.1* provides a Mixed-Integer Linear Programming formulation of this problem. We can use natural intuition to explain this model, and refer it to the term "Natural Formulation", denoted (NF). The structure of its constraints matrix makes it quite inappropriate for the use of polyhedral approaches. Moreover, our computational results suggest that CPLEX's Branch-and-Bound algorithm does not perform well when using this formulation. Therefore, we introduce an alternative "Echelon Stock Reformulation" abbreviated as (EF), which is presented in *Section 2.2*. This second formulation allows us to change the perspective, and see the deterministic problem as a series of single-echelon lot-sizing sub-problems, linked together by inter-echelon coupling constraints. In *Section 2.3*, we define and analyse each of these sub-problems, introduce the valid inequalities in the class (k,U) and discuss the separation algorithm. Using these results, we design a Cut-and-Branch algorithm for solving the problem under the formulation (EF). Finally, in *Section 2.4*,the computational experiments and results are presented and discussed.

2.1 Deterministic MILP Natural Formulation

2.1.1 Problem Statement

We first describe our modelling assumptions and introduce the notation for parameters of the problem:

- The re-manufacturing process is considered over a planning horizon with T *discrete time periods*. Denote $\mathbf{t} = \mathbf{1}, ..., \mathbf{T}$ the set of time periods ($T > 0$).
- It involves three main echelons: (1) *Disassembling* the used product into a series of recoverable parts, (2) *Refurbishing* the recoverable parts into serviceable parts on dedicated production processes, (3) *Reassembling* the serviceable parts into re-manufactured products. We assume that each product consists of $\mathbf{I}$ small parts/components ($I > 0$). Thus, in summary, the considered problem concerns $2I+2$ *"items"*. For the sake of notation, we use $\mathbf{i} = \mathbf{0}$ to denote the *Used Products*; $\mathbf{i} = \mathbf{1}, ..., \mathbf{I}$ as the set of *Recoverable Parts* obtained from disassembling the used products;

$\mathbf{i = I + 1, ..., 2I}$ as the set of *Serviceable Parts* obtained from refurbishing the recoverable parts; and finally, denote $\mathbf{i = 2I + 1}$ the *Finished Products* obtained from reassembling the serviceable parts.

- It involves $I + 2$ operation processes, including the *Disassembling Process* $\mathbf{p = 0}$; *Refurbishing Processes* $\mathbf{p = 1, ..., I}$ corresponding to refurbishing item p into item $I + p$; the *Reassembling Process* $\mathbf{p = I + 1}$. The capacities of these operation processes are assumed to be unlimited, i.e. do not depend on the availability of the machines or human resources.

- The *Quantity of Returns (Used Products)* that the company collects before each period t is known before the actual beginning of the whole re-manufacturing planning, and is denoted $\mathbf{r_t}$.

- The *Demand for Finished Products* of the customers in each period t is also determined beforehand, and is denoted $\mathbf{d_t}$.

- We have on hand a precise description of the *Bill of Material*. We denote $\boldsymbol{\alpha_i}$ the number of each part $i = 1, ..., I$ that one unit of used product/ finished product contains. For example, if a laptop consists of 2 hard disks and 1 battery; if we let $i = 10$ stands for hard disk and $i = 11$ for the battery, then $\alpha_{10} = 2, \alpha_{11} = 1$. We also assume the identity of this bill of material between disassembling and reassembling processes. For the sake of notation, $\alpha_0 = \alpha_{I+1} = 1$.

- The *Set-up Costs* for the processes are assumed to be deterministic, but might be time-dependent (i.e. vary over periods). $\mathbf{c_{p,t}}$ denotes the fixed cost to be paid in order to operate process p at period t. This cost does not depend on the quantity (the lot-size) of items processed by p in t.

- The *Unit Inventory Holding Costs* for item $i = 0, ..., 2I + 1$ in period t are determined and we denote them $\mathbf{h_{i,t}}$.

- The company has the option to lose sales, i.e. not to satisfy all demand of customers at the end of some periods. However, this is penalized in the objective function through a high penalty cost. We denote the *Unit Penalty Cost* $\mathbf{l_t}$ for each unit of demand lost in period t.

These parameters are represented by the following Figure 2.1

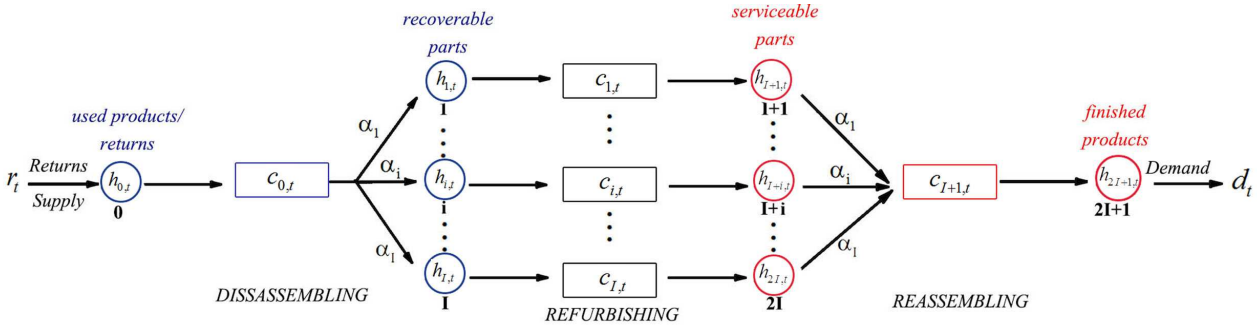

Figure 2.1: Parameters description of the re-manufacturing system

2.1.2 Decision Variables

To ultimately minimize the production planning cost, for each period $t = 1, ..., T$, the company needs to answer the series of questions: Should we set-up the process p? If yes, then how many corresponding items should we operate on that process? How many items should we store at the end of period t? How many products are accepted to be lost-sales?

Accordingly, we define the variables:

- $X_{0,t}$: quantity of used products disassembled in period t,
- $X_{p,t}$: quantity of parts p processed by the refurbishing process $p = 1, ..., I$ in period t,
- $X_{I+1,t}$: quantity of finished products reassembled in period t,
- $Y_{0,t} \in \{0, 1\}$: set-up variable for the disassembling process in period t,
- $Y_{p,t} \in \{0, 1\}$: set-up variable for the refurbishing process $p = 1, ..., I$ in period t,
- $Y_{I+1,t} \in \{0, 1\}$: set-up variable for the reassembling process in period t,
- $I_{i,t}$: inventory level of item $i = 0, ..., 2I + 1$ at the end of period t,
- L_t: quantity of lost-sales of finished products in period t.

2.1.3 Mixed Integer Linear Programming Natural Formulation

The first and most "natural" formulation of the problem is presented as follows:

$$(\mathbf{NF}) \qquad Z^* = min \sum_{t=1}^{T} \Big[\sum_{p=0}^{I+1} c_{p,t} Y_{p,t} + \sum_{i=0}^{2I+1} h_{i,t} I_{i,t} + l_t L_t \Big] \tag{2.1}$$

$$X_{p,t} \leq \min \left\{ \sum_{\tau=1}^{t} \alpha_p r_\tau, \sum_{\tau=t}^{T} \alpha_p d_\tau \right\} Y_{p,t} \qquad \forall p = 0, ..., I+1, \forall t \tag{2.2}$$

$$I_{0,t} = I_{0,t-1} + r_t - X_{0,t} \qquad \forall t \tag{2.3}$$

$$I_{i,t} = I_{i,t-1} + \alpha_i X_{0,t} - X_{i,t} \qquad \forall i = 1, ..., I, \forall t \tag{2.4}$$

$$I_{i+I,t} = I_{i+I,t-1} + X_{i,t} - \alpha_i X_{I+1,t} \qquad \forall i = 1, ..., I, \forall t \tag{2.5}$$

$$I_{2I+1,t} = I_{2I+1,t-1} + X_{I+1,t} - d_t + L_t \qquad \forall t \tag{2.6}$$

$$I_{i,0} = 0 \qquad \forall i = 0, ..., 2I+1 \tag{2.7}$$

$$I_{i,t} \geq 0 \qquad \forall i = 0, ..., 2I+1, \forall t \tag{2.8}$$

$$L_t \geq 0 \qquad \forall t \tag{2.9}$$

$$X_{p,t} \geq 0, Y_{p,t} \in \{0, 1\} \qquad \forall p = 0, ..., I+1, \forall t \tag{2.10}$$

The natural formulation model (NF) can be explained briefly as follows: The objective function (2.1) aims at minimizing the total re-manufacturing cost summing up the set-up costs, the inventory holding costs and the lost-sales penalty for non-satisfaction of the customer demand on every period.

Constraints (2.2) are Big-M type constraints which ensure that production quantity variables $X_{p,t}$ are strictly positive only when binary decision variables $Y_{p,t}$ equals 1, i.e. production can only take place if the corresponding process has been set up. As an explanation for the choice of coefficient, in each period t, for process p operating on item i, we cannot produce more than the maximum quantity of item i available obtained from collected returns up to period t, and do not need to produce more than what is needed to satisfy all possible demands in the future; thus the upper bound coefficient coincides with minimum of the two terms (note that $\alpha_0 = \alpha_{I+1} = 1$).

Constraints (2.3)-(2.6) are inventory balance constraints: they guarantee that the inventory level of item i at the end of period t equals the sum of the inventory level at the end period $t-1$ and the incoming flows (transformed into i from other items) minus the outflows spending on transforming i into successive items (to satisfy the demand).

2.2 Echelon Stock Reformulation

As mentioned in the introduction on the Branch-and-Bound algorithm and the polyhedral approaches for MILP resolution, by the use of valid inequalities, we can improve the quality of the lower bounds, reduce the number of nodes in Branch-and-Bound tree, and ultimately cut down the algorithm running time. However, before actually working with the cuts/valid inequalities, we notice that constraints (2.3)-(2.6) involve terms corresponding to a dependent demand, i.e. a demand coming from our production decisions rather than from external customers. These terms create a strong coupling between these constraints and make the structure of the problem under formulation (NF) more complex (for example, variable $X_{0,t}$ both appears in (2.3) and (2.4), etc.).

To circumvent this difficulty, one method is the concept of Echelon Stock. In fact, this allows us to see our deterministic problem as a series of single-echelon single-item uncapacitated lot-sizing sub-problems linked by some inter-echelon coupling constraints. Each of these sub-problems can be strengthened by the (k, U) valid inequalities. This enables us to develop a Branch-and-Cut (and Cut-and-Branch) algorithm for solving the problem on a new reformulation. To explain the reformulation model in this section, we first define the concept of Echelon Stock.

2.2.1 Echelon Demand and Echelon Stock

In period t, for each item $i = 1, ..., I, ..., 2I$ (the recoverable and serviceable parts), the *Echelon Demand* $ed_{i,t}$ can be understood as a translation of the external demand d_t for the finished product $2I + 1$ into the demand for item i. In other words, to satisfy the demand for d_t finished products, we need $\alpha_i d_t$ recoverable parts i and $\alpha_i d_t$ serviceable parts $I + i$; explicitly, $ed_{i,t} = ed_{I+i,t} = \alpha_i d_t, \forall i = 1, ..., I$. Moreover, each unit of used product can be disassembled into fixed numbers of recoverable parts, which in turn correspond to the quantities needed for reassembling one unit of finished product. In other words, to satisfy the demand d_t of finished products, we only need to disassemble the same amount d_t of used products, i.e. $ed_{0,t} = d_t$.

