A multi-stage stochastic integer programming approach for a multi-echelon lot-sizing problem with returns and lost sales

Franco Quezada[a], Céline Gicquel[b], Safia Kedad-Sidhoum[c], Dong Quan Vu[d]

[a]*Sorbonne Université, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France*
[b]*Université Paris-Saclay, Laboratoire de Recherche en Informatique, LRI, 91190 Gif-sur-Yvette, France*
[c]*CNAM, Centre d'Études et de Recherche en Informatique et Communications, CEDRIC, F-75003 Paris, France*
[d]*Nokia Bell Labs, Nokia Paris-Saclay, Route de Villejust, 91620 Nozay, France*

Abstract

We consider an uncapacitated multi-item multi-echelon lot-sizing problem within a remanufacturing system involving three production echelons: disassembly, refurbishing and reassembly. We seek to plan the production activities on this system over a multi-period horizon. We consider a stochastic environment, in which the input data of the optimization problem are subject to uncertainty. We propose a multi-stage stochastic integer programming approach relying on scenario trees to represent the uncertain information structure and develop a Branch-and-Cut algorithm in order to solve the resulting mixed-integer linear program to optimality. This algorithm relies on a new set of tree inequalities obtained by combining valid inequalities previously known for each individual scenario of the scenario tree. These inequalities are used within a cutting-plane generation procedure based on a heuristic resolution of the corresponding separation problem. Computational experiments carried out on randomly generated instances show that the proposed Branch-and-Cut algorithm performs well as compared to the use of a stand-alone mathematical solver. Finally, rolling horizon simulations are carried out to assess the practical performance of the stochastic planning model with respect to the deterministic model.

Keywords:
Stochastic lot-sizing, remanufacturing system, lost sales, multi-stage stochastic integer programming, scenario tree, valid inequalities, branch-and-cut algorithm

1. Introduction

Industrial companies face an increasing pressure from customers and governments to become more environmentally responsible and mitigate the environmental impact of their products. One way of achieving this objective is to remanufacture the products once they have reached their end-of-life. Remanufacturing is defined as a set of processes transforming end-of-life products (used products or returns) into like-new finished products, once again usable by customers, mainly by rehabilitating damaged components [1]. By reusing the materials and components embedded in used products, it both contributes in reducing pollution emissions and natural resource consumption.

In this work, we study a remanufacturing system which involves three key processes: disassembly of used products brought back by customers, refurbishing of the recovered parts and reassembly into like-new finished products. We aim at optimizing the production planning for the corresponding three-echelon system over a multi-period horizon. Production planning involves making decisions about the production level (i.e. which products and how much of them should be made), the timing (i.e. when the products should be made) and the resources to be used. Within a remanufacturing context, production planning includes making decisions on the used products returned by customers, such as how much and when used products should be disassembled, refurbished or reassembled in order to build new or like-new products. The main objective is to meet customers demand for the remanufactured products in the most cost-effective way.

Lot-sizing problems arise in production situations which involve setup operations such as tool changes, machine calibration and machine installation incurring fixed setup costs. As a naive perception, to reduce these setup costs, production should be run using large lot sizes. However, this generates desynchronized 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 the setup costs and minimizing the inventory holding costs, under constraints on customer demand satisfaction and practical limitations of the system. In the present work we investigate the problem of minimizing the setup cost and the inventory holding cost together with a penalty cost for the lost sales induced by the demand not satisfied on time within a remanufacturing environment. We thus investigate a 3-echelon lot-sizing problem with returns and lost sales.

As compared to classical manufacturing systems which produce end-products from virgin raw materials and new components, remanufacturing systems involve several complicating characteristics, among which is a high level of uncertainty in the input data needed to make planning decisions. This is mainly due to a lack of control on the return flows of used products, both in terms of quantity and quality, and to the difficulty of forecasting the demand for new (or like-new) products. Even in cases where companies apply special policies to collect the used products from customers, i.e., product life-cycle contracts or collecting incentives, these parameters remain difficult to accurately predict. The fact that production planning and control activities are more complex for remanufacturing firms due to uncertainties is extensively discussed in [2] and [3]. Lage and Filho [4] provided a recent literature review about production planning and control for remanufacturing systems and analyzed whether the gap identified by Guide [3] was fully investigated, and concluded that no work deals simultaneously with all of the complicating characteristics involved in production planning and control activities in a remanufacturing environment. In the same way, Ilgin and Gupta [5] provided a review of the state of the art in environmentally conscious manufacturing and product recovery and investigated the production planning field within a remanufacturing environment. Most works reported in [5] for planning production activities did not take into account any uncertain parameters and the authors concluded that more studies are needed to better control the effects of uncertainties in remanufacturing systems. Hence, this work is an attempt at closing this gap. Namely, we investigate a production planning model where uncertainties related to the quantity and quality of returned products, the customers demand, and the costs are simultaneously taken into account.

In recent years, stochastic lot-sizing problems either for remanufacturing systems or for hybrid manufacturing/remanufacturing systems have been studied under several modeling and uncertainty assumptions. Multi-period single-echelon single-item stochastic lot-sizing problems have been studied in [6], [7] and [8], taking under consideration both stochastic demand and returns quantity. Kilic [6] and Kilic et al. [7] included customer service level constraints and developed a heuristic approach based on a static-dynamic uncertainty strategy. Naeem et al. [8] introduced a backlogging cost to be paid whenever the demand is not satisfied on time. They proposed a stochastic dynamic programming approach to deal with the uncertain parameters. Subsequently, Macedo et al. [9] and Hilger et al. [10] studied a multi-item variant of the problem taking into account both stochastic demand and returns quantity. Macedo et al. [9] extended the previous works ([6], [7] and [8]) by considering also stochastic setup costs and proposed a two-stage stochastic programming model that assumes production and setup as first-stage decision variables and inventory, disposal, and backlogging as second-stage decision variables. Hilger et al. [10] studied the multi-item variant under capacity constraints and proposed a nonlinear model formulation that is approximated by two models. In the first approximation, the nonlinear functions of the expected values are approximated by piecewise linear functions such that the problem can be converted into a mixed-integer problem that can be solved using a standard mixed-integer programming (MIP) solver. The second approximation uses an approach based on sample averages where the random variables are represented by samples of independently generated scenarios. In contrast to the above mentioned works which considered single-echelon production systems, Wang and Huang [11] and Fang et al. [12] studied multi-period multi-echelon multi-product stochastic lot-sizing problems for remanufacturing systems comprising several operations such as disassembly, recycling and reassembly. Both works focused on stochastic demand. Wang and Huang [11] developed a two-stage stochastic programming model aiming at finding a compromise between the expected cost and the solution robustness. Fang et al. [12] proposed a multi-stage stochastic programming approach which resulted in the formulation of a large-size MILP and developed a Lagrangian relaxation-based heuristic algorithm to solve it.

We focus on a multi-echelon system with not only disassembly and reassembly operations, but also refurbishing operations. We explicitly consider uncertain input parameters and propose a multi-stage stochastic programming approach. Our work is therefore closely related to the one of Fang et al. [12]. However, these authors focused only on stochastic demand and developed a heuristic solution approach. In contrast, we consider the uncertainty on the demand, the return quantity and quality and the production costs and we aim at developing an exact solution method for the problem.

Multi-stage stochastic integer programming approaches usually rely on scenario trees to represent the uncertain information structure and result in the formulation of large-size mixed integer linear programs. One key element in efficiently solving large-size MILP to optimality is the quality of the bounds provided by the linear relaxation of the problem as it has a strong impact of the numerical efficiency of the branch-and-bound algorithm. Linear programming relaxation strengthening techniques focused on stochastic lot-sizing problems expressed on scenario trees have been studied in [13], [14], [15] and [16]. Guan et al. [13] investigated an uncapacitated lot-sizing problem and extended the (ℓ, S) valid inequalities known for the deterministic variant to a general facet-defining class called (Q, S_Q) for the stochastic variant. Later, di Summa and Wosley [15] studied a capacitated lot sizing problem and extended the work of Guan et al. [13] by proving that the (Q, S_Q) valid inequality is dominated by a mixing inequality. Additionally, Guan et al. [14] proposed a general method for generating cutting planes for multi-stage stochastic integer programs based on combining valid inequalities for the individual scenarios. Zhang et al. [16] investigated a dynamic stochastic lot-sizing problem with service level constraints and formulated the problem as a multi-stage chance-constrained program. The authors developed a branch-and-cut method for the multi-stage setting based on a set of valid inequalities obtained by a mixing procedure. Nonetheless, all these works have focused on single-echelon production systems and do not consider used product returns nor lost sales. In contrast, we investigate a multi-stage stochastic integer programming approach dealing with a multi-echelon multi-item stochastic lot-sizing problem with lost sales within a remanufacturing environment.