Based on this definition of echelon demand, we would like to present the concept of **Echelon Stock** of each item in our multi-echelon system, which corresponds to the quantity of item held in

inventory in the whole system, either as itself or as a component in its successors in the bill of material. Explicitly, if we call $\mathbf{EI_{i,t}}$ as the variables corresponding to the Echelon Stock of item i at period t, we have:

- For finished product $2I+1$: $EI_{2I+1,t} = I_{2I+1,t}, \forall t$.
- For serviceable parts $i = I+1, ..., 2I$: $EI_{i,t} = I_{i,t} + \alpha_{i-I} EI_{2I+1,t} = I_{i,t} + \alpha_{i-I} I_{2I+1,t}, \forall t$.
- For recoverable parts $i = 1, ..., I$: $EI_{i,t} = I_{i,t} + EI_{I+i,t} = I_{i,t} + I_{I+i,t} + \alpha_i I_{2I+1,t}, \forall t$.
- For the used product 0: $EI_{0,t} = I_{0,t} + \frac{EI_{1,t}}{\alpha_1} = I_{0,t} + \frac{I_{1,t}}{\alpha_1} + \frac{I_{I+1,t}}{\alpha_1} + I_{2I+1,t}, \forall t$. Remark that, similar to the explanation of used products echelon demand, here in this definition of $EI_{0,t}$, the choice of using part 1 is arbitrary and that any other recoverable part $i = 2, ..., I$ might have been used in the definition of $EI_{0,t}$ without any consequence on the model.

2.2.2 MILP Echelon Stock Reformulation

Firstly, we would like to re-write the summation term relating to inventory variables in the objective function:

$$\begin{aligned}
\sum_{i=0}^{2I+1} h_{i,t} I_{i,t} &= h_{0,t} I_{0,t} + \sum_{i=1}^{I} h_{i,t} I_{i,t} + \sum_{i=1}^{I} h_{i+I,t} I_{i+I,t} + h_{2I+1,t} I_{2I+1,t} \\
&= h_{0,t} \left(EI_{0,t} - \frac{EI_{1,t}}{\alpha_1} \right) + \sum_{i=1}^{I} h_{i,t} \left(EI_{i,t} - EI_{i+I,t} \right) \\
&\qquad + \sum_{i=1}^{I} h_{i+I,t} \left(EI_{i+I,t} - \alpha_i EI_{2I+1,t} \right) + h_{2I+1,t} EI_{2I+1,t} \\
&= \underbrace{h_{0,t}}_{eh_{0,t}} EI_{0,t} + \underbrace{\left(h_{1,t} - \frac{h_{0,t}}{\alpha_1} \right)}_{eh_{1,t}} EI_{1,t} + \sum_{i=2}^{I} \underbrace{h_{i,t}}_{eh_{i,t}, i=2,...,I} EI_{i,t} + \sum_{i=I+1}^{2I} \underbrace{\left(h_{i,t} - h_{i-I,t} \right)}_{eh_{i,t}, i=I+1,...,2I} EI_{i,t} \\
&\qquad + \underbrace{\left(h_{2I+1,t} - \sum_{i=1}^{I} \alpha_i h_{I+i,t} \right)}_{eh_{2I+1,t}} EI_{2I+1,t} \\
&= \sum_{i=0}^{2I+1} eh_{i,t} EI_{i,t}
\end{aligned}$$

To simplify the notation for our new reformulation, here, we have defined the *Unit Echelon Stock Holding Costs* $\mathbf{eh_{i,t}}$ such that: $eh_{0,t} = h_{0,t}$; $eh_{1,t} = h_{1,t} - \frac{h_{0,t}}{\alpha_1}$; $eh_{i,t} = h_{i,t}$, $\forall i = 2, ..., I$; $eh_{i,t} = h_{i,t} - h_{i-I,t}$, $\forall i = I+1, ..., 2I$; $eh_{2I+1,t} = h_{2I+1,t} - \sum_{i=1}^{I} \alpha_i h_{I+i,t}$.

Using this notation and simply substituting inventory variables $I_{i,t}$ by Echelon Stock variables $EI_{i,t}$ in the objective function and the constraints, we obtain the new MILP Echelon Stock Reformulation (EF):

$$(\mathbf{EF}) \qquad Z^* = min \sum_{t=1}^{T} \Big[\sum_{p=0}^{I+1} c_{p,t} Y_{p,t} + \sum_{i=0}^{2I+1} eh_{i,t} EI_{i,t} + l_t L_t \Big] \tag{2.11}$$

$$X_{p,t} \leq \min\left\{\sum_{\tau=1}^{t} \alpha_p r_\tau, \sum_{\tau=t}^{T} \alpha_p d_\tau\right\} Y_{p,t} \qquad \forall p = 0, ..., I+1, \forall t \qquad (2.12)$$

$$EI_{0,t} = EI_{0,t-1} + r_t - d_t + L_t \qquad \forall t \qquad (2.13)$$

$$EI_{i,t} = EI_{i,t-1} + \alpha_i X_{0,t} - \alpha_i d_t + \alpha_i L_t \qquad \forall i = 1, ..., I, \forall t \qquad (2.14)$$

$$EI_{i+I,t} = EI_{i+I,t-1} + X_{i,t} - \alpha_i d_t + \alpha_i L_t \qquad \forall i = 1, ..., I, \forall t \qquad (2.15)$$

$$EI_{2I+1,t} = EI_{2I+1,t-1} + X_{I+1,t} - d_t + L_t \qquad \forall t \qquad (2.16)$$

$$EI_{i,0} = 0 \qquad \forall i = 0, ..., 2I+1 \qquad (2.17)$$

$$EI_{0,t} - \frac{EI_{1,t}}{\alpha_1} \geq 0 \qquad \forall t \qquad (2.18)$$

$$EI_{i,t} - EI_{I+i,t} \geq 0 \qquad \forall i = 1, ..., I, \forall t \qquad (2.19)$$

$$EI_{I+i,t} - \alpha_i EI_{2I+1,t} \geq 0 \qquad \forall i = 1, ..., I, \forall t \qquad (2.20)$$

$$EI_{2I+1,t} \geq 0 \qquad \forall t \qquad (2.21)$$

$$L_t \geq 0 \qquad \forall t \qquad (2.22)$$

$$EI_{i,t} \geq 0 \qquad \forall i = 0, ..., 2I, \forall t \qquad (2.23)$$

$$X_{p,t} \geq 0, Y_{p,t} \in \{0, 1\} \qquad \forall p = 0, ..., I+1, \forall t \qquad (2.24)$$

The constraints (2.18)-(2.21) ensure that the natural inventory variables $I_{i,t}$ are positive, and constraint (2.17) comes from the constraint (2.7) in model (NF). More importantly, notice that in this new formulation (EF), constraints (2.13)-(2.16) do not contain dependent demand terms. This thus allows us to decompose the Deterministic problem into a series of sub-problems and to deal with them separately.

2.3 Sub-problems and Branch-and-Cut Algorithm

In the previous section, we have built up the new formulation (EF) for our deterministic problem. This formulation can be seen as several single-echelon lot-sizing sub-problems linked together by the group of constraints (2.18)-(2.21). These sub-problems are relaxed from (EF); therefore, valid inequalities strengthening each of these sub-problems would also be valid inequalities strengthening (EF).

2.3.1 Refurbishing and Reassembling Processes Sub-problem

For each of the refurbishing processes $p = 1, ..., I$ as well as for the reassembling process $p = I+1$, we consider the sub-problem of (EF) corresponding to p:

$$\textbf{(Sub(p))} \qquad Z_p^* = min \sum_{t=1}^{T} (c_{p,t} Y_{p,t} + eh_{I+p,t} EI_{I+p,t} + l_t L_t) \qquad (2.25)$$

$$X_{p,t} \leq \min\left\{\sum_{\tau=1}^{t} \alpha_p r_\tau, \sum_{\tau=t}^{T} \alpha_p d_\tau\right\} Y_{p,t} \qquad \forall t \qquad (2.26)$$

$$EI_{I+p,t} = EI_{I+p,t-1} + X_{p,t} - \alpha_p (d_t - L_t) \qquad \forall t \tag{2.27}$$

$$EI_{I+p,t} \geq 0 \qquad \forall t \tag{2.28}$$

$$L_t \geq 0, X_{I+p,t} \geq 0, Y_{I+p,t} \in \{0,1\} \qquad \forall t \tag{2.29}$$

The class of Valid Inequalities called (k,U) inequalities was mentioned in Chapter 1. It is used in [16] to strengthen the Linear Relaxation of an uncapacitated single-echelon single-product lot-sizing problem with lost sales.

Applying this to the sub-problem (Sub(p)) $\forall p = 1, ..., I+1$, we have the description of (k, U) valid inequalities used to strengthen sub-problem (Sub(p)):

$$EI_{I+p,k} \geq \alpha_p \sum_{\tau \in U} \left[d_\tau (1 - \sum_{\theta=k+1}^{\tau} Y_{p,\theta}) - L_\tau \right] \tag{2.30}$$

for any $0 \leq k \leq T-1$ and $U \subset \{k+1, ..., T\}$

Remark 2.3.1. *Given $p = 1, ..., I+1$, for any such k and subset U, we can easily prove that any inequalities of the form (2.30) is indeed "valid" for the problem (Sub(p)), i.e. any feasible solution of (Sub(p)) satisfies (2.30).*

Sketch of Proof

Given that $(EI_{I+p,t}, X_{p,t}, L_t)_t$ satisfies (2.27), we have

$$EI_{I+p,k} = \sum_{\tau=1}^{k} X_{p,\tau} - \sum_{\tau=1}^{k} \alpha_p (d_\tau - L_\tau) \tag{2.31}$$

Denote $t = \max\{\tau : \tau \in U\}$, we can easily rewrite the inequality (2.30) under the form:

$$\sum_{\tau=1}^{k} X_{p,\tau} + \alpha_p \sum_{\theta=k+1}^{t} Y_{p,\theta} \left(\sum_{\substack{\theta \leq \tau \leq t \\ \tau \in U}} d_\tau \right) \geq \alpha_p \sum_{\tau=1}^{k} (d_\tau - L_\tau) + \alpha_p \sum_{\tau \in U} (d_\tau - L_\tau) \tag{2.32}$$

* Case 1: Suppose that $Y_{p,\tau} = 0, \forall \tau = k+1, ..., t$, then since $X_{p,\tau} = 0, \forall \tau = k+1, ..., t$ we have:

$$\begin{aligned} &\sum_{\tau=1}^{k} X_{p,\tau} + \sum_{\theta=k+1}^{t} Y_{p,\theta} \left(\sum_{\theta \leq \tau \leq t, \tau \in U} d_\tau \right) = \sum_{\tau=1}^{t} X_{p,\tau} \\ &\overset{(2.31)}{\geq} \alpha_p \sum_{\tau=1}^{k} (d_\tau - L_\tau) + \alpha_p \sum_{\tau=k+1}^{t} (d_\tau - L_\tau) \geq \alpha_p \sum_{\tau=1}^{k} (d_\tau - L_\tau) + \alpha_p \sum_{\tau \in U} (d_\tau - L_\tau) \end{aligned}$$

* Case 2: Otherwise, let $\delta = \min\{\tau \in \{k+1, ..., t\} : Y_{p,\tau} = 1\}$, similar to case 1, deduce that:

$$\sum_{\tau=1}^{k} X_{p,\tau} \geq \alpha_p \sum_{\tau=1}^{k} (d_\tau - L_\tau) + \alpha_p \sum_{k+1 \leq \tau \leq \delta, \tau \in U} (d_\tau - L_\tau)$$

Moreover, $\alpha_p \sum_{\theta=k+1}^{t} Y_{p,\theta} \left(\sum_{\theta \leq \tau \leq t, \tau \in U} d_\tau \right) \geq \alpha_p Y_{p,\delta} \left(\sum_{\delta \leq \tau \leq t, \tau \in U} d_\tau \right) \geq \alpha_p \sum_{\delta \leq \tau \leq t, \tau \in U} (d_\tau - L_\tau)$

Thus, we have the inequality (2.32) in this case. □

Remark 2.3.2. *The set of valid inequalities (2.30) combined with the constraints (2.26)-(2.29) together construct a polyhedron which has all variables* $(Y_{p,t})_t$ *integral at all its extreme points. And thus, the class of* (k, U) *valid inequalities indeed tightens the linear relaxation of (Sub(p)) (for the proof of this remark, see [16]).*

The intuition underlying the valid inequality (2.30) can be explained as follows: Take the reassembling process $p = I + 1$ as an example. In each period k, we consider the quantity of finished products we currently have in inventory and then look into the future demand needed to be satisfied. For each $\tau \in U$, if $\sum_{\theta=k+1}^{\tau} Y_{I+1,\theta} \geq 1$, the demand d_τ of the period τ can be satisfied by the reassembling process occurring in one of the periods $k+1, ..., \tau$, and thus, does not have to be in stock at the end of period k. Otherwise, if $\sum_{\theta=k+1}^{\tau} Y_{I+1,\theta} = 0$, the demand d_τ cannot be satisfied by reassembling more products, therefore the actual sales $d_\tau - L_\tau$ should be already in stock at the end of period k. The intuition corresponding to refurbishing processes can be explained similarly.