The contributions of the present work are threefold. Firstly, we propose a multi-stage stochastic integer programming approach for a stochastic lot-sizing problem arising in a remanufacturing context. This is in contrast with most previously published works on lot-sizing for remanufacturing which consider either a deterministic setting or develop two-stage stochastic programming approaches. Secondly, we consider a multi-echelon multi-item setting, whereas most papers dealing with a multi-stage decision process for stochastic lot-sizing problems assume either a single-echelon or a single-item setting. Finally, we propose a branch-and-cut framework to solve the resulting large-size mixed integer linear program. The algorithm relies on a new set of valid inequalities obtained by mixing previously known path inequalities [17]. These valid inequalities increase exponentially in the number of nodes in the scenario tree. We provide an efficient cutting-plane generation strategy to identify the useful subset of this class. Our computational experiments show that the proposed method is capable of significantly decreasing the computation time needed to obtain guaranteed optimal solutions.

The remaining part of this paper is organized as follows: Section 2 formally describes the problem and proposes a mixed integer linear programming model. In Section 3, a reformulation of the problem based on the echelon-stock concept is presented. This reformulation allows us to identify a series of single-echelon subproblems embedded in the general multi-echelon problem. Section 4 introduces a new class of valid inequalities to strengthen the linear relaxation of each single-echelon subproblem. Cutting-plane generation algorithms are developed in Section 5. Section 6 reports the results of computational experiments and discusses the performance of our branch-and-cut algorithm. Finally, Section 7 gives the conclusions with possible issues for further research.

2. Problem description and mathematical formulation

2.1. System description

We consider a remanufacturing system comprising three main production echelons (see Figure 1): disassembly, refurbishing and reassembly, and seek to plan the production activities in this system over a multi-period horizon. We assume that there is a single type of used product which, in each period, is returned in limited quantity by customers. These used products are first disassembled into parts. Due to the usage state of the used products, some of these parts are not recoverable and have to be discarded during disassembly. In order to reflect the variations in the quality of the used products, the yield of the disassembly process, i.e. the proportion of parts which will be recoverable, is assumed to be part-dependent and time-dependent. The remaining recoverable parts are then refurbished on dedicated refurbishing processes. The serviceable parts obtained after refurbishing are reassembled into remanufactured products which have the same bill-of-material as the used products. These remanufactured products are used to satisfy the dynamic demand of customers.

All the production processes are assumed to be uncapacitated. However, the system might not be able to satisfy the customer demand on time due to part shortages if there are not enough used products returned by customers or if their quality is low. In this situation, the corresponding demand is lost incurring a high penalty cost to account

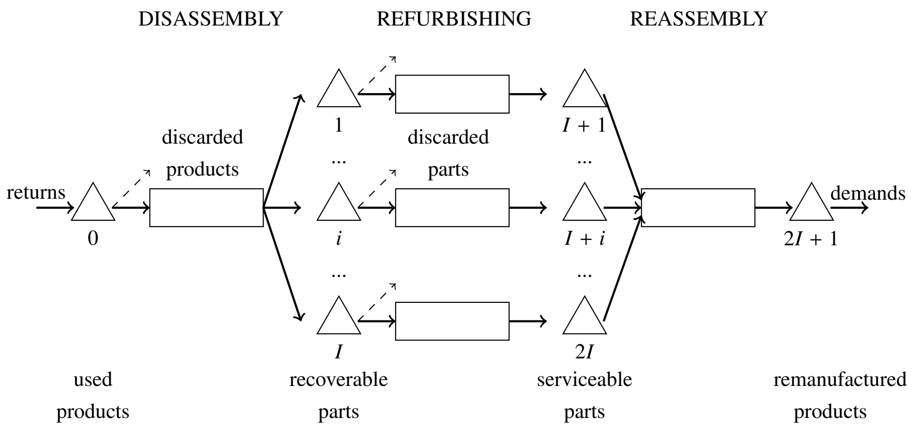

Figure 1: Illustration of studied remanufacturing system

for the loss of customer goodwill. Moreover, note that some used products are allowed to be discarded before being disassembled: this option might be useful in case more used products are returned that what is needed to satisfy the demand for remanufactured products. Similarly, some of the recoverable parts obtained from the disassembly process may be discarded. In case there is a strong unbalance between the part-dependent disassembly yields, this option might be used in a production plan to avoid an unnecessary accumulation in inventory of the easy-to-recover parts.

We aim at finding an optimal production plan, i.e. a production plan complying with all the practical limitations of the system while minimizing the total production cost. This cost comprises the production fixed setup costs to be incurred each time a production takes place on a process, the inventory holding costs for all items involved in the system, the lost-sales costs penalizing the unsatisfied demand and the disposal costs for the discarded used products and parts.

Ahn et al. [18] studied a deterministic and particular case of the problem, in which the quantity of returned products is unlimited and the lost sales and the discarding quantities are assumed to be zero. The authors proved that, under these assumptions, the problem is NP-hard. Therefore, our problem is NP-hard as well.

2.2. *Uncertainty*

As mentioned in Section 1, one of the main challenges to be faced when planning remanufacturing activities is the high level of uncertainty in the problem parameters. In what follows, we propose a production planning model in which all problem parameters, except the bill-of-material coefficients, are subject to uncertainty.

We consider a multi-stage decision process corresponding to the case where the value of the uncertain parameters unfolds little by little following a discrete-time stochastic process and the production decisions are adapted progressively as more and more information is collected. This leads to the representation of the uncertainty via a scenario tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$. Each node $n \in \mathcal{V}$ corresponds to a single planning period t belonging to a single decision stage $s \in \mathcal{S}$. It represents the state of the system 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 variables from the current node. The nodes of the scenario tree are indexed from 0 to $|\mathcal{V}| - 1$. Each node n (except root node 0) belongs to a time period $t + 1$ and has a unique predecessor node denoted a^n corresponding to the time period t. The probability associated with the state represented by the node n is ρ^n. Each non-terminal node n is a root node of a subtree $\mathcal{T}(n)$. The set $\mathcal{L}(n)$ represents the set of leaf-nodes belonging to $\mathcal{T}(n)$. The set of nodes on the path from a node n to a node μ is denoted by $\mathcal{P}(n, \mu)$. Note that we assume without loss of generality that the nodes in the sets $\mathcal{P}(\cdot)$ and $\mathcal{L}(\cdot)$ follow the same indexation as the scenario tree $\mathcal{T}$.

We use the following notations for the problem formulation:

- I: number of part types involved in one product,

- $\mathcal{I}$: set of all products involved in the system, $\mathcal{I} = \{0, ..., 2I+1\}$, where $i = 0$ corresponds to returned product and $i = 2I + 1$ corresponds to remanufactured product,
- $\mathcal{I}_r$: set of recoverable parts provided by the disassembly process, $\mathcal{I}_r = \{1, ..., I\}$,
- $\mathcal{I}_s$: set of serviceable parts provided by the refurbishing processes, $\mathcal{I}_s = \{I+1, ..., 2I\}$,
- $\mathcal{J}$: set of production processes, $\mathcal{J} = \{0, ..., I+1\}$, where $p = 0$ corresponds to the disassembly process, $p = 1, ..., I$ correspond to the refurbishing processes and $p = I+1$ corresponds to the reassembly process.

The deterministic parameter is:

- α_i: number of parts i embedded in a returned/remanufactured product,

The stochastic parameters are introduced as follows:

- r^n: quantity of used products (returns) collected at node $n \in \mathcal{V}$,
- d^n: customers demand at node $n \in \mathcal{V}$,
- π_i^n: proportion of recoverable parts $i \in \mathcal{I}_r$ obtained by disassembling one unit of returned product at node $n \in \mathcal{V}$,
- l^n: unit lost-sales penalty cost at node $n \in \mathcal{V}$,
- f_p^n: setup cost for process $p \in \mathcal{J}$ at node $n \in \mathcal{V}$,
- h_i^n: unit inventory cost for part $i \in \mathcal{I}$ at node $n \in \mathcal{V}$,
- q_i^n: unit cost for discarding a recoverable part or a returned product $i \in \mathcal{I}_r \cup \{0\}$ at node $n \in \mathcal{V}$,
- g^n: unit cost for discarding the unrecoverable parts obtained while disassembling one unit of returned product at node $n \in \mathcal{V}$.