2.3.2 Cut Generation Algorithm for Sub-problems

Given a fractional solution $\left(\tilde{X}_{p,t}, \tilde{Y}_{p,t}, \widetilde{EI}_{I+p,t}, \tilde{L}_t\right)_t$ of the Linear Relaxation of the sub-problem Sub(p), we seek to design an effective *cut generation algorithm* in order to determine which inequalities of the class (k, U) (under the form (2.30)) *should be added* to the formulation while applying a Cut-and-Branch algorithm. We can apply a simple separation algorithm where we check all possible sets U and add all the (k, U) valid inequalities which are violated by $\left(\tilde{X}_{p,t}, \tilde{Y}_{p,t}, \widetilde{EI}_{I+p,t}, \tilde{L}_t\right)_t$, for each period k. However, our preliminary computational results show that this strategy is not effective in term of computation time. Thus, our next objective is to figure out how to carefully select the inequalities (cuts) to be added.

Our first idea is to be more deliberate in building the set U. More precisely, for a given period k, we seek to find the set U providing the *most-violated cut* if it exists. This can be done by applying the following rule to maximize the difference between the left-hand side and the right-hand side of (2.30):

For each $\tau = k+1, ..., T$, *only when* $d_\tau(1 - \sum_{\theta=k+1}^{\tau} \tilde{Y}_{p,\theta}) - \tilde{L}_\tau > 0$ *we would include* τ *into the set* U.

Although our computational results show some improvement by exploiting this idea; the computation time was still prohibitively long for medium-size instances. This is explained by the fact that the cut generation algorithm still generates a large number of similar inequalities, i.e. inequalities involving the same variables and having the same effect on the feasible set of the problem.

Therefore, we now look for alternative "strategies" to improve the algorithm while still constructing the set U by the "most-violated cut" idea; for example, using a stopping criterion, or only choosing period k which gives the most violated inequality in each running iteration or reducing the numbers of inequalities added in each running iteration. After testing many strategies on experimental instances, we choose to use the *"maxU strategy"*, which is the most effective one, in term of reducing the computation time and enhancing the quality of solution.

The idea of this new strategy is to skip the checking procedures for the periods between k and $t = \max\{\tau : \tau \in U\}$ (in case there exists at least one violated cut generated). In fact, by doing this, we do not really "skip" the similar inequalities corresponding to those in-between-periods but only "postpone" and check them up later (if they are truly interesting inequalities, they will be added during a later running iteration). The cut generation algorithm with this "MaxU-strategy" (while keeping constructing set U by most violated cut strategy) is written explicitly as follows:

MaxU-Strategy Cut Generation Algorithm for Sub(p) and $\left(\tilde{X}_{p,t}, \tilde{Y}_{p,t}, \widetilde{EI}_{I+p,t}, \tilde{L}_t\right)_t$

Initialize $k := 0$.

Step 0: *If* $(k \leqslant T-1)$, then go to Step 1; *Else* STOP.

Step 1: Determine subset U (and period $maxU$) using the following rule:

For $\tau = k+1, ..., T$,

- if $\left[d_\tau(1 - \sum_{\theta=k+1}^{\tau} \tilde{Y}_{p,\theta}) - L_\tau\right] > 0$, then $\tau \in U$,
- else $\tau \notin U$.

EndFor

If $U \neq \emptyset$, then set $maxU = max\{\tau, \tau \in U\}$; *Else* stop, go to *step 2.2.*

Step 2: Compute $viol = \widetilde{EI}_{I+p,k} - \alpha_p \sum_{\tau \in U} \left[d_\tau(1 - \sum_{\theta=k+1}^{\tau} \tilde{Y}_{p,\theta}) - L_\tau\right]$.

- *(Step 2.1) If* $viol < 0$, we have found a valid inequality of type (2.30) violated by the current fractional solution (more precisely the most violated valid inequality corresponding to period k). *Then* we set $k := maxU$, go back to step 0.
- *(Step 2.2) Else* there is no violated valid inequality corresponding to period k, we set $k := k+1$, go back to step 0

By using this "MaxU-Strategy Separation Algorithm" for (Sub(p)), we significantly improved the lower bounds quality while adding a reasonable number of cuts to the formulation.

2.3.3 Disassembling Process Sub-problem

Finally, we consider the sub-problem corresponding to the Disassembling echelon, which is slightly different from the previous mentioned sub-problems:

$$\textbf{(Sub(0))} \qquad Z_0^* = min \sum_{t=1}^{T} \left(c_{0,t} Y_{0,t} + \sum_{i=0}^{I} eh_{i,t} EI_{i,t} + l_t L_t \right) \tag{2.33}$$

$$X_{0,t} \leq \min\left\{\sum_{\tau=1}^{t} \alpha_0 r_\tau, \sum_{\tau=t}^{T} \alpha_0 d_\tau\right\} Y_{0,t} \qquad \forall t \tag{2.34}$$

$$EI_{0,t} = EI_{0,t-1} + r_t - d_t + L_t \qquad \forall t \tag{2.35}$$

$$EI_{i,t} = EI_{i,t-1} + \alpha_i X_{0,t} - \alpha_i (d_t - L_t) \qquad \forall i = 1, ..., I, \forall t \tag{2.36}$$

$$EI_{i,t} \geq 0 \qquad \forall i = 0, ..., I \tag{2.37}$$

$$L_t \geq 0, X_{0,t} \geq 0, Y_{0,t} \in \{0, 1\} \qquad \forall t \tag{2.38}$$

Notice that the constraints (2.36) make the sub-problem (Sub(0)) become an uncapacitated single-echelon multi-product lot-sizing problem with strong set-up interactions and lost sales, which is in fact, more complicated than the previous sub-problems.

Still, we can strengthen (Sub(0)) by (k, U) valid inequalities with one of the variables corresponding to the recoverable parts $i = 1, ..., I$, say, choosing $i = 1$. The inequalities thus have the form:

$$EI_{1,k} \geq \alpha_1 \sum_{\tau \in U} \left[d_\tau (1 - \sum_{\theta=k+1}^{\tau} Y_{0,\theta}) - L_\tau \right] \tag{2.39}$$

for any $0 \leq k \leq T-1$ and $U \subset \{k+1, ..., T\}$

2.3.4 Cut-and-Branch Algorithm for Echelon Stock Formulation

Finally, by using the cut generation algorithms to generate the appropriate cuts for each of the sub-problems corresponding to I+2 processes $p = 0, ..., I+1$, we come up with the Cut-and-Branch (and Branch-and-Cut) algorithm which allows us to solve (by exact method) the whole deterministic problem under (EF) formulation. We here describe explicitly the Cut-and-Branch algorithm for solving (EF):

Cut-and-Branch Algorithm for (EF)

- Firstly, solve the Linear Relaxation of (EF). If the solution is integral, STOP, we have found an optimal solution of (EF). Otherwise, denote the fractional solution $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$
- (*Cutting Plane Algorithm*) While there is still some new valid inequalities to be found:
  - Initialize a new constraints set $CUTs = \emptyset$
  - For each $p = 1, ..., I+1$, run the "MaxU-Strategy Cut Generation Algorithm" for (Sub(p)) with valid inequality form (2.30) on the corresponding elements of $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$). Add all the found valid inequalities into the set $CUTs$.
  - Run the "MaxU-Strategy Cut Generation Algorithm" for (Sub(0)) with valid inequality form (2.39) on the corresponding elements of $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$). Add all the found valid inequalities into the set $CUTs$.
  - Add all the cuts found in set CUTs into the formulation of Linear Relaxation of (EF)
  - Re-solve this strengthened Linear Relaxation, update new solution $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$
- After the previous step is done iteratively, the final formulation of the Linear Relaxation is the expected strengthened formulation with all needed cuts added iteratively. Combining this strengthened formulation with the integrality constraints we have the strengthened MILP reformulation of (EF). Solve this MILP reformulation by Branch-and-Bound algorithm

Remark 2.3.3. *We also solve the problem under formulation (EF) with a Branch-and-Cut algorithm,*

in which we apply the above cutting plane algorithm at each node in the Branch-and-Bound tree instead of only at the root node as in Cut-and-Branch algorithm. Our computational results showed that the Branch-and-Cut algorithm leads to either long computation time or memory problems. Trying to improve it, we tested some strategies to reduce the number of nodes in the Branch-and-Bound tree where we applied our cut generation algorithm. Nevertheless, none of the these strategies showed a better performance than the Cut-and-Branch algorithm.

2.4 Computational Results for Deterministic Problem

To test the effectiveness of our solution approach on the problem under formulation (EF), we implemented it by coding the algorithms in C++ language, on a computer running under Windows 10, built with Intel Core-i7 Processor, 8Gb RAM. We used ILOG-CPLEX 12.6 mathematical programming software package for solving the LPs and MILPs to optimality.

To compare the performance, we generated instances where the number of components in a product was set to $I = 5$, the horizon was composed of $T \in \{50, 100, 200, 400, 600, 800, 1000\}$ (for $T > 1000$, all considered algorithms ran into memory problems due to the huge amount of constraints and variables). Each of the above combination between I and T was tested on 5 random instances. The reason for choosing large values of T rather than of I is because we would like to apply a similar algorithm to the stochastic problem (Chapter 3), where T corresponds to the number of nodes in the scenario tree, which is usually large.

Based on the numerical values used in [2] and [13], the data instances were generated according to the following rule (the notation $DU(a, b)$ stands for the discrete uniform distribution within range $[a, b]$):

- The demand for finished products d_t and the quantity of returned used products r_t in each period were generated from $DU(100, 1000)$. The lost-sales unit penalty costs l_t were fixed and set to 10000, for all t.

- The bill of material was generated such that $\alpha_0 = \alpha_{I+1} = 1$, and for each $p = 1, ..., I$, α_p was generated from $DU(1, 6)$.

- The set-up costs for Disassembling process $c_{0,t}$ were generated from $DU(50000, 70000)$, the set-up costs $c_{p,t}$ for each Refurbishing processes $p = 1, ..., I$ were generated from $DU(4000, 8000)$, and lastly, the Reassembling set-up costs $c_{I+1,t}$ from $DU(50000, 70000)$.

- Finally, the unit inventory holding costs for used products $h_{0,t}$ was fixed and set to 1. The unit inventory holding costs $h_{i,t}$ for recoverable parts $i = 1, ..., I$ were generated from $DU(2, 7)$. The serviceable parts unit inventory costs $h_{i,t}$ were generated from $DU(7, 12)$, $i = I + 1, ..., 2I$. Lastly, to ensure non negative echelon costs, we generated finished product unit inventory costs $h_{2I+1,t}$ from $\sum_{i=1}^{I} \alpha_i h_{I+i,t} + \varepsilon$ where we generated ϵ from $DU(50, 100)$.

  Each of these instances were solved with the three following solution approaches:

- *Solve natural formulation (NF) directly by ILOG-CPLEX solver*

- *Solve echelon stock formulation (EF) directly by ILOG-CPLEX solver*
- *Solve formulation (EF) by our Cut-and-Branch algorithm with (k,U) valid inequalities where the auxiliary LPs and strengthened MILP problems are solved by ILOG-CPLEX solver.*

We used the default settings of ILOG-CPLEX except for the stopping time criterion, which was set to 900 seconds. Some of the larger instances could not be solved to optimality before the limit of 900 seconds. For these instances, the quality of (approximated) solution was estimated by the "MIP Gap = (Z*(best integer solution) - Objective value of best node)/Z*(best integer solution)". We also evaluated the quality of the Linear Relaxation with and without (k,U) cuts by the term "LP Gap = (MILP objective value - Linear Relaxation objective value)/MILP objective value". The computational results for $I = 5$ is given in the Table 2.1.