Note that due to the unknown quality of the returned product, there exists an implicit flow of unrecoverable parts generated when disassembling used products. We thus introduce $g^n = \sum_{i=1}^{I} q_i^n (1 - \pi_i^n)\alpha_i$ which represents the unit cost of the parts that cannot be recovered when a returned product is disassembled. Moreover, we assume that at each stage, the realization of the random parameters happens before we have to make a decision for this stage, i.e. we assume that the values of r^n, d^n, π_i^n, l^n, f_p^n, h_p^n, q^n and g^n are known before we have to decide on the production plan at node $n \in \mathcal{V}$. We also assume that $l^n \gg g^n$ for all $n \in \mathcal{V}$.

2.3. MILP formulation

We propose a multi-stage stochastic integer programming model based on the uncertainty representation described above. The decision variables involved in the model are:

- X_p^n: quantity of parts processed on process $p \in \mathcal{J}$ at node $n \in \mathcal{V}$,
- $Y_p^n \in \{0, 1\}$: setup variable for the process $p \in \mathcal{J}$ at node $n \in \mathcal{V}$,
- S_i^n: inventory level of part $i \in \mathcal{I}$ at node $n \in \mathcal{V}$,
- Q_i^n: quantity of part $i \in \mathcal{I}_r \cup \{0\}$ discarded at node $n \in \mathcal{V}$,
- L^n: lost sales of remanufactured products at node $n \in \mathcal{V}$.

The mixed integer linear programming model is given below.

$$Z^* = min \sum_{n \in \mathcal{V}} \rho^n \Big(\sum_{p \in \mathcal{J}} f_p^n Y_p^n + \sum_{i \in \mathcal{I}} h_i^n S_i^n + l^n L^n + \sum_{i \in \mathcal{I}_r \cup \{0\}} q_i^n Q_i^n + g^n X_0^n \Big) \tag{1}$$

subject to

$$X_p^n \leq M_p^n Y_p^n \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \tag{2}$$
$$S_0^n = S_0^{a^n} + r^n - X_0^n - Q_0^n \qquad \forall n \in \mathcal{V} \tag{3}$$
$$S_i^n = S_i^{a^n} + \pi_i^n \alpha_i X_0^n - X_i^n - Q_i^n \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} \tag{4}$$
$$S_i^n = S_i^{a^n} + X_{i-I}^n - \alpha_i X_{I+1}^n \qquad \forall i \in I_s, \forall n \in \mathcal{V} \tag{5}$$
$$S_{2I+1}^n = S_{2I+1}^{a^n} + X_{I+1}^n - d^n + L^n \qquad \forall n \in \mathcal{V} \tag{6}$$
$$S_0^0 = r_0^0 - X_0^0 - Q_0^0 \tag{7}$$
$$S_i^0 = \pi_i^0 \alpha_i X_0^0 - Q_i^0 \qquad \forall i \in \mathcal{I}_r \tag{8}$$
$$S_i^0 = X_{i-I}^0 - \alpha_{i-I} X_{I+1}^0 \qquad \forall i \in \mathcal{I}_s \tag{9}$$
$$S_{2I+1}^0 = X_{I+1}^0 - d^0 + L^0 \tag{10}$$
$$S_i^n \geq 0 \qquad \forall i \in \mathcal{I}, \forall n \in \mathcal{V} \tag{11}$$
$$L^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{12}$$
$$X_p^n \geq 0, Y_p^n \in \{0, 1\} \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \tag{13}$$

The objective function (1) aims at minimizing the expected total cost, over all nodes of the scenario tree. This cost is the sum of the expected setup, inventory holding, lost sales and disposal costs. Constraints (2) link the production quantity variables to the setup variables. Constraints (3)-(10) are the inventory balance constraints. Constraints (3) (resp. (4) and (5)) involve a term corresponding to a dependent demand X_0^n (resp. X_i^n and $\alpha_i X_{I+1}^n$) whereas Constraints (6) only involve an independent demand term d^n. Without loss of generality, we assume that the initial inventories are all set to 0. Finally, Constraints (11)-(13) provide the domain of the decision variables.

The value of M_p^n can be set by using an upper bound on the quantity that can be processed on process p at node n. This quantity is limited by two elements: the availability of the used products already returned by customers and the future demand for remanufactured products. Thus, for a given process p and a node n, M_p^n is computed as the minimum between:

- A value provided by the maximum amount of input product (used product, recoverable part or serviceable part) that can be available for processing on process p at node n. This value is computed by summing the values of r^v on the nodes v belonging to the path from the root node to the node n,
- A value provided by the maximum demand for the output product (recoverable part, serviceable part or remanufactured product) of process p at node n. This value is computed by considering the maximum future demand for the output product over the set $\mathcal{P}(n, \lambda)$. It is the maximum, over all leaf nodes λ in $\mathcal{L}(n)$, of the cumulated demand on the path from the node n to the leaf node λ.

This leads to the following expressions for constants M_p^n:

- $M_0^n = \min\left\{ \sum_{v \in \mathcal{P}(0,n)} r^v, \max_{\lambda \in \mathcal{L}(n)} \left\{ \frac{\sum_{v \in \mathcal{P}(n,\lambda)} d^v}{min_{i=1...I} \pi_i^n} \right\} \right\}$
- $M_p^n = \min\left\{ \sum_{v \in \mathcal{P}(0,n)} (\alpha_p r^v \max_{\mu \in \mathcal{P}(v,n)} \pi_p^\mu), \max_{\lambda \in \mathcal{L}(n)} \left\{ \sum_{v \in \mathcal{P}(n,\lambda)} \alpha_p d^v \right\} \right\}$, for $p = 1, ..., I$.
- $M_{I+1}^n = \min\left\{ \sum_{v \in \mathcal{P}(0,n)} r^v, \max_{\lambda \in \mathcal{L}(n)} \left\{ \sum_{v \in \mathcal{P}(n,\lambda)} d^v \right\} \right\}$

Even if the problem (1)-(13) is a mixed-integer linear program displaying a structure similar to the one of its deterministic counterpart, its resolution by a mathematical programming solver poses some computational difficulties in practice. Namely, the problem size is basically proportional to the number of nodes in the scenario tree, which is significantly larger than the number of periods of the planning horizon. Moreover, the presence of the big-M type constraints (2) leads to a poor quality of the lower bounds provided by the linear relaxation of the problem.

In what follows, we propose a branch-and-cut algorithm in order to solve to optimality medium-size instances of the problem. We first provide a reformulation of the problem that enables to decompose the multi-echelon problem into a series of single-echelon subproblems. We then investigate two sets of valid inequalities (path inequalities and tree inequalities) that can be used to strengthen the formulation of each of these single-echelon subproblems. These valid inequalities are added to the problem formulation using a cutting-plane strategy during the course of the branch-and-bound search.

3. Mathematical reformulation

The concept of echelon stock has been widely used to develop solution approaches for multi-echelon lot-sizing problems (the reader is referred to [19] for further details). The main advantages of the reformulation is that it helps decomposing the multi-echelon problem into a series of single-echelon lot-sizing problems for which formulation strengthening techniques such as valid inequalities or extended reformulations are available. As each subproblem is a relaxed version of the overall multi-echelon problem, valid inequalities strengthening the linear relaxation of each subproblem will strengthen the linear relaxation of the overall multi-echelon problem.

3.1. Echelon stock reformulation

The echelon stock of a product in a multi-echelon production system corresponds to the total quantity of the product held in inventory, either as such or as a component of its bill-of-material. For each product $i \in \{1, ..., 2I+1\}$ we define the echelon inventory variables as follows:

- $E_i^n = S_i^n + E_{I+i}^n = S_i^n + S_{I+i}^n + \alpha_i S_{2I+1}^n$, for $i \in \mathcal{I}_r$, for $n \in \mathcal{V}$
- $E_i^n = S_i^n + \alpha_i E_{2I+1}^n = S_i^n + \alpha_i S_{2I+1}^n$, for $i \in \mathcal{I}_s$, for $n \in \mathcal{V}$
- $E_{2I+1}^n = S_{2I+1}^n$, for $n \in \mathcal{V}$

Moreover, we define the unit echelon inventory holding cost eh_i^n as follows:

- $eh_i^n = h_i^n$, for $i \in \mathcal{I}_s$, for $n \in \mathcal{V}$
- $eh_i^n = h_i^n - h_{i-I}^n$, for $i \in \mathcal{I}_r$, for $n \in \mathcal{V}$
- $eh_{2I+1}^n = h_{2I+1}^n - \sum_{i \in \mathcal{I}_r} \alpha_i h_i^n$, for $n \in \mathcal{V}$

Note that we do not define an echelon inventory variable for the used product $i = 0$ as the presence of node-varying values of the disassembly yield π makes it difficult to properly define an independent demand for this product. This leads to the following mixed-integer linear programming formulation:

$$Z^* = min \sum_{n \in \mathcal{V}} \rho^n \Big(\sum_{p \in \mathcal{J}} f_p^n Y_p^n + h_0^n S_0^n + \sum_{i \in \mathcal{I} \setminus \{0\}} eh_i^n E_i^n l^n L^n + \sum_{i \in \mathcal{I}_r \cup \{0\}} q_i^n Q_i^n + g^n X_0^n \Big) \tag{14}$$

subject to:

$$X_p^n \leq M_p^n Y_p^n \quad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \tag{15}$$
$$S_0^n = S_0^{a^n} + r^n - X_0^n - Q_0^n \quad \forall n \in \mathcal{V} \tag{16}$$
$$E_i^n = E_i^{a^n} + \pi_i^n \alpha_i X_0^n - \alpha_i d^n + \alpha_i L^n - Q_i^n \quad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} \tag{17}$$
$$E_i^n = E_i^{a^n} + X_{i-I}^n - \alpha_i d^n + \alpha_i L^n \quad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{V} \tag{18}$$
$$E_{2I+1}^n = E_{2I+1}^{a^n} + X_{I+1}^n - d^n + L^n \quad \forall n \in \mathcal{V} \tag{19}$$
$$S_0^0 = r_0^0 - X_0^0 - Q_0^0 \tag{20}$$
$$E_i^0 = \pi_i^0 \alpha_i X_0^0 - \alpha_i d^0 + \alpha_i L^0 - Q_i^0 \quad \forall i \in \mathcal{I}_r \tag{21}$$
$$E_i^0 = X_{i-I}^0 - \alpha_{i-I} d^0 + \alpha_{i-I} L^0 \quad \forall i \in \mathcal{I}_s \tag{22}$$
$$E_{2I+1}^0 = X_{I+1}^0 - d^0 + L^0 \tag{23}$$
$$E_i^n - E_{I+i}^n \geq 0 \quad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} \tag{24}$$
$$E_i^n - \alpha_i E_{2I+1}^n \geq 0 \quad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{V} \tag{25}$$
$$E_{2I+1}^n \geq 0 \quad \forall n \in \mathcal{V} \tag{26}$$
$$S_0^n \geq 0, L^n \geq 0 \quad \forall n \in \mathcal{V} \tag{27}$$
$$X_p^n \geq 0, Y_p^n \in \{0,1\} \quad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \tag{28}$$

As in the previous formulation, the objective function (14) aims at minimizing the expected cost, over all nodes of the scenario tree. Constraints (15) are defined as Constraints (2) of the (1)-(13) formulation. Constraints (16)-(20) are inventory balance constraints. Constraints (16) use the classical inventory variables, whereas Constraints (17)-(19) make use of the echelon inventory variables. Contrary to Constraints (4)-(5) of the natural formulation, Constraints (17)-(19) do not involve a dependent demand term but only an external demand term. Constraints (24)-(26) ensure consistency between the echelon inventory at the different levels of the bill-of-material and guarantee that the physical inventory of each product remains non-negative. Finally, Constraints (12)-(28) define the domain of the decision variables.

3.2. Single echelon subproblems

The introduction of echelon inventory variables leads to the elimination of dependent demand in the inventory balance equations of the (1)-(13) formulation. This induces that the constraint matrix of (14)-(28) displays a specific structure: it can be decomposed in a series of single-echelon single-resource lot-sizing subproblems coupled by the linking constraints (16), (24)-(26). The single-echelon subproblems are defined as follows.

For each refurbishing/reassembly process p, we have the following subproblem:

$$Z_p^* = min \sum_{n \in \mathcal{V}} \rho^n \Big(f_p^n Y_p^n + eh_{p+I}^n E_{p+I}^n + l^n L^n \Big) \tag{29}$$

subject to:

$$X_p^n \leq M_p^n Y_p^n \quad \forall n \in \mathcal{V} \tag{30}$$
$$E_{p+I}^n = E_{p+I}^{a^n} + X_p^n - \alpha_p d^n + \alpha_p L^n \quad \forall n \in \mathcal{V} \tag{31}$$
$$E_{p+I}^0 = X_p^0 - \alpha_p d^0 + \alpha_p L^0 \tag{32}$$
$$E_{p+I}^n \geq 0 \quad \forall n \in \mathcal{V} \tag{33}$$
$$L^n \geq 0, X_p^n \geq 0, Y_p^n \in \{0,1\} \quad \forall n \in \mathcal{V} \tag{34}$$

For the disassembly process, for each item $i \in \mathcal{I}_r$, we have the following subproblem:

$$Z_0^* = \min \sum_{n \in \mathcal{V}} \rho^n \Big(f_0^n Y_0^n + \sum_{i=1}^{I} eh_i^n E_i^n + \sum_{i=1}^{I} q_i^n Q_i^n + l^n L^n + g^n X_0^n \Big) \tag{35}$$

subject to:

$$X_0^n \leq M_0^n Y_0^n \qquad \forall n \in \mathcal{V} \tag{36}$$

$$E_i^n = E_i^{a^n} + \pi_i^n \alpha_i X_0^n - \alpha_i d^n + \alpha_i L^n - Q_i^n \qquad \forall n \in \mathcal{V} \tag{37}$$

$$E_i^0 = \pi_i^0 \alpha_i X_0^0 - \alpha_i d^0 + \alpha_i L^0 - Q_i^0 \tag{38}$$

$$E_i^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{39}$$

$$L^n \geq 0, X_0^n \geq 0, Q_i^n \geq 0, Y_0^n \in \{0, 1\} \qquad \forall n \in \mathcal{V} \tag{40}$$

Each subproblem (29)-(34) or (35)-(40) is an uncapacitated single-echelon single-item lot-sizing problem with lost sales. The deterministic variant of this problem was studied by Loparic et al. [17] who proposed a family of valid inequalities called (k, U) inequalities, to strengthen the linear relaxation. We discuss in Section 4 how these inequalities known for the deterministic variant of the problem can be used to solve the stochastic problem expressed on a scenario tree.

4. Valid inequalities

In this section, we provide (k, U) inequalities for each single-echelon subproblem described in Section 3. We first exploit these (k, U) inequalities considering their application to each individual scenario, i.e. to each individual path from a non-terminal node n to a leaf node $\lambda \in \mathcal{L}(n)$ in the scenario tree $\mathcal{T}$. Next, we extend them to a more general class of inequalities. This is done by exploiting the scheme proposed by Guan et al. [14] for generic multi-stage stochastic integer programs. The idea is to mix valid inequalities corresponding to different individual scenarios to obtain valid inequalities for the whole scenario tree (or for a subtree). Throughout this section we will refer to a (k, U) inequality applied to an individual scenario as a *path inequality* and to a (k, U) inequality applied to a subtree as a *tree inequality*.

4.1. Path inequalities

We first recall the relevant notation introduced in Section 2. Each node k of the scenario tree $\mathcal{T}$, except for the root node, has a unique parent, and each non-terminal node k is the root of a subtree $\mathcal{T}(k)$, with $\mathcal{T}(0) = \mathcal{T}$. Let $\mathcal{L}(k)$ be the set of leaf nodes such that there exists a path from the node $k \in \mathcal{V}$ to the leaf node and let c_k^λ be the immediate successor of node k belonging to the set $\mathcal{P}(k, \lambda)$, for $\lambda \in \mathcal{L}(k)$. Let $U_{k,\lambda} \subseteq \mathcal{P}(c_k^\lambda, \lambda)$ be a subset of nodes belonging to the path from the node c_k^λ to the leaf node λ.