Table 2.1: Computational Results on Deterministic Problem with $I = 5$

		(NF)	CPLEX Directly on (EF)			(k,U) Cut-and-Branch on (EF)			
T	I	Time	LP Gap*	MIP Gap**	Time	Cuts$^+$	LP Gap*	MIP Gap**	Time
50	5	2.65s	20.94%	0.01%	0.47s	759.2	0.27%	0.01%	0.27s
100	5	5.07s	45.32%	0.01%	3.73s	1544.2	0.27%	0.01%	3.17s
200	5	$> 900s$	0.16%	0.01%	5.23s	3338.2	0.14%	0.01%	5.13s
400	5	$> 900s$	34.29%	0.01%	29.50s	6582	0.13%	0.01%	14.03s
600	5	$> 900s$	40.%	0.01%	57.69s	9868.8	0.05%	0.01%	23.30s
800	5	$> 900s$	59.37%	0.14%	739.95s	12711.2	0.11%	0.01%	29.99s
1000	5	$> 900s$	50.47%	0.10%	900.40s	16310.4	0.04%	0.01%	59.26s

The results are average on 5 random instances

The natural formulation is solved directly by CPLEX

For the size (number of variables and constraints) of the instances, see Appendix B

* The smaller the LP Gap, the better quality of the corresponding Linear Relaxation

** CPLEX's MIP Gap by default is 0.01%, thus 0.01% mean "exact" optimal solution

+ Number of (k,U) cuts added by Cut-and-Branch algorithm

Results Discussion: *When using the (NF) formulation, CPLEX could not solve to optimality the instances with $T \geq 200$ within 900 seconds. Using "CPLEX directly on (EF)" rather than "CPLEX directly on (NF)", we can reduce the average computational time from more than 643.97 seconds to 284.14 seconds.*

Moreover, we observe that our Cut-and-Branch algorithm performed much more effectively on reformulation (EF) than using directly the CPLEX solver, especially on the larger instances. Using our (k, U) Cut-and-Branch algorithm rather than CPLEX directly on (EF), we can reduce the average computational time by 92.22%, from more than 248.14 seconds to only 19.31 seconds. Remarkably, for the instances $T \geq 800$, CPLEX could not directly solve them to optimality (with the average MIP gap is 0.12%), while our Cut-and-Branch algorithm found the optimal solutions with default value of MIP Gap (0.01%). Based on these optimistic results, we expect to have the same effectiveness for the stochastic problem.

Chapter 3

Stochastic Problem

The previous chapter is devoted to a study on the Deterministic version of the problem, where we assume that all the parameters are well determined before the first period. However, as acknowledged in the introduction, one of the main challenges in dealing with re-manufacturing Lot-sizing problems is the high uncertainty in the input parameters, due to the fact that companies can hardly control the exact timing, quantity and quality of the used products collected as well as the demand of customers. These input values will only be unfolded gradually as realizations of discrete-time stochastic processes. Moreover, in industrial situations, we usually do not need to build the whole production plan beforehand but only a part of it. We thus often have the possibility to re-adjust it based on the information unfolded later on. In other words, some of the production planning decisions may be postponed to a later point in time, after some additional information of the uncertainty realization becomes available. We thus propose a multi-stage stochastic programming approach enabling us to take advantage of the dynamic nature of the production planning process.

In this chapter, *Section 3.1* introduces scenario trees as a way of representing uncertainties in a multi-stage decision context. This leads to the formulation as a large-size MILP having a structure similar to the deterministic problem studied in Chapter 2. The solution approach from the deterministic case is now extended to the stochastic version, again with natural formulation (STO-NF) and echelon stock reformulation (STO-EF) (*Section 3.2*). *Section 3.3* and *Section 3.4* introduce the cut generation algorithm and the Cut-and-Branch algorithm designed for solving the stochastic problem under its echelon stock reformulation. The computational results obtained with this Cut-and-Branch algorithm is presented in *Section 3.5*. Finally, we suggest an interesting extension using the pairing process and tree inequality concept in *Section 3.6*.

3.1 Uncertainty Representation via Scenario Tree

We would like to model the situations where a company seeks to plan re-manufacturing activities over a multi-period horizon. In each **planning period**, the company has to decide its production and inventory level. In practice, the time is usually discretized into short-length periods (typically corresponding to a shift or a day) in order to ensure the practical relevance of the production plan. However, the time discretization used to update forecasts is usually much sparser (forecasts typically

are updated every week or month). Let b be the number of planning periods between two forecasts updates. New information about uncertainty realization will become available *not* at the end of each period but rather *only after* b periods. Consequently, adjusting the production plan will be required only at the end of every b-periods interval. In other words, the **decision stages** to be used in our multi-stage stochastic approach correspond to sets of b planning periods. Moreover, we assume that the uncertainty realization of all b planning periods belonging to a decision stage s is unfolded at the beginning of the decision stage s.

To reflect the dynamic nature of the decision process, the information structure could be represented by using a scenario tree, where each node n corresponds to a single planning period t belonging to a single decision stage s. It represents the state of the world that can be distinguished by the information unfolded up to that period t. At any non-terminal node of the tree, there are one or several branches to indicate future possible outcomes of the random variable from the current node. Each node of time period $t+1$ is thus connected to a single node belonging to time period t. An example of scenario tree with decision stages consisting of multi-periods can be found in [31].

We only consider the case where every decision stage has the same *length* (denote **b**) and each node at the last period of the stage (except last stage) has the same *number of immediate children/successors* (denote **c**). Notice that the Deterministic problem considered in Chapter 2 is a particular case of the Stochastic problem, in which $c = 1$, (i.e. one single scenario exists). The Figure 3.1 is the example of a scenario tree for planning over 3 decision stages (3 weeks, new realization of forecast become available at the end of each week), each containing 5 planning periods (5 working days), and each node at the last period of stage 1 and 2 having 3 immediate children ($b = 5, c = 3$).

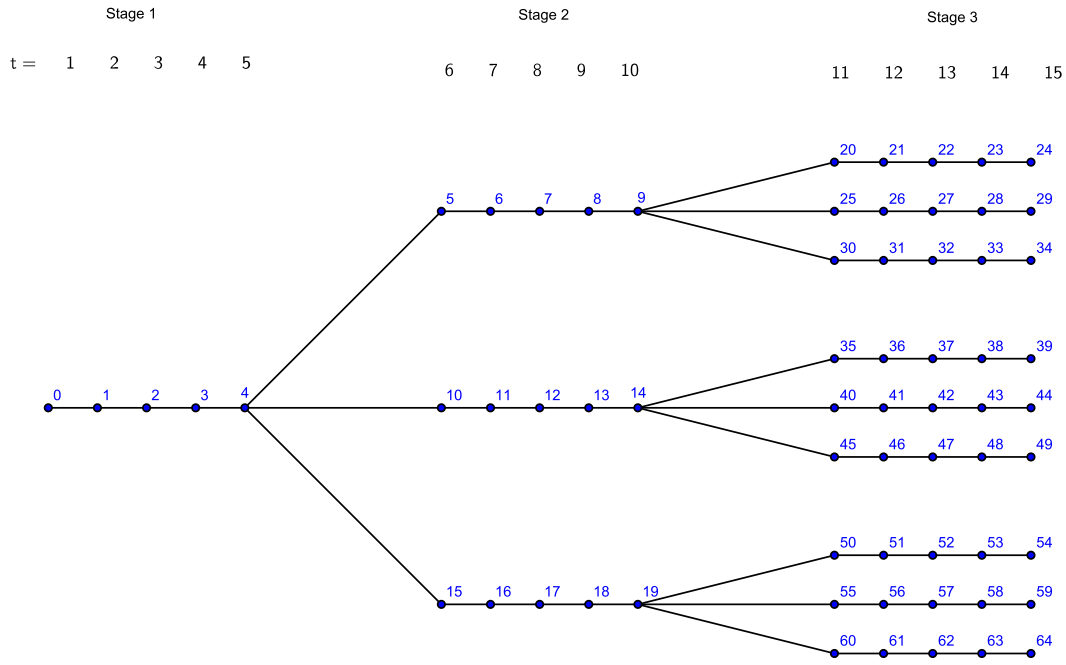

Figure 3.1: Uncertainty representation via a scenario tree

We define the related terms and parameters for the scenario tree:

- $\mathbf{s = 1, 2, ..., S}$: set of decision stages. $\mathbf{t = 1, ..., T}$: set of planning periods. Due the same fixed length of stages, $T = b.S$.

- $\mathbf{n = 0, ..., N}$: set of labels of the nodes in the scenario tree. According to our specific tree structure, we have $N = p\frac{(1-c^S)}{1-c} - 1$. For each node n, $\boldsymbol{\sigma_n}$ denotes the decision stage to which node n belongs. $\boldsymbol{\tau_n}$ denotes the planning period to which node n belongs.

- $n = 0$ is called *the root node*. Terminal nodes (nodes without any children) are called *leaf nodes*. Each non-terminal node is the root node of a sub-tree $\mathcal{T}(n)$. The set of leaf nodes of sub-tree $\mathcal{T}(n)$ is denoted by $\boldsymbol{\mathcal{L}(n)}$ (we can read $\mathcal{L}(n)$ shortly as *leaf set of node* n and understand it as set of leaf nodes connected to n).

- $\mathbf{a_n}$: *predecessor/parent of node* n *in the tree*. Except for the root node, each node in the tree has one and only one predecessor.

- $\mathcal{P}(\mathbf{n}, \mathbf{m})$: path in the scenario tree between two nodes n and m such that $\tau_n < \tau_m$.

- $\boldsymbol{\pi_{a_n,n}}$: probability transition from node a_n to node n. $\boldsymbol{\pi_{n,m}}$: transition probability from node n to node m. $\pi_{n,m} = \prod_{\nu \in \mathcal{P}(n,m)} \pi_{a(\nu),\nu}$. Then, $\boldsymbol{\pi_n}$ is the occurring probability of node n. We set $\pi_0 = 1$ and for $n \neq 0$, we have: $\pi_n = \pi_{1,n}$; moreover, $\sum_{n,\tau_n=t} \pi_n = 1, \forall t$.

3.2 Multi-stage Stochastic Programming Models and Echelon Stock Reformulation

3.2.1 Problem Statement

We consider a full recourse model, i.e. a model in which we assume to have the full flexibility to adjust the production planning at each node of the scenario tree, without taking into account a previously established production plan. The number of parts contained in a finished product is assumed to be deterministic (it does not vary over the nodes). Moreover, we also assume that at each stage, the realization of the random parameters happens before we have to make a decision for this stage. We describe the notation:

- $\mathbf{r_n}$: quantity of used products (returns) collected at node n,

- $\mathbf{d_n}$: quantity of finished products demanded by customers at node n,

- $\mathbf{l_n}$: unit lost-sales penalty cost,

- $\mathbf{sc_{0,n}}$: set-up cost for disassembling process at node n. $\mathbf{sc_{p,n}}$: set-up cost for refurbishing process p at node n, $\forall p = 1, ..., I$. $\mathbf{sc_{I+1,n}}$: set-up cost for reassembling process at node n,

- $\mathbf{h_{0,n}}$: unit inventory cost for used products at node n; $\mathbf{h_{i,n}}$: unit inventory cost for part i (recoverable and serviceable parts) at node n, $\forall i = 1, ..., 2I$; $\mathbf{h_{2I+1,n}}$: unit inventory cost for finished products at node n.

We define the variables using in the model:

- $X_{0,n}$: quantity of used products disassembled at node n ,
- $X_{p,n}$: quantity of items processed in refurbishing process p at node n, $\forall p = 1, ..., I$,
- $X_{I+1,n}$: quantity of finished products reassembled at node n,
- $Y_{0,n} \in \{0, 1\}$: set-up variable for the disassembling process at node n,
- $Y_{p,n} \in \{0, 1\}$: set-up variable for the refurbishing process $p = 1, ..., I$ at node n,
- $Y_{I+1,n} \in \{0, 1\}$: set-up variable for the reassembling process at node n,
- $I_{i,n}$: inventory level of item $i = 0, ..., 2I + 1$ at node n,
- L_n: quantity of lost sales of finished products at node n.