For each process $p \in \{1...I + 1\}$, we have the following proposition:

Proposition 1. *Let $k \in \mathcal{V}$ and $\lambda \in \mathcal{L}(k)$. Let $U_{k,\lambda} \subset \mathcal{P}(c_k^\lambda, \lambda)$. The following (k, U) inequality*

$$E_{p+I}^k \geq \alpha_p \sum_{v \in U_{k,\lambda}} \Big[d^v \Big(1 - \sum_{\mu \in \mathcal{P}(c_k^v, v)} Y_p^v \Big) - L^v \Big] \tag{41}$$

is valid for the problem (14)-(28).

The proof is direct following the proof in [17].

The intuition underlying path inequalities can be understood as follows. We consider the inventory level of the product $p + I$ of node k and look for the future demands for this product in the path originated from the node k to the leaf node λ. For a node $v \in U_{k,\lambda}$, if $\sum_{\mu \in \mathcal{P}(c_k^\lambda, v)} Y_p^v \geq 1$, the demand of node v, $\alpha_p d^v$, can be satisfied by a production in one of the nodes $\mu \in \mathcal{P}(c_k^v, v)$ and does not have to be in stock at the node k. But if $\sum_{\mu \in \mathcal{P}(c_k^v, v)} Y_p^v = 0$, the demand $\alpha_p d^v$ cannot be produced in any node $\mu \in \mathcal{P}(c_k^v, v)$ meaning that the portion of this demand which will be satisfied by a production $\alpha_p(d^v - L^v)$, should already be in stock at node k.

Moreover, for process $p = 0$ and each part $i \in \mathcal{I}_r$, we also have a path inequality defined as follows:

$$E_i^k \geq \alpha_i \sum_{v \in U_{k,\lambda}} \Big[d^v \Big(1 - \sum_{\mu \in \mathcal{P}(c_k, v)} Y_0^v \Big) - L^v \Big]$$

We note that the right-hand side of this inequality has the same expression for each $i \in \mathcal{I}_r$. In order to exploit this fact and limit the number of valid inequalities to be investigated, we consider only the inequality corresponding to the item $i \in \mathcal{I}_r$ for which the value E_i^k / α_i is minimum. This leads to the following proposition.

Proposition 2. *Let $k \in \mathcal{V}$ and $\lambda \in \mathcal{L}(k)$. Let $U_{k,\lambda} \subset \mathcal{P}(c_k^\lambda, \lambda)$. The following (k, U) inequality*

$$\min_{i \in \mathcal{I}_r} \Big[\frac{E_i^k}{\alpha_i}\Big] \geq \sum_{\nu \in U_{k,\lambda}} \Big[d^\nu \Big(1 - \sum_{\mu \in \mathcal{P}(c_k,\nu)} Y_0^\nu\Big) - L^\nu\Big] \tag{42}$$

is valid for the problem (14)-(28).

4.2. Tree inequalities

Now, we investigate a new family of valid inequalities obtained by considering a subtree of the scenario tree as proposed by Guan et al. in [13] and [14]. The authors proposed a general scheme to obtain valid inequalities for multi-stage stochastic integer programs by mixing several path inequalities. In what follows, we apply this scheme to derive a new set of tree inequalities based on a mixing of the path inequalities discussed above. We first introduce additional notations to properly define this new set of valid inequalities. Let $\mathcal{V}(k)$ be the subset of nodes belonging to the subtree $\mathcal{T}(k)$ and $U = \cup_{\lambda \in \mathcal{L}(k)} U_{k,\lambda}$ be a set of nodes defining a tree inequality. This enables us to introduce the following proposition for each process $p \in \{1, ..., I+1\}$.

Proposition 3. *Let $k \in \mathcal{V}$ and $U \subset \mathcal{V}(k)$. Let $\sigma = \{\sigma_1, ..., \sigma_{|\mathcal{L}(k)|}\}$ be a sequence of leaf nodes belonging to $\mathcal{L}(k)$ in increasing order of cumulative demand $\sum_{\nu \in U_{k,\lambda}} d^\nu$: $\sum_{\nu \in U_{k,\sigma_1}} d^\nu \leq ... \leq \sum_{\nu \in U_{k,\sigma_l}} d^\nu \leq ... \leq \sum_{\nu \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^\nu$. We set $\sum_{\nu \in U_{k,\sigma_0}} d^\nu = 0$. The following inequality*

$$E_{p+I}^k + \alpha_p \sum_{\nu \in U} L^\nu + \alpha_p \sum_{\mu \in \mathcal{V}(k) \backslash \{k\}} \phi^\mu Y_p^\mu \geq \alpha_p \sum_{\nu \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^\nu \tag{43}$$

is valid for problem (14)-(28), with

$$\phi^\mu = \min\Big\{ \max_{\lambda \in \mathcal{L}(\mu)} \{ \sum_{\nu \in U_{a^\mu,\lambda}} d^\nu\}, \sum_{l=1...|\mathcal{L}(k)| \ s.t \ \sigma_l \in \mathcal{L}(\mu)} \Big(\sum_{\nu \in U_{k,\sigma_l}} d^\nu - \sum_{\nu \in U_{k,\sigma_{l-1}}} d^\nu\Big)\Big\}$$

Proof. Without loss of generality, we drop the index of the production process and assume $\alpha_p = 1$, for all $p \in \{1, ..., I+1\}$. Let k be a non-leaf node. For each leaf node $\lambda \in \mathcal{L}(k)$, we arbitrarily choose a subset of nodes $U_{k,\lambda} \subset \mathcal{P}(c_k^\lambda, \lambda)$. We thus obtain a set of $|\mathcal{L}(k)|$ path inequalities defined as follows:

$$E^k \geq \sum_{\nu \in U_{k,\lambda}} \Big[d^\nu(1 - \sum_{\mu \in \mathcal{P}(c_k^\lambda,\nu)} Y^\mu) - L^\nu\Big] \tag{44}$$

We rewrite the inequalities (44) in a form making it easier to apply Theorem 2 in [14].

$$E^k + \sum_{\nu \in U_{k,\lambda}} L^\nu + \sum_{\mu \in \mathcal{P}(c_k^\lambda,\lambda)} (\sum_{\nu \in U_{a^\mu,\lambda}} d^\nu) Y^\mu \geq \sum_{\nu \in U_{k,\lambda}} d^\nu$$

Let $\sigma = \{\sigma_1, ..., \sigma_{|\mathcal{L}(k)|}\}$ be a sequence of leaf nodes in increasing order of cumulative demand $\sum_{\nu \in U_{k,\lambda}} d^\nu$: $\sum_{\nu \in U_{k,\sigma_1}} d^\nu \leq ... \leq \sum_{\nu \in U_{k,\sigma_l}} d^\nu \leq ... \leq \sum_{\nu \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^\nu$. We set $\sum_{\nu \in U_{k,\sigma_0}} d^\nu = 0$. Now, we use Theorem 2 in [14] to combine these $|\mathcal{L}(k)|$ path inequalities and derive a new tree inequality as follows:

$$E^k + \sum_{\nu \in \cup_{\lambda \in \mathcal{L}(k)} U_{k,\lambda}} L^\nu + \sum_{\mu \in \mathcal{V} \backslash \{k\}} \phi^\mu Y^\mu \geq (\sum_{\nu \in U_{k,|\mathcal{L}(k)|}} d^\nu)$$

where, the coefficient ϕ^μ for $\mu \in \mathcal{V} \setminus \{k\}$, is given by:

$$\phi^\mu = \min\Big\{ \max_{\lambda \in \mathcal{L}(\mu)} \{ \sum_{\nu \in U_{a^\mu,\lambda}} d^\nu\}, \sum_{\lambda \in \mathcal{L}(\mu)} \Big(\sum_{\nu \in U_{k,\lambda}} d^\nu - \sum_{\nu \in U_{k,\lambda-1}} d^\nu\Big)\Big\}$$

with $\sum_{\nu \in U_0} d^\nu$ set to 0. □

The same scheme can be applied starting with the path inequalities (42) for the disassembly process $p = 0$. This leads to the following proposition:

Proposition 4. *Let $k \in \mathcal{V}$ and $U \subset \mathcal{V}(k)$. Let $\sigma = \{\sigma_1, ..., \sigma_{|\mathcal{L}(k)|}\}$ be a sequence of leaf nodes belonging to $\mathcal{L}(k)$ in the increasing order of the cumulative demand $\sum_{\nu \in U_{k,\lambda}} d^\nu$: $\sum_{\nu \in U_{k,\sigma_0}} d^\nu \le \sum_{\nu \in U_{k,\sigma_1}} d^\nu \le ... \le \sum_{\nu \in U_{k,\sigma_l}} d^\nu \le ... \le \sum_{\nu \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^\nu$. We set $\sum_{\nu \in U_{k,\sigma_0}} d^\nu = 0$. The following inequality*

$$\min_{i \in \mathcal{I}_r} \Big[\frac{E_i^k}{\alpha_i}\Big] + \sum_{\nu \in U} L^\nu + \sum_{\mu \in \mathcal{V}(k) \setminus \{k\}} \phi^\mu Y_0^\mu \ge \sum_{\nu \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^\nu \tag{45}$$

is valid for problem (14)-(28), with

$$\phi^\mu = \min\Big\{ \max_{\lambda \in \mathcal{L}(\mu)} \{ \sum_{\nu \in U_{a^\mu,\lambda}} d^\nu \}, \sum_{l=1...|\mathcal{L}(k)| \text{ s.t } \sigma_l \in \mathcal{L}(\mu)} \Big(\sum_{\nu \in U_{k,\sigma_l}} d^\nu - \sum_{\nu \in U_{k,\sigma_{l-1}}} d^\nu \Big)\Big\}$$

5. Cutting-plane generation

The number of valid inequalities (41), (42), (43) and (45) is too large to allow adding all of them a priori to the formulation. Hence, a cutting-plane generation strategy is needed to add only a subset of these valid inequalities into the MILP formulation. Consequently, the corresponding separation problems must be solved in order to identify which inequalities have to be incorporated in the formulation. In what follows, we discuss an exact separation algorithm for the path inequalities and a heuristic one for the tree inequalities. These separation algorithms will be used within a cutting-plane generation procedure aiming at strengthening the linear relaxation of the problem (14)-(28).

5.1. Separation algorithm for path inequalities

Given a solution $(\tilde{Y}, \tilde{L})$ of the linear relaxation of the problem, solving the separation problem for path inequalities consists in finding the most violated inequality (41)-(42) if it exists or prove that no such inequality exists. For a given process p, node $k \in \mathcal{V}$ and leaf node $\lambda \in \mathcal{L}(k)$, finding the most violated inequality corresponds to identifying the set $U_{k,\lambda}$ maximizing the right-hand side of the inequality. We note that the value of the term corresponding to a node $\nu \in U_{k,\lambda}$ in the right-hand side of (41)-(42) does not depend on the other nodes belonging to $U_{k,\lambda}$. Hence, each node of $\mathcal{P}(c_k^\lambda, \lambda)$ can be considered individually: if it has a positive contribution in maximizing the right-hand side value of the inequality, we add it to set $U_{k,\lambda}$, if not, it is discarded. This leads to the following exact separation algorithm for inequalities (41)-(42):

For a given process p, node $k \in \mathcal{V}$ and leaf node $\lambda \in \mathcal{L}(k)$, the set $U_{k,\lambda}$ is built by adding all the nodes in the set $\mathcal{P}(c_k^\lambda, \lambda)$, which satisfy the following inequality:

$$\alpha_p \Big[d^\nu \Big(1 - \sum_{\mu \in P(c_k^\nu, \nu)} \tilde{Y}_p^\mu \Big) - \tilde{L}^\nu \Big] > 0$$

We underline that the above strategy can be implemented in polynomial time [17], namely, the running-time of the proposed separation algorithm is $O(P^2)$, where P corresponds to the number of nodes in the set $\mathcal{P}(c_k^\lambda, \lambda)$.

5.2. Cutting-plane generation for path inequalities

Our preliminary computational experiments showed that adding all the violated inequalities of class (41)-(42) found at each iteration of the cutting-plane generation led to the introduction of a large number of additional constraints in the problem formulation. Moreover, many of these inequalities involved the same subsets of setup variables Y_p^n and had thus similar effects in terms of strengthening the relaxation of the problem.

In order to limit the increase in the formulation size, we propose the following cutting plane generation strategy to add violated path inequalities to the formulation. This strategy relies on two main ideas.

The first idea consists in adding, for a given process p and node $k \in \mathcal{V}$, at most one valid inequality at each iteration of the cutting-plane generation, namely the inequality corresponding to the leaf node $\lambda \in \mathcal{L}(k)$ providing the largest violation of the path inequality, i.e. to the leaf node $\lambda_{min} = argmin_{\lambda \in \mathcal{L}(k)} \tilde{E}_{p+I}^k - \alpha_p \sum_{\nu \in U_{k,\lambda}} \Big[d^\nu \Big(1 - \sum_{\mu \in \mathcal{P}(c_k^\lambda, \nu)} \tilde{Y}_p^\mu \Big) - \tilde{L}^\nu \Big]$.

The second idea aims at avoiding the addition of inequalities involving similar subsets of setup variables Y_p^n, during an iteration of the cutting-plane generation algorithm. This is achieved by using the following strategy. During a given iteration of the algorithm, each time a violated inequality is added to the formulation, we record ν_{min}, the last node of the path $\mathcal{P}(c_k^{\lambda_{min}}, \lambda_{min})$ added to the set $U_{k,\lambda_{min}}$. The inequality added to the formulation involves a subset of setup variables Y_p^n corresponding to nodes n belonging to the subpath $\mathcal{P}(c_k^{\lambda_{min}}, \nu_{min})$. As the valid inequalities generated when considering the leaf node λ_{min} at nodes $n \in \mathcal{P}(c_k^{\lambda_{min}}, \nu_{min})$ are likely to involve the same setup variables Y_p^n and have a redundant effect on the formulation strengthening, we do not consider generating them during the current iteration and thus temporarily remove the leaf node λ_{min} from the leaf node set of nodes $n \in \mathcal{P}(k, \nu_{min})$.

Note that this cutting-plane generation strategy implies that all valid inequalities are still potentially considered for inclusion in the formulation and that the separation problem is solved exactly.

The cutting-plane generation algorithm is summarized as follows:

Algorithm 1: Cutting-plane generation for path inequalities

```
Data: instances parameters and linear relaxation solution $(\tilde{E}, \tilde{Y}, \tilde{L})$
Result: set of path inequalities $S_{path}$
1  Initialize $S_{path} \leftarrow \emptyset$
2  foreach $p \in \mathcal{J}$ do
3      $\mathcal{L}'(\cdot) \leftarrow \mathcal{L}(\cdot)$
4      foreach $k \in \mathcal{V}$ do
5          $viol_{min} \leftarrow 0$
6          foreach $\lambda \in \mathcal{L}'(k)$ do
7              $U_{k,\lambda} \leftarrow \emptyset$, $\nu_{last} \leftarrow k$
8              foreach $\nu \in \mathcal{P}(c_k^{\lambda}, \lambda)$ do
9                  if $\left[d^{\nu}\left(1 - \sum_{\mu \in \mathcal{P}(c_k^{\lambda}, \nu)} \tilde{Y}_p^{\mu}\right) - \tilde{L}^{\nu}\right] > 0$ then
10                     $U_{k,\lambda} \leftarrow U_{k,\lambda} \cup \{\nu\}$, $\nu_{last} \leftarrow \nu$
11                 end
12             end
13             $viol \leftarrow \tilde{E}_{p+I}^{k} - \alpha_p \sum_{\nu \in U_{k,\lambda}} \left[d^{\nu}\left(1 - \sum_{\mu \in \mathcal{P}(c_k^{\lambda}, \nu)} \tilde{Y}_p^{\mu}\right) - \tilde{L}^{\nu}\right]$
14             if $viol < viol_{min}$ then
15                 $viol_{min} \leftarrow viol$, $\lambda_{min} \leftarrow \lambda$, $\nu_{min} \leftarrow \nu_{last}$
16             end
17         end
18         if $viol_{min} < 0$ then
19             $S_{path} \leftarrow S_{path} \cup \left\{E_{p+I}^{k} > \alpha_p \sum_{\nu \in U_{k,\lambda_{min}}} [d^{\nu}(1 - \sum_{\mu \in \mathcal{P}(c_k^{\lambda}, \nu)} Y_p^{\mu}) - L^{\nu}]\right\}$
20             foreach $\nu \in \mathcal{P}(c_k^{\lambda_{min}}, \nu_{min})$ do
21                 $\mathcal{L}'(\nu) \leftarrow \mathcal{L}'(\nu) \setminus \{\lambda_{min}\}$
22             end
23         end
24     end
25 end
```