3.2.2 Stochastic MILP Natural Formulation

$$\textbf{(STO-NF)} \qquad Z^* = min \sum_{n=0}^{N} \pi_n \Big[\sum_{p=0}^{I+1} sc_{p,n}.Y_{p,n} + \sum_{i=0}^{2I+1} h_{i,n} I_{i,n} + l_n L_n \Big] \tag{3.1}$$

$$X_{p,n} \leq \min \left\{ \sum_{\nu \in \mathcal{P}(0,n)} \alpha_p r_\nu, \max_{\lambda \in \mathcal{L}(n)} \left\{ \sum_{\nu \in \mathcal{P}(n,\lambda)} \alpha_p d_\nu \right\} \right\} \times Y_{p,n} \quad , \forall p = 0, ..., I+1; \forall n \tag{3.2}$$

$$I_{0,n} = I_{0,a_n} + r_n - X_{0,n} \quad , \forall n = 1, ..., N \tag{3.3}$$

$$I_{0,0} = r_0 - X_{0,0} \tag{3.4}$$

$$I_{i,n} = I_{i,a_n} + \alpha_i X_{0,n} - X_{i,n} \quad , \forall i = 1, ..., I; \forall n = 1, ..., N \tag{3.5}$$

$$I_{i,0} = \alpha_i X_{0,0} - X_{i,0} \quad , \forall i = 1, ..., I \tag{3.6}$$

$$I_{i+I,n} = I_{i+I,a_n} + X_{i,n} - \alpha_i X_{I+1,n} \quad , \forall i = 1, ..., I; \forall n = 1, ..., N \tag{3.7}$$

$$I_{i+I,0} = X_{i,0} - \alpha_i X_{I+1,0} \quad , \forall i = 1, ..., I \tag{3.8}$$

$$I_{2I+1,n} = I_{2I+1,a} + X_{I+1,n} - d_n + L_n \quad , \forall n = 1, ..., N \tag{3.9}$$

$$I_{2I+1,0} = X_{I+1,0} - d_0 + L_0 \tag{3.10}$$

$$I_{i,n} \geqslant 0 \quad , \forall i = 0, ..., 2I+1; \forall n \tag{3.11}$$

$$L_n \geqslant 0 \quad , \forall n \tag{3.12}$$

$$X_{p,n} \geqslant 0, Y_{p,n} \in \{0, 1\} \quad , \forall p = 0, ..., I+1; \forall n \tag{3.13}$$

The objective function (3.1) aims at minimizing the expected cost, over all nodes of the scenario tree. This cost is the sum of the set-up costs, the inventory costs and the lost-sales penalty costs at node n multiplied to the occurring probability π_n of node n. The "big-M" constraints (3.2) can be explained as follows. For each process $p = 0, ..., I + 1$ at node n, we take the minimum between the

quantity of items available to be processed (in the path from root node up to n), and the demand for output items needed to be processed (maximum total demand among all paths from n up to all leaf-nodes in $\mathcal{L}(n)$). We remark that, for the sake of notation, we set $\alpha_0 = \alpha_{I+1} = 1$.

3.2.3 Stochastic Echelon Stock Reformulation Model

Similar to the formulation (NF) of the Deterministic case, the problem (STO-NF) displays a strong coupling between inventory constraints, and thus, is complicated to solve. We thus reformulate this model by using the Echelon Stock variables and Echelon Inventory Costs defined as:

- For finished product $i = 2I+1$: $EI_{2I+1,n} = I_{2I+1,n}$; and $eh_{2I+1,n} = h_{2I+1,n} - \sum_{i=1}^{I} \alpha_i h_{I+i,n}$
- For serviceable parts $i = I+1, ..., 2I$: $EI_{i,n} = I_{i,n} + \alpha_{i-I} I_{2I+1,n}$; and $eh_{i,n} = h_{i,n} - h_{i-I,n}$
- For recoverable parts $i = 1, ..., I$: $EI_{i,n} = I_{i,n} + I_{I+i,n} + \alpha_i I_{2I+1,n}$; and $eh_{1,n} = h_{1,t} - \frac{h_{0,n}}{\alpha_1}$, $eh_{i,n} = h_{i,n}$, $\forall i = 2, ..., I$;
- Finally, for the used product $i = 0$: $EI_{0,n} = I_{0,n} + \frac{I_{1,n}}{\alpha_1} + \frac{I_{I+1,n}}{\alpha_1} + I_{2I+1,n}$; and $eh_{0,n} = h_{0,n}$

Now we can reformulate (STO-NF) into the Stochastic Echelon Stock Reformulation:

$$\textbf{(STO-EF)} \qquad Z^* = min \sum_{n=0}^{N} \pi_n \Big[\sum_{p=0}^{I+1} sc_{p,n}.Y_{p,n} + \sum_{i=0}^{2I+1} eh_{i,n} EI_{i,n} + l_n L_n \Big] \tag{3.14}$$

$$X_{p,n} \leq \min \left\{ \sum_{\nu \in \mathcal{P}(0,n)} \alpha_p r_\nu, \max_{\lambda \in \mathcal{L}(n)} \left\{ \sum_{\nu \in \mathcal{P}(n,\lambda)} \alpha_p d_\nu \right\} \right\} \times Y_{p,n} \quad , \forall p = 0, ..., I+1; \forall n \tag{3.15}$$

$$EI_{0,n} = EI_{0,a_n} + r_n - d_n + L_n \quad , \forall n = 1, ..., N \tag{3.16}$$

$$EI_{0,0} = r_0 - d_0 + L_0 \tag{3.17}$$

$$EI_{i,n} = EI_{i,a_n} + \alpha_i X_{0,n} - \alpha_i d_n + \alpha_i L_n \quad , \forall i = 1, ..., I; \forall n = 1, ..., N \tag{3.18}$$

$$EI_{i,0} = \alpha_i X_{0,0} - \alpha_i d_0 + \alpha_i L_0 \quad , \forall i = 1, ..., I \tag{3.19}$$

$$EI_{i+I,n} = EI_{i+I,a_n} + X_{i,n} - \alpha_i d_n + \alpha_i L_n \quad , \forall i = 1, ..., I; \forall n = 1, ..., N \tag{3.20}$$

$$EI_{i+I,0} = X_{i,0} - \alpha_i d_0 + \alpha_i L_0 \quad , \forall i = 1, ..., I \tag{3.21}$$

$$EI_{2I+1,n} = EI_{2I+1,a_n} + X_{I+1,n} - d_n + L_n \quad , \forall n = 1, ..., N \tag{3.22}$$

$$EI_{2I+1,0} = X_{I+1,0} - d_0 + L_0 \tag{3.23}$$

$$EI_{0,n} - \frac{EI_{1,n}}{\alpha_1} \geqslant 0 \quad , \forall n \tag{3.24}$$

$$EI_{i,n} - EI_{i+I,n} \geqslant 0 \quad , \forall i = 1, ...I, \forall n \tag{3.25}$$

$$EI_{i+I,n} - \alpha_i EI_{2I+1,n} \geqslant 0 \quad , \forall i = 1, ..., I; \forall n \tag{3.26}$$

$$EI_{2I+1,n} \geqslant 0 \quad , \forall n \tag{3.27}$$

$$EI_{i,n} \geq 0 \quad , \forall i = 0, ..., 2I, \forall n \tag{3.28}$$

$$L_n \geqslant 0 \quad , \forall n \tag{3.29}$$

$$X_{p,n} \geqslant 0, Y_{p,n} \in \{0,1\} \qquad ,\forall p = 0, ..., I+1, \forall n \tag{3.30}$$

The group (3.16), (3.18), (3.20), (3.22) are the constraints on echelon inventory at every node from 1 to N linking with its predecessor. Constraints (3.17), (3.19), (3.21), (3.23) reflect echelon inventory at node 0 which has no predecessor. Constraints (3.24)-(3.27) are to ensure the positive sign of the natural inventory variables. Similar to the Deterministic problem, the use of Echelon Stock variables allows us to decompose the problem (under formulation (STO-EF) into a series of single-item single-echelon sub-problems.

3.3 Sub-problems Description and (k,U) Valid Inequalities Forms

By a direct translation from the deterministic case, we consider $I+2$ sub-problems of the Echelon Stock Reformulation Model (STO-EF), corresponding to $I+2$ processes $p = 0, ..., I+1$, linked up by the constraints (3.24)-(3.27). We describe these sub-problems and their corresponding (k,U) valid inequalities:

- For the Refurbishing and the Reassembling echelon, corresponding to each process $p = 1, ..., I+1$, we consider the sub-problem with the form:

$$\textbf{(Sto(p))} \qquad Z_p^* = min \sum_{n=0}^{N} \pi_n \left(sc_{p,n} Y_{p,n} + eh_{I+p,n} EI_{I+p,n} + l_n L_n \right) \tag{3.31}$$

$$X_{p,n} \leq \min \left\{ \sum_{\nu \in \mathcal{P}(0,n)} \alpha_p r_\nu, \max_{\lambda \in \mathcal{L}(n)} \left\{ \sum_{\nu \in \mathcal{P}(n,\lambda)} \alpha_p d_\nu \right\} \right\} \times Y_{p,n} \qquad \forall n \tag{3.32}$$

$$EI_{I+p,n} = EI_{I+p,a_n} + X_{p,n} - \alpha_p d_n + \alpha_p L_n \qquad ,\forall n = 1, ..., N \tag{3.33}$$

$$EI_{I+p,0} = X_{p,0} - \alpha_p d_0 + \alpha_p L_0 \tag{3.34}$$

$$L_n \geqslant 0, EI_{I+p,n} \geqslant 0, X_{p,n} \geqslant 0, Y_{p,n} \in \{0,1\} \qquad ,\forall n \tag{3.35}$$

Notice that f*or each non-leaf node* k, there are $|\mathcal{L}(k)|$ paths from k leading to $|\mathcal{L}(k)|$ leaf nodes corresponding to the last period T.

Then, for each process $p = 1, ..., I+1$, for any non-leaf node k, consider *each leaf node* $\lambda \in \mathcal{L}(k)$, and any set $U_{k,\lambda} \subset \mathcal{P}(c_k, \lambda)$ (where c_k is the child of k belonging to the path $\mathcal{P}(k, \lambda)$), we have the form of $(k, U_{k,\lambda})$ valid inequalities corresponding to the sub-problem (Sto(p)):

$$EI_{p+I,k} \geq \alpha_p \sum_{\nu \in U_{k,\lambda}} \left[D_\nu \left(1 - \sum_{\mu \in \mathcal{P}(c_k,\nu)} Y_{p,\mu} \right) - L_\nu \right] \tag{3.36}$$

- The sub-problem associated with the Disassembling process $p = 0$ has the form :

$$\textbf{(Sto(0))} \qquad Z_0^* = min \sum_{n=0}^{N} \pi_n \left(sc_{0,n} Y_{0,n} + \sum_{i=0}^{I} eh_{i,n} EI_{i,n} + l_n L_n \right) \tag{3.37}$$

$$X_{0,n} \leq \min\left\{\sum_{\nu\in P(0,n)} r_\nu,\ \max_{\lambda\in L(n)}\left\{\sum_{\nu\in P(n,\lambda)} d_\nu\right\}\right\} \times Y_{0,n} \quad \forall n \tag{3.38}$$

$$EI_{0,n} = EI_{0,a_n} + r_n - d_n + L_n \quad ,\forall n = 1,...,N \tag{3.39}$$

$$EI_{0,0} = r_0 - d_0 + L_0 \tag{3.40}$$

$$EI_{i,n} = EI_{i,a_n} + \alpha_i X_{0,n} - \alpha_i d_n + \alpha_i L_n \quad ,\forall i = 1,...,I; \forall n = 1,...,N \tag{3.41}$$

$$EI_{i,0} = \alpha_i X_{0,0} - \alpha_i d_0 + \alpha_i L_0 \quad ,\forall i = 1,...,I \tag{3.42}$$

$$EI_{i,n} \geqslant 0 \quad ,\forall i = 0,...,I, \forall n \tag{3.43}$$

$$L_n \geqslant 0, X_{0,n} \geqslant 0, Y_{0,n} \in \{0,1\} \quad ,\forall n \tag{3.44}$$

Similar to the reasoning for sub-problem (Sub(0)) in deterministic case, to build the valid inequalities for sub-problem (Sto(0)), we can choose variables associated with any recoverable part $i = 1,...,I$; for example, $i = 1$.

For each non-leaf node k, for each leaf node $\lambda \in \mathcal{L}(k)$, and the set $U_{k,\lambda} \subset \mathcal{P}(c_k, \lambda)$ (where c_k is the child of k belonging to the path $\mathcal{P}(k,\lambda)$). The form of $(k, U_{k,\lambda})$ valid inequalities corresponding to sub-problem (Sto(0)):

$$EI_{1,k} \geq \alpha_1 \sum_{\nu\in U_{k,\lambda}} \left[D_\nu\left(1 - \sum_{\mu\in\mathcal{P}(c_k,\nu)} Y_{0,\mu}\right) - L_\nu\right] \tag{3.45}$$

Remark 3.3.1. *Since the above-mentioned valid inequalities corresponding to the Path from each node k to each of its leaf nodes in $\mathcal{L}(n)$, in their paper [8], Y. Guan et al. append them with the term "path inequalities" to be distinguished from the tree inequalities presented later on.*

3.4 Cut Generation Algorithm for Sub-problems and Cut-and-Branch Algorithm

3.4.1 Cut Generation Algorithm Strategies

We now design a cut generation algorithm for selecting the valid inequalities to be added into the formulation while applying a Cut-and-Branch algorithm. Again, learning from previous analysis of deterministic case, obviously we need a careful selection rather than a naive implementation where we add all possible inequalities under the forms (3.36) and (3.45) which are violated by $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$ (the fractional solution of Linear Relaxation problem). After testing on the practical instances, for each sub-problem (Sto(p)), $p = 1,...,I+1$, we come up with a combination of the following strategies (the cut generation algorithm for sub-problem Sto(0) can be similarly obtained):

1/ Strategy in building the set $U_{k,\lambda}$:

For each non-leaf node $k = 0,...,N$, for each leaf node λ in leaf set $\mathcal{L}(k)$, we would be more careful in building up the set $U_{k,\lambda}$ by checking if the term $\alpha_p\left[D_\nu\left(1 - \sum_{\mu\in P(c_k,\nu)} \tilde{Y}_{p,\mu}\right) - \tilde{L}_\nu\right] > 0$, for

every ν in the path $\mathcal{P}(c_k, \lambda)$. Only include $\nu \in U_{k,\lambda}$ if YES (here, c_k denotes the immediate child of node k belonging to the corresponding path $\mathcal{P}(k, \lambda)$).

The set $U_{k,\lambda}$ built by this rule provides the "most violated cut associated with λ", where the difference between left-hand-side and right-hand-side is the largest.

2/ "MaxU-skipping-leaves" strategy:

Similar to the analysis on Deterministic problem, we notice that, by considering all leaf-nodes λ in the leaf set $\mathcal{L}(k)$ and build the corresponding set $U_{k,\lambda}$, we in fact generated many inequalities having a similar effect on the feasible set of (STO-EF). We improve the algorithm by only considering some interesting leaf nodes instead of every leaf node in $\mathcal{L}(k)$. Again, the trick is to postpone the checking procedure on the sets $U_{k,\lambda}$ associated with the "in-between" leaf nodes:

+ Given a non-leaf node k, and the considering leaf $\lambda^* \in \mathcal{L}(k)$; we would build the corresponding set U_{k,λ^*} by above strategy.

+ If $U_{k,\lambda^*} \neq \emptyset$, set $maxU_{k,\lambda_j} = max\{\nu, \nu \in U_{k,\lambda_j}\}$.

+ We erase the considering leaf node λ^* from the leaf set of every node in the path $\mathcal{P}(c_k, maxU_{k,\lambda_j})$ (i.e $\forall \omega \in \mathcal{P}(c_k, maxU_{k,\lambda_j}), L(\omega) = \mathcal{L}(\omega) \backslash \{\lambda^*\}$).

3/ "Most-violated-leaf-node" strategy:

Finally, for each non-leaf node k, according to the "MaxU-skipping-leaves" strategy, the leaf set $\mathcal{L}(k)$ has been updated with smaller or equal cardinality than the original leaf set. For each leaf-node λ^* remained in this updated leaf set $\mathcal{L}(k)$, we compute:

$$viol_{k,\lambda^*} = \widetilde{EI}_{p+I,k} - \alpha_p \sum_{\nu \in U_{k,\lambda^*}} \left[D_\nu \left(1 - \sum_{\mu \in \mathcal{P}(c_k,\nu)} \tilde{Y}_{p,\mu} \right) - L_\nu \right]$$

Among those remained leaf-nodes, we only choose the leaf-node λ_{viol} which provides the "most-violated cut". It is such that: $viol_{k,\lambda_{viol}} = \min \{viol_{k,\lambda^*} : \lambda^* \in \mathcal{L}(k)\}$ (remember that this leaf-set $\mathcal{L}(k)$ is already updated by "MaxU-skipping-leaves" strategy).

3.4.2 Cut Generation Algorithm for Sub-problems

As an example, we write down explicitly the Cut Generation Algorithm on sub-problem (Sto(p)) with the valid inequality form (3.36). ($\forall p = 1, ..., I+1$)

Cut Generation Algorithm for (Sto(p)) and $\left(\tilde{X}_{p,n}, \tilde{Y}_{p,n}, \widetilde{EI}_{I+p,n}, \tilde{L}_n\right)_n$

- Initialize $k = 0$.
- **Step 0:** If $k \leqslant N$, go to Step 1; else STOP.
- **Step 1:** If k is leaf node, stop, go to Step 4. Else if k is a non-leaf node, set $Min\_viol := 0$, then:
  **(Step 1.1)* If $\mathcal{L}(k) = \emptyset$, stop, go to step 3. Otherwise, if $\mathcal{L}(k) \neq \emptyset$, we set $\lambda^* = \min \{\lambda : \lambda \in L(k)\}$.
  **(Step 1.2)* Determine subset U_{k,λ^*} using the following rule: Iteratively check for each $\nu \in \mathcal{P}(c_k, \lambda^*)$:

+ If $\sum_{\mu \in \mathcal{P}(c_k,\nu)} \tilde{Y}_{i,\mu} \geq 1$, stop, come to step 1.3.

+ Else if $\sum_{\mu \in \mathcal{P}(c_k,\nu)} \tilde{Y}_{i,\mu} < 1$:

- if $\alpha_p \left[D_\nu \left(1 - \sum_{\mu \in P(c_k,\nu)} \tilde{Y}_{p,\mu} \right) - \tilde{L}_\nu \right] > 0$, then $\nu \in U_{k,\lambda^*}$.
- else $\nu \notin U_{k,\lambda^*}$.

*(Step 1.3) If $U_{k,\lambda^*} \neq \emptyset$, set $maxU_{k,\lambda^*} = max\{\tau, \tau \in U_{k,\lambda^*}\}$; else stop, go to step 2.2.

- **Step 2**: Compute $viol_{k,\lambda^*} = \widetilde{EI}_{p+I,k} - \alpha_p \sum_{\nu \in U_{k,\lambda^*}} \left[D_\nu \left(1 - \sum_{\mu \in \mathcal{P}(c_k,\nu)} \tilde{Y}_{p,\mu} \right) - L_\nu \right]$.

+ If $viol_{k,\lambda^*} < 0$:

- We have found a valid inequality of type (3.36) violated by the current fractional solution (more precisely the most violated valid inequality corresponding to node k and leaf node λ^*).

- Then we erase the leaf node λ^* from the leaf set $\mathcal{L}(\omega)$, for every in-between node $\omega \in \mathcal{P}(k, maxU_{k,\lambda^*})$, i.e. update $L(\omega) = \mathcal{L}(\omega) \backslash \{\lambda^*\}, \forall \omega \in \mathcal{P}(k, maxU_{k,\lambda^*})$.

- If $viol_{k,\lambda^*} < Min\_viol$, we have found the current most violated cut among those corresponding to all leaf-nodes remained in the current updated leaf-set $\mathcal{L}(k)$. Then we update $Min\_viol := viol_{k,\lambda^*}$.

- Go back to step 1.1 with updated $\mathcal{L}(k)$.

+ Else if $viol_{k,\lambda^*} \geq 0$, (Step 2.2): there is no violated valid inequality corresponding to node k and leaf node λ^*, we exclude λ^* from $\mathcal{L}(k)$, go back to step 1.1.

- **Step 3:** If the final value of $Min\_viol < 0$, the final cut associated to that final value of the term Min_viol is the needed "most-violated-leaf" cut corresponding to node k we seek for. Otherwise, there is no cut corresponds to node k.

- **Step 4:** Update $k := k + 1$, go back to step 0.

For the Disassembling echelon sub-problem, we can build a similar cut generation algorithm by replacing the corresponding terms and inequalities form.

3.4.3 Cut-and-Branch Algorithm for Stochastic Echelon Stock Formulation

Finally, we come up with the Cut-and-Branch algorithm for solving our stochastic problem under formulation (STO-EF):

Cut-and-Branch Algorithm for (STO-EF)

- Firstly, solve the Linear Relaxation of (STO-EF). If the solution is integral, STOP, we have found an optimal solution of (STO-EF). Otherwise, call the fractional solution by $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$.

- (*Cutting Plane Algorithm*) While there is still some new valid inequalities to be found:
  - Initialize a new constraints set $CUTs = \emptyset$.
  - For each process $p = 1, ..., I+1$ of the Refurbishing and Reassembling echelons, reset the leaf set $\mathcal{L}(n)$ for every node $n = 0, ..., N$ according to the scenario-tree. Then, run the "Cut Generation Algorithm" for (Sto(p)) with valid inequality form (3.36) on the corresponding elements of $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$). Add all the found valid inequalities into the set $CUTs$.
  - Reset the leaf set $\mathcal{L}(n)$ for every node $n = 0, ..., N$ according to the scenario-tree. Run the "Cut Generation Algorithm" for (Sto(0)) with valid inequality form (3.45) on the corresponding elements of $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$). Add all the found valid inequalities into the set $CUTs$.
  - Add all the cuts found in set CUTs into the formulation of strengthened Linear Relaxation of (STO-EF)
  - Re-solve this strengthened Linear Relaxation, update new solution into $\left(\tilde{X}, \tilde{Y}, \widetilde{EI}, \tilde{L}\right)$.
- After the previous step, the final formulation of the Linear Relaxation is the expected strengthened formulation with all needed cuts added iteratively. Combining this strengthened formulation with the integrality constraints we have the strengthened MILP reformulation of (STO-EF). Solve this MILP reformulation by Branch-and-Bound algorithm.

Remark 3.4.1. *The computational results suggested that the Branch-and-Cut algorithm performs on (STO-EF) less effectively than the Cut-and-Branch algorithm.*

3.5 Computational Results for Stochastic Problem

The computer specifications and computer tools (C++, ILOG-CPLEX) used for our computational experiments on the Stochastic problem are the same as the ones used for the experiments done on Deterministic problem.

We considered the number of components in a product, $I = 5$ and $I = 10$, and consecutively changed the length-of-stage b and the number of immediate successors c of each last-period-of-stage node. As a consequence, the combination of these parameters generated scenario trees which have the maximum label of nodes $N \in \{64, 104, 200, 424, 604, 727, 1092, 1455, 1704\}$ (algorithms run into memory problems on larger size instances). For each scenario tree size, we randomly generated 5 instances.

For the sake of comparison, the range of parameters generated were kept totally the same as in Deterministic version. In constructing the scenario tree, we set the probability transition from each node k at the end of each stage (nodes where branching process occurring on the scenario tree), to each of its immediate successor c^k as: $\pi_{k,c^k} = \frac{1}{c}$.

For each instance, we tested two solution approaches on (STO-EF) model:

- *Solve the deterministic problem under the formulation (STO-EF) directly by ILOG-CPLEX.*
- *Solve formulation (STO-EF) by our Cut-and-Branch algorithm where the auxiliary LPs and strengthened MILP problems are solved by ILOG-CPLEX solver.*

Once again, we implemented the algorithms with all default settings of ILOG-CPLEX, except for the stopping time criterion which was set to 900 seconds. We also evaluated the quality of solutions and the Linear Relaxation by the terms MIP Gap and term LP Gap. The computational results for the case $I = 5$ are given in the table (3.1). For the case $I = 10$, please see Appendix C.