5.3. Separation algorithm for tree inequalities

Solving the separation problem for inequalities (43) and (45) given a non-leaf node k requires identifying which nodes of $\mathcal{V}(k)$ should be selected in the set U in order to minimize the difference between the left-hand side and the right-hand side of the inequality. This is challenging as contrary to the case of path inequalities, it is not possible to consider each node of $\mathcal{V}(k)$ individually. Namely, selecting a node ν of $\mathcal{V}(k)$ in the set U not only changes the left-hand side of the inequality by a quantity $L^{\nu} + \phi^{\nu} Y^{\nu}$ but also potentially impacts the value of the coefficient ϕ^{μ} for all other nodes in $\mathcal{V}(k) \setminus \{k\}$. In addition, selecting a node ν of $\mathcal{V}(k)$ potentially changes the order of the sequence σ and hence the value of the right-hand side of the inequality. These interactions significantly complicate the resolution of the separation problem. Therefore, we consider a heuristic separation approach based on a neighborhood search to solve the separation problem corresponding to the tree inequalities. The separation algorithm is summarized as follows:

Algorithm 2: Cutting-plane generation for tree inequalities

```
Data: instances parameters and linear relaxation solution $(\tilde{E}, \tilde{Y}, \tilde{L})$
Result: set of tree inequalities $S_{tree}$
Initialize $S_{tree} \leftarrow \emptyset$, $\mathcal{V}' := \{v \in \mathcal{V} : |\mathcal{E}(v)| > 1\}$
foreach $p \in \mathcal{J}$ do
    foreach $k \in \mathcal{V}'$ do
        $U \leftarrow \emptyset$, $viol_{curr} \leftarrow \infty$
        foreach $v \in \mathcal{V}(k)$ do
            if $\left[d^v\left(1 - \sum_{\mu \in \mathcal{P}(c_k^\lambda, v)} \tilde{Y}_p^\mu\right) - \tilde{L}^v\right] > 0$ then
                $U \leftarrow U \cup \{v\}$
            end
        end
        if $U = \emptyset$ then
            break;
        else
            Compute $\phi^\mu$ for every $\mu \in \mathcal{V}(k) \setminus \{k\}$ and update the sequence $\sigma$ for the set $U$
            $viol_{min} \leftarrow \tilde{E}_{p+I}^k + \alpha_p \sum_{v \in U} \tilde{L}^v + \alpha_p \sum_{\mu \in \mathcal{V}(k) \setminus \{k\}} \phi^\mu \tilde{Y}_p^\mu - \alpha_p \sum_{v \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^v$
            while $viol_{min} < viol_{curr}$ do
                $viol_{curr} \leftarrow viol_{min}$
                foreach $v \in U$ do
                    $U' \leftarrow U \setminus \{v\}$
                    Compute $\phi^\mu$ for every $\mu \in \mathcal{V}(k) \setminus \{k\}$ and update the sequence $\sigma$ for the set $U'$
                    $viol \leftarrow \tilde{E}_{p+I}^k + \alpha_p \sum_{v \in U'} \tilde{L}^v + \alpha_p \sum_{\mu \in \mathcal{V}(k) \setminus \{k\}} \phi^\mu \tilde{Y}_p^\mu - \alpha_p \sum_{v \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^v$
                    if $viol < viol_{min}$ then
                        $viol_{min} \leftarrow viol$, $U \leftarrow U'$
                    end
                end
            end
            if $viol_{min} < 0$ then
                Compute $\phi^\mu$ for every $\mu \in \mathcal{V}(k) \setminus \{k\}$ and update the sequence $\sigma$ for the set $U$
                $S_{tree} \leftarrow S_{tree} \cup \left\{E_{p+I}^k > \alpha_p \sum_{v \in U} L^v + \alpha_p \sum_{\mu \in \mathcal{V}(k) \setminus \{k\}} \phi^\mu Y_p^\mu - \alpha_p \sum_{v \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^v\right\}$
            end
        end
    end
end
```

The intuition behind Algorithm 2 is provided as follows. Firstly, Algorithm 2 builds an initial set U by considering all nodes that might appear in the right-hand side of a path inequality for each leaf node λ of $\mathcal{L}(k)$ and computes the violation value (*viol*) corresponding to the tree inequality defined by the set U. Secondly, the algorithm explores the neighborhood of the current set U, where each neighbor set U' is obtained by removing one node from the set U. Note that computing the value of a tree inequality is not straightforward and is particularly time-consuming due to the fact that the coefficients ϕ and the sequence σ of the leaf nodes need to be updated for each neighbor set of U. A *first improvement* strategy is carried out by the algorithm, namely, each time a neighbor set U' with a lower violation value than the current violation value of set U is found, the algorithm updates the set U to the neighbor set U'. Finally, the algorithm stops when no neighbor set U' has a lower violation value than the violation value provided by the set U.

6. Computational results and discussion

We develop a branch-and-cut algorithm for solving the problem (14)-(28). This algorithm relies on the cutting-plane generation algorithms proposed in Section 5 to add valid inequalities into the Echelon Stock reformulation discussed in Section 3. We provide in this Section the results of some computational experiments carried out on randomly generated instances of the problem. The main objective of these experiments is to assess the effectiveness of the branch-and-cut algorithm by comparing it with the one of a stand-alone mathematical programming solver. .

In what follows, we introduce the setting used to randomly generate instances based on the data presented in [18] and [20] before discussing the detailed results of our computational experiments.

6.1. Instances generation

The following test data were randomly generated based on the instances generation scheme provided in [18].

- The demand for finished products d^n at each node n were generated from the discrete uniform distribution $DU(100, 1000)$.
- The bill of material was generated such that $\alpha_0 = \alpha_I + 1 = 1$, and for each $p = 1, ..., I, \alpha_p$ was generated from $DU(1, 6)$.
- The set-up costs for the disassembly process f_0^n were generated from $DU(50000, 70000)$, the set-up costs f_p^n for each refurbishing process $p = 1, ..., I$ from $DU(4000, 8000)$, and the set-up cost for the reassembly process f_{I+1}^n from $DU(50000, 70000)$.
- The unit inventory holding costs for used products h_0^n was fixed and set to 1. The unit inventory holding costs h_i^n for each recoverable parts $i \in \mathcal{I}_r$ were generated from $DU(2, 7)$. The unit inventory holding costs h_i^n for each serviceable part $i \in \mathcal{I}_s$ were generated from $DU(7, 12)$. To ensure non negative echelon costs, we generated the unit inventory holding costs for the remanufactured products, h_{2I+1}^n, from $\sum_{i=1}^{I} \alpha_i h_{I+i}^n + \varepsilon$ where we generated ϵ from $DU(80, 100)$.

For the proportion of recoverable parts π_i^n, $i \in \mathcal{I}_r$, obtained by disassembling one unit of used product at node $n \in \mathcal{V}$, we defined three intervals for the uniform probability distribution corresponding to three quality levels. These intervals are based on the values presented in the case study reported by Jayaraman [20].

- Low nominal quality level (Q1):, $\pi_i^n \sim U[0.08, 0.25]$
- Medium nominal quality level (Q2):, $\pi_i^n \sim U[0.11, 0.58]$
- High nominal quality level (Q3):, $\pi_i^n \sim U[0.21, 0.79]$

Similarly, we defined three intervals for the discrete uniform distribution corresponding to three levels of returned product volumes.

- Low nominal level of returns (R1): $r^n \sim DU(335, 2150)$.
- Medium nominal level of returns (R2): $r^n \sim DU(1738, 3454)$.
- High nominal level of returns (R3): $r^n \sim DU(704, 7942)$.

Finally, we define the following probability distribution for the input parameters specific to the problem studied in this paper.