Table 3.1: Computational Results on Stochastic Problem for $I = 5$

		CPLEX directly on (STO-EF)			**(k,U) Cut-and-Branch on (STO-EF)**			
N	I	LP Gap*	MIP Gap**	Time	Cuts$^+$	LP Gap*	MIP Gap**	Time
64	5	7.86%	0.01%	1.36s	701.4	1.52%	0.01%	1.18s
104	5	4.67%	0.01%	2.42s	964.6	1.09%	0.01%	2.75s
200	5	7.21%	0.01%	9.58s	2241.2	1.16%	0.01%	7.53s
424	5	13.06%	0.01%	90.86s	4056.8	1.66%	0.01%	12.40s
604	5	14.63%	0.02%	522.61s	6438.2	1.49%	0.01%	29.11s
727	5	5.40%	0.02%	588.78s	4181.2	1.53%	0.01%	211.60s
1092	5	8.91%	0.03%	746.38s	8994.6	1.45%	0.01%	197.02s
1455	5	10.14%	0.05%	900.70s	14310.6	1.27%	0.02%	649.69s
1704	5	15.86%	0.18%	900.71s	17191.4	1.60%	0.04%	900.90s

The results are average on 5 random instances

* The smaller the LP Gap, the better quality of the corresponding Linear Relaxation

** CPLEX's MIP Gap by default is 0.01%, thus 0.01% mean "exact" optimal solution

* ***Results Discussion***: *The computational results suggested that the proposed Cut-and-Branch algorithm compares well with the direct resolution of formulation (STO-EF) with ILOG-CPLEX. Namely:*

- *Without adding valid inequalities to the formulation, ILOG-CPLEX had difficulties in solving the stochastic problem, even with the echelon stock reformulation (STO-EF). Specifically, within the time limit of 900 seconds, ILOG-CPLEX failed to optimize the solutions of (STO-EF) for instances with $N \geq 604$ (for model (STO-NF) it failed for instances with $N \geq 200$).*
- *Also using reformulation (STO-EF); however, our Cut-and-Branch algorithm performed ways better than the direct use of ILOG-CPLEX. The effectiveness displayed in both computation time and quality of solutions. Our algorithm solved the problems to optimality for instances with N up to 1092 and significantly reduced the total computation time by 75%.*
- *Since we set the stopping criterion on computation time, when it got to big instances ($N = 1455$ and $N = 1704$), both the direct application of CPLEX and the use of our Cut-and-Branch*

algorithm could not ultimately optimize the solution within 900 seconds. However, we can observe that after 900 seconds, our Cut-and-Branch algorithm succeeded in closing the MIP-gap (the difference between the objective function value of best integer solution and the so-far best lower bound) almost up to the default gap 0.01% (average 0.03%), i.e. the solution approximated by our algorithm is much closer to optimality than the approximated solution found by ILOG-CLPEX on its own (average MIP gap is 0.12%).

3.6 Pairing Process and Tree Inequalities

The previous section suggests a quite optimistic result of our Cut-and-Branch algorithm performance. However, we would like to further improve the algorithm. This section suggests the idea of using the *Pairing operation* (proposed in [8]) to create new and stronger valid inequalities of the problem (STO-EF) based on (k, U) inequalities, which is indicated by the following theorem:

Theorem 3.6.1. *Suppose that the inequalities $g_1X + a_1Y \geqslant b_1$ and $g_2X + a_2Y \geqslant b_2$ with $b_1 \leqslant b_2$ are valid for the set X, then the "pairing inequality" $\varphi X + \phi Y \geqslant b_2$ is valid for X (where $\varphi = \max\{g_1, g_2\}$ and $\phi = \min\{a_1 + (b_1 - b_2), \max\{a_1, a_2\}\}$).*

The authors of [8] suggest that, by applying this pairing process on the "path inequalities", we would obtain stronger (in term of strengthening the formulation) valid inequalities called "tree inequalities". However, in [8], Y. Guan et al. take their interest in the class of valid inequalities named (Q, S_Q) which is generalized from classic (l, S) class, and design a separation algorithm for these tree inequalities by using a heuristic approach (after strengthening the generated tree inequalities).

We would like to investigate this idea, and try to design an exact separation algorithm for tree inequalities generated from any subset of (k, U) valid inequalities class, or at least some certain or special cases. Notice that we can rewrite the inequality (3.36) associated with node k and leaf λ as the following form: (to ease the notation, we temperately forget index p of processes):

$$\sum_{\nu \in \mathcal{P}(0,a_k)} (0.EI_\nu + 0.Y_\nu) + (EI_k + 0.Y_k) + \sum_{\nu \in \mathcal{P}(c_k, \max U_{k,\lambda})} \left(0.EI_\nu + D_\nu^{U_{k,\lambda}}.Y_\nu\right) + \sum_{\nu \in \mathcal{P}(c_{\max U_{k,\lambda}}, \lambda)} (0.EI_\nu + 0.Y_\nu) \geq \sum_{\nu \in U_{k,\lambda}} (D_\nu - L_\nu)$$

where we denote $D_\nu^U \equiv \sum_{\theta \in [\mathcal{P}(\nu, \max U) \cap U]} D_\theta$ for any node ν, set U; and as before, $\forall j \in \mathcal{P}(0, \lambda)$, denote c_j the child of j which belongs in that path.

On first investigation, in the case where we consider the tree inequalities paired from any 2 path inequalities corresponding to 2 arbitrary nodes k and j, to their two arbitrary leaf nodes λ_k, λ_j, the amount of path inequalities to check is exponential in the number of nodes. Moreover, it is impossible, in our opinion, to design a general exact separation algorithm for a tree inequality from any such path inequalities. Hence, the open question is to which cases of path inequalities we could have a effective separation algorithm of the corresponding tree inequality. Within the limitation of this project, we did not succeed in answering this question, it would be the potential extension of the future projects.

Chapter 4

Conclusion and Perspective

4.1 Findings

- (1) The internship project considers the Uncapacitated Multi-Echelon Re-manufacturing Lot-Sizing Model with Lost Sales under Uncertainty. Studying and solving the two versions (Deterministic and Stochastic) of the problem gives theoretical and practical benefits in term of operational research theory, economy and environment protection. We propose models containing three echelons: Disassembling, Refurbishing, and Reassembling; mathematically expressed as MILP problems. Due to the computational experiments, we are confident that our algorithms could be used in real-life situations to solve small upto medium size problems.

- We investigate the Deterministic problem, where we temporarily assume that all input parameters are determined beforehand:

  (2) Utilizing the concept of echelon stock, we reformulate the model. This gives us advantage in decomposing the problem into a series of sub-problems and solving them separately.

  (3) We succeed in applying the idea and writing the specific form of (k,U) valid inequalities class for our sub-problems. We prove the validity and effectiveness of (k,U) form for our sub-problems.

  (4) We design the Cut-and-Branch algorithm for solving the Echelon Stock Reformulation Deterministic model and getting the computational results where our Cut-and-Branch algorithm performed well-better than normal Branch-and-Bound algorithm.

- We study the Stochastic problem, where some input parameters (quantity of returns and demand of customers) are random processes and would only be unfolded little by little at time progressing:

  (5) By using the scenario tree concept, we represent the information data structure, and model the Stochastic problem as a MILP problem

  (6) We reformulate the problem into the Stochastic Echelon Stock model which allows us to decompose and analyse the corresponding sub-problems as well as avoid the strong connection between inventory constraints.

  (7) Exploiting the knowledge obtained from Deterministic case, we modify and improve the cut generation algorithm for sub-problems corresponding to production processes.

(8) We design a Cut-and-Branch algorithm on the Stochastic problem under echelon stock reformulation, and then we run some computational experiments to test its performance. The results show significant improvements as compared to the algorithms implemented by ILOG-CPLEX solver itself. Our Cut-and-Branch algorithm succeeds in solving to optimality medium-size problems. In general, it successively reduces the computation time and enhances the quality of solutions on computational experiments

(9) The results suggest new ideas in strengthening the formulation by pairing process and tree inequalities.

4.2 Project Limitation and Extension

In this present work, we assumed an unlimited production capacity (only with set-up cost and without cost on production unit), it would be interesting to see how the algorithms behave in case the production capacity is limited. Moreover, in real-life situations, not only the demand and returns quantity, but also the quality of used-products brought back by customers might be uncertain. This will thus have impact on the data structure of the problems. For the time being, we have not been able to effectively reflect this stochastic factor of returns quality in our model, since it is not a straightforward application as demand and returns quantity.

Regarding computational experiments, within the limited time of this internship and given facilities, we only tested on a small number of instances. It would be interesting to have results on more varied cases and sizes on more powerful computers. In constructing the scenario tree, we also made some assumptions on the probability transition, the length of stages, the number of immediate children,etc. Relaxing these assumptions is another potential perspective for future work.

In this work, we did not succeed in implementing a Branch-and-Cut algorithm which can outrun the Cut-and-Branch algorithm (although it already performed quite well); however, by utilizing more programming tools and macros from ILOG-CPLEX or perhaps another solver, we hope we could manage to improve the Branch-and-Cut algorithm in near future.

The idea of pairing process and tree inequality suggests a new open approach to create even stronger valid inequalities for our Stochastic problem. The remaining challenge is to design a practically effective separation algorithm.

Bibliography

[1] N. Absi, S. Kedad-Sidhoum, 2008. The multi-item capacitated lot-sizing problem with setup times and shortage costs. *European Journal of Operational Research*, vol. 185, no.3, pp. 1351–1374, 2008.

[2] H. Ahn, D. Lee and H. Kim., 2011. Solution algorithms for dynamic lot-sizing in re-manufacturing systems. *International Journal of Production Research*, vol. 49(22), pp 6729-6748.

[3] R. Jans and Z. Degreave, 2008. Modeling industrial lot-sizing problems: a review. *International Journal of Production Research*, Taylor & Francis, 2008, 46 (06), pp.1619-1643.

[4] R. Jans and Z. Degreave, 2007. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, 177 (2007) 1855–1875.

[5] Kuik, R. and M. Salomon , and L. Van Wassenhove, 1994. Batching Decisions: Structure and Models. *European Journal of Operational Research*, 75 (1994), 243–263.

[6] Y. Guan, S. Ahmed, G. L. Nemhauser, A. J. Miller., 2004. A Branch-and-Cut Algorithm for the Stochastic Uncapacitated Lot-Sizing Problem. *Mathematical Programming*, 105:55-84, 2004.

[7] Y. Guan, S. Ahmed and G. L. Nemhauser., 2006. Sequential pairing of mixed integer inequalities. *Discrete Optimization*, Vol 4 (1), pp 21-39.

[8] Y. Guan, S. Ahmed and G. L. Nemhauser., 2009. Cutting planes for multi-stage stochastic integer programs. *Operations Research*, vol. 57 (2), pp 28-298, 2009.

[9] Ilgin M.A., Gupta S.M., 2010. Environmentally conscious manufacturing and product recovery (ECMPRO): A review of the state of the art. *Journal of Environmental Management,* 91 (2010) 563–591.

[10] Guide M.A., Jr, V.D.R., 2000. Production planning and control for remanufacturing: industry practice and research needs. *Journal of Operations Management*, 18 (4), 467–483.

[11] M.J.R. Helmrich, R. Jans, W. van den Heuvel and A.P.M. Wagelmans., 2014. Economic lot-sizing with re-manufacturing: complexity and efficient formulations. *IIE Transactions, vol 46(1)*, pp 67-86, 2014.

[12] Hwang, H.C, van den Heuvel, W, and Wagelmans, A.P.M., 2013. The economic lot-sizing problem with lost sales and bounded inventory. *IIE Transactions*, 45(8), 912–924. doi:10.1080/0740817X.2012.724187.

[13] Jayaraman, V., 2006. Production planning for closed-loop supply chains with recovery and reuse: an analytical approach. *International Journal of Operation Management*, 18(4),467-483.

[14] K. Kim, I. Song, J. Kim, B. Jeong, 2006. Supply planning model for re-manufacturing system in reverse logistics environment. *Computers & Industrial Engineering*, 51 (2006) 279–287.

[15] S. Küçükyavuz, 2012. On mixing sets arising in chance-constrained programming. *Math. Program.*, Ser. A (2012) 132:31–56. DOI 10.1007/s10107-010-0385-3.

[16] M.Loparic, Y. Pochet and L.A. Wolsey., 2001. The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming*, vol. 89, pp 487-504, 2001.

[17] Lund, R.T., 1984. Re-manufacturing. *Technology Review*, 87 (2), 18-28.

[18] G.S. Piperagkas, I. Konstantaras, K. Skouri and K.E. Parsopoulos., 2012. Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics. *Computers & Operations Research*, 39 (2012) 1555–1565.

[19] Y. Pochet and L. Wolsey., 2005. Production Planning by Mixed Integer Programming. *Springer Series in Operations Research and Financial Engineering*. ISBN-10: 0-387-29959-9.

[20] Y. Pochet and L. Wolsey., 1991. Solving multi-item lot-sizing problems using strong cutting planes. *Management Science*, vol 37(1), pp 53-67, 1991.

[21] P. Pineyro, O. Viera., 1991. The economic lot-sizing problem with remanufacturing and one-way substitution. *Int. J. Production Economics*, 124 (2010) 482–488.

[22] F. Sahling, 2013. A Column-Generation Approach for a Short-Term Production Planning Problem in Closed-Loop Supply Chains. *Business Research*, Vol 6, Issue 1, May 2013, 55–75.

[23] M. Di Summa and L.A.Wolsey., 2007. Lot-Sizing on a Tree. *Opearations Research Letters*, 36 (2008) 7-13.

[24] J. Sung and B. Jeong., 2014. A Heuristic for Disassembly Planning in Remanufacturing System. *The Scientific World Journal*, Vol 2014, Article ID 949527, 10 pages.

[25] C. Tamar and Y. Liron, 2015. The Periodic Joint Replenishment Problem is Strongly NP-Hard. Eprint: arXiv:1511.02454.

[26] H. Tempelmeier, S. Herpers., 2011. Dynamic uncapacitated lot sizing with random demand under a fillrate constraint. *European Journal of Operational Research* 212 (2011) 497–507.

[27] R. H. Teunter, Z. P. Bayindir and W. Van Den Heuvel., 2006. Dynamic lot-sizing with product returns and re-manufacturing. *International Journal of Production Research*, vol. 44(20), pp 4377-4400, 2006.

[28] V. Vargas, 2009. An optimal solution for the stochastic version of the Wagner–Whitin dynamic lot-size model. *European Journal of Operational Research*, 198 (2009) 447–451.

[29] H.M. Wagner and T. Whitin, 1958. Dynamic version of the economic lot size model. *Management Science*, Vol. 5, pp. 89–96, 1958.

[30] T. Wu, C. Zhang, Z. Liang, S. C.H.Leung., 2013. A Lagrangian relaxation-based method and models evaluation for multi-level lot-sizing problems with back-orders. *Computers & Operations Research*, vol. 40, pp 1852-1863, 2013.

[31] M.K. Zanjani, M. Nourelfath and D. Ait-Kadi., 2013. A multi-stage stochastic programming approach for production planning with uncertainty in the quality of raw materials and demand.*International Journal of Production Research*, Vol. 48, No. 16, 15 August 2010, 4701–4723.

[32] M. Zhang, S. Kucukyavuz, H. Yaman, 2012. A polyhedral study of multi-echelon lot sizing with intermediate demands. *Operations Research*, vol 60 (4), pp 918 - 935 (2012).

Appendices

Appendix A: Small Illustrative Example of Lot-Sizing Modelled as MILP

- We introduce a deterministic uncapacitated single-item production planning, whose:
  - Planning horizon: T = 10 days
  - Demand of customers at the end of each period:

Day	1	2	3	4	5	6	7	8	9	10
Demand	0	10	0	0	0	10	0	0	10	20

  - Uncapacitated production: Company has unlimited production resources, and once after setting up the process, it can produce any quantity of products, which does not depend on the availability of the machines or human resources.
  - The Set-up cost is 500€ for each time we decide to operate the production process; and unit inventory cost is 5€ per unit for storing through one period.
  - The objective is to minimize the total cost while satisfying every demand of customers at the end of corresponding periods.
- To illustrate the needs in trading-off between minimizing the set-up cost and inventory holding cost, we consider the three following perspective:
  - **Perspective 1: Only focus on minimizing set-up cost:** A rather naive idea is that, since we have uncapacitated production, we can produce single large-size lot of products which cover the sum of every demand in the future, i.e only have to pay the set-up cost once. In our specific example, this means set-up and produce 50 products in period 2, and sell to customers 10 products. The remained 40 products will be stocked in inventory and gradually sell correspondingly in period 6, 9 and 10. In the end, the total cost is 500€+ 1350€= 1850€ (illustrated in Figure A1)
  - **Perspective 2: Only focus on minimizing total inventory holding cost:** In contrast, we can think of producing just enough products satisfying demand in each periods to reduce as much as possible the payment on inventory. More details for our example, we set-up and produce successively 10, 10, 10 and 20 products at period 2, 6, 9 and 10. The total cost will then thus be: 500€+ 500€+ 500€+ 500€= 2000€ (illustrated in Figure A2)

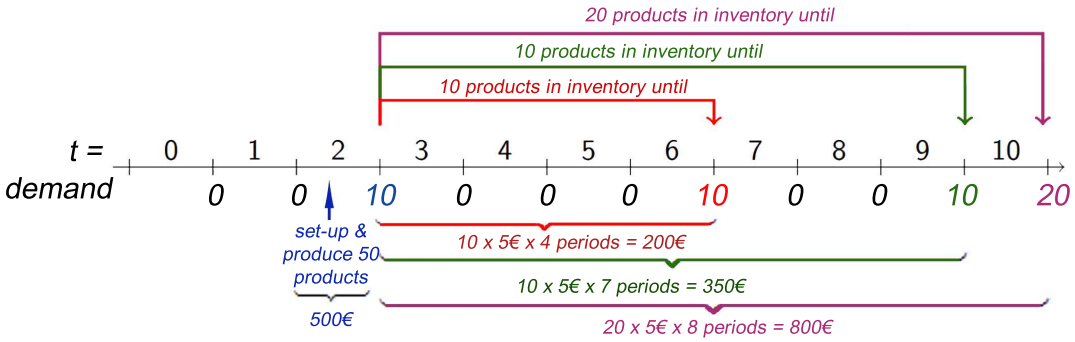

Figure A1: Perspective 1 - Only focus on minimizing set-up cost

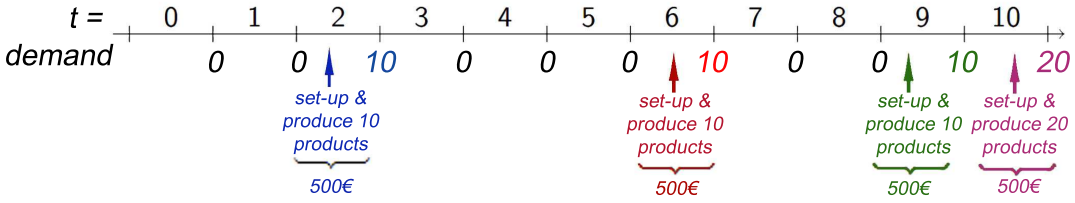

Figure A2: Perspective 2 - Only focus on minimizing total inventory holding cost

– **Perspective 3: Trading-off between set-up and inventory holding costs:** We show that when we find the balance between minimizing set-up cost and inventory cost, the total cost we have to pay is much smaller than the other two naive perspectives. For our example, we should set-up and produce 20 products in period 2, sell 10 products at the end of period 2. The remained 10 products will be stocked and sell at the end of period 6. We then set-up and produce 30 products in period 9, sell 10 products in period 9 and put 20 into inventory for 1 period and sell them in period 10. The total cost is: 500€+ 500€+ 300€= 1300€ (illustrated in Figure A3)

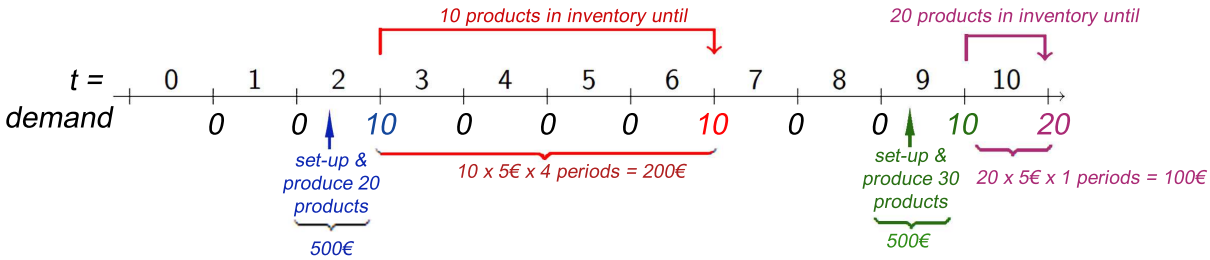

Figure A3: Perspective 3 - Trading-off between set-up and inventory holding costs

Appendix B: Computational Experiments for Deterministic Problem

Table 1: Size Description of the Computational Instances for Deterministic Problem

T	I	Number of Variables	Number of Constraints *
50	5	1377	1562
100	5	2727	3112
200	5	5427	6212
400	5	10827	12412
600	5	16227	18612
800	5	21627	24812
1000	5	27027	31012

* The number of constraints does not include positivity and integrality constraints

Appendix C: Computational Experiments for Stochastic Problem

Table 2: Computational Results on Stochastic Problem for $I = 10$

		CPLEX directly on (STO-EF)			**(k,U) Cut-and-Branch on (STO-EF)**			
N	I	LP Gap*	MIP Gap**	Time	Cuts^{+}	LP Gap*	MIP Gap**	Time
64	10	11.57%	0.01%	1.56s	1025.8	1.70%	0.01%	1.63
104	10	7.47s%	0.01%	2.60s	1597.6	1.37%	0.01%	2.70s
200	10	11.37%	0.01%	6.85s	3429.2	1.65%	0.01%	6.35s
424	10	11.42%	0.01%	35.84s	6877.2	1.71%	0.01%	25.06s
604	10	8.70%	0.01%	675.91s	11123	1.30%	0.01%	410.63s
727	10	6.29%	0.01%	33.84s	7386.4	1.56%	0.01%	35.49s
1092	10	6.48%	0.02%	353.84s	15254	1.29%	0.01%	143.52s

The results are average on 5 random instances

For the specific scenario tree structure and size of each instance, see Appendix C

For instances with more than 1093 nodes, both approaches run into memory problems

* The smaller the LP Gap, the better quality of the corresponding Linear Relaxation

** CPLEX's MIP Gap by default is 0.01%, thus 0.01% mean "exact" optimal solution

+ Number of (k,U) cuts generated by Cut-and-Branch algorithm

Table 3: Size Description of the Computational Instances for Stochastic Problem

N	I	Tree Description	Number of Variables	Number of Constraints *
64	5	$S = 3, b = 5, c = 3$	1755	2470
104	5	$S = 3, b = 5, c = 4$	2835	3990
200	5	$S = 4, b = 5, c = 3$	5400	7600
424	5	$S = 4, b = 5, c = 4$	11475	16150
604	5	$S = 5, b = 5, c = 3$	16335	22990
727	5	$S = 6, b = 2, c = 3$	19656	27664
1092	5	$S = 6, b = 3, c = 3$	29484	41496
1455	5	$S = 6, b = 4, c = 3$	39312	55335
1704	5	$S = 5, b = 5, c = 4$	46035	64790
64	10	$S = 3, b = 5, c = 3$	3055	4420
104	10	$S = 3, b = 5, c = 4$	4935	7140
200	10	$S = 4, b = 5, c = 3$	9400	13600
424	10	$S = 4, b = 5, c = 4$	19975	28900
604	10	$S = 5, b = 5, c = 3$	28435	41140
727	10	$S = 6, b = 2, c = 3$	34216	49504
1092	10	$S = 6, b = 3, c = 3$	51324	74256

* The number of constraints does not include positivity and integrality constraints