- The lost-sales unit penalty costs l^n were set to 10000, at each node $n \in \mathcal{V}$.
- The cost for discarding one unit of recoverable part, $i \in \mathcal{I}_r \cup \{0\}$, is defined at each node $n \in \mathcal{V}$ as follows, $q_i^n = h_i^n * \frac{T}{\beta}$, where $\beta \sim U[2, T]$.

- The cost for discarding the unrecoverable parts generated during the disassembly process is computed as $g^n = \sum_{i=1}^{I} q_i^n(1 - \pi_i^n)\alpha_i$.

We set the number of parts in a product to $I = 5$, the length of a decision stage to $b \in \{1, 2, 3\}$ periods and the number of stages to $s \in \{4, 5, 6, 7, 8, 9\}$. The number of immediate successors c of each last-period-of-stage node is defined between 2 and 6. This leads to 16 different structures of the scenario trees. For each combination of scenario tree structure, used product quality level and used product quantity level, we randomly generated 10 instances, resulting in a total of 1440 instances.

6.2. Results

Each instance was solved using the echelon stock formulation (14)-(28) discussed in Section 3 by two alternative branch-and-cut methods:

1. The standard branch-and-cut algorithm embedded in the mathematical programming solver CPLEX 12.6.1 with the solver default settings,
2. The customized branch-and-cut algorithm proposed in this paper. Algorithm 1 is used to add path inequalities at the root node of the branch-and-bound search tree. Callbacks based on Algorithm 2 were used to add tree valid inequalities to the formulation during the course of the branch-and-bound search tree. The related linear programs and mixed-integer linear program were solved by CPLEX 12.6.1 with the solver default settings.

The algorithms were implemented in C++ using the Concert Technology environment. All tests were run on the supercomputing infrastructure of the National Laboratory for High Performance Computing (NLHPC), which consists in servers HP SL230, with Intel Ivy Bridge E5-2660V2 cores. We set the cluster to use two 2.20GHz cores and 16GB RAM to solve each instance. We imposed a time limit of 900 seconds. The corresponding results are displayed in Tables 1 and 2.

For each set of instances, we report six performance measures:

1. "Gap_{LP}" is the average percentage integrality gap. It is computed as the relative difference between the lower bound provided by the linear relaxation of the formulation and the value of the optimal integer solution. In case the instance could not be solved to optimality, the value of the best integer feasible solution found is used.
2. "Gap_{MIP}" is the average percentage residual gap reported by CPLEX. It is computed as the relative difference between the best lower bound and the best integer feasible solution found by the solver within the time limit.
3. "Time" is the average CPU time (in seconds) needed to find a guaranteed optimal integer solution (we used the value of 900s in case a guaranteed optimal integer solution could not be found within the computation time limit).
4. "Optimal" is the number of instances solved to optimality within the time limit.
5. "Solved" is the number of instances for which the algorithm could run without reaching the memory limit.
6. "Cuts" reports the average number of cuts add into the formulation.

The corresponding results are provided in Tables 1 and 2. Table 1 displays the results according to the scenario tree structure and size. Instances are grouped into two categories in order to be analyzed: small (between 126 and 381 nodes) and medium (between 468 and 1022 nodes).

We first note that, for small-size instances, the proposed branch-and-cut algorithm performs much better than CPLEX solver. This can be seen by the fact that the number of instances solved to optimality by the proposed algorithm (377) is more than twice the one solved by CPLEX (164). Moreover, the average residual gap is decreased from 0.11% with CPLEX to 0.05% with the proposed algorithm and the average computation time is decreased from 733s to 506s. This is mainly explained by the effectiveness of the cutting-plane generation at tightening the LP relaxation gap at the root node is shown on columns Gap_{LP}. Namely, a significant tightening of the LP relaxation is achieved via our approach since the average integrality gap is reduced from over 12.0% to less than 2.1%. This translates into an important reduction of the number of nodes explored by the branch-and-bound algorithm (from over 281,000 with CPLEX solver to less than 116,000 with our branch-and-cut algorithm). For medium-size, we note that none of the two algorithms could solve the problem to optimality within 900 seconds. However, we observe that the average residual gap is smaller with the proposed branch-and-cut algorithm than with CPLEX solver. Namely, it is reduced from 0.34% to 0.23%.

Table 1: Comparison between default ILOG-CPLEX configuration branch-and-cut algorithm. The instances are grouped by the number of node in the scenario tree.

Instances	CPLEX default					Branch-and-cut					
Nodes	Gap_{LP}	Gap_{MIP}	Time	Optimal	Solved	Gap_{LP}	Gap_{MIP}	Time	Optimal	Solved	Cuts
126	11.76	0.02	238.44	73	90	2.07	0.01	150.24	79	90	1449
189	13.40	0.05	660.96	30	90	1.71	0.02	275.16	72	90	2449
242	11.00	0.09	754.10	23	90	2.41	0.04	438.34	54	90	2411
254	12.33	0.11	784.48	15	90	2.03	0.03	511.11	47	87	2984
255	10.16	0.07	814.25	12	90	2.47	0.03	379.09	66	90	2766
255	11.50	0.13	818.19	11	90	2.13	0.05	649.18	33	90	2991
363	12.35	0.22	900.16	0	90	1.92	0.10	833.53	13	90	4162
381	13.81	0.21	900.30	0	90	1.66	0.09	815.81	13	89	5000
Average	12.04	0.11	733.86	164	720	2.05	0.05	506.11	377	716	2787
468	10.71	0.24	900.17	0	90	2.02	0.14	888.43	4	90	4865
510	12.74	0.29	900.27	0	90	1.95	0.16	900.96	1	89	5943
511	9.54	0.22	888.42	2	90	2.15	0.14	878.58	5	90	5619
682	10.21	0.32	900.20	0	86	2.31	0.23	901.23	0	90	6102
728	10.46	0.37	900.16	0	87	2.08	0.28	901.19	0	90	6822
765	13.63	0.39	900.12	0	84	1.59	0.23	901.15	0	90	9637
777	10.87	0.41	900.15	0	88	2.12	0.31	901.40	0	90	7258
1022	12.13	0.50	900.22	0	81	1.89	0.34	901.80	0	90	11236
Average	11.27	0.34	898.66	2	696	2.01	0.23	896.84	10	719	6540

It is worth mentioning that the scenario structure seems to have an impact on the performance of the proposed branch-and-cut algorithm. For instance, for the same scenario tree size of 255 nodes, the proposed branch-and-cut algorithm was more efficient at finding optimal solutions for the tree structure $(b, s, c) = (1, 8, 2)$ than for the scenario tree structure $(b, s, c) = (3, 4, 4)$. A reasonable explanation for this is that one structure propitiates the generation of tree inequalities more than the other. For instance, the structure $(3, 4, 4)$ involves 64 individual scenarios whereas the structure $(1, 8, 2)$ involves 256 individual scenarios that can be combined to find tree inequalities.

Table 2: Comparison between default ILOG-CPLEX configuration branch-and-cut algorithm. The instances are grouped by the nominal returns and quality levels.

Instances		CPLEX default					Branch-and-cut					
R	Q	Gap_{LP}	Gap_{MIP}	Time	Optimal	Solved	Gap_{LP}	Gap_{MIP}	Time	Optimal	Solved	Cuts
1	1	1.33	0.17	900.09	0	159	0.62	0.17	899.64	1	157	5132
	2	4.08	0.18	868.33	9	157	1.17	0.16	818.31	22	160	4768
	3	10.38	0.20	826.51	17	160	2.12	0.11	686.57	47	160	4677
Average		5.28	0.18	864.88	26	476	1.31	0.15	800.89	70	477	4857
2	1	3.76	0.15	852.85	10	159	1.02	0.13	807.19	23	159	4705
	2	13.48	0.19	765.83	30	157	2.27	0.09	596.36	60	160	4519
	3	24.54	0.41	828.31	17	156	3.51	0.19	677.57	50	160	4638
Average		13.86	0.25	815.79	57	472	2.27	0.14	693.47	133	479	4621
3	1	6.72	0.07	693.01	42	159	1.28	0.04	545.95	77	160	4438
	2	16.37	0.28	797.71	19	154	2.58	0.14	630.09	58	160	4531
	3	24.94	0.38	800.47	22	155	3.68	0.19	659.35	49	159	4605
Average		15.93	0.24	763.05	83	468	2.51	0.12	611.70	184	479	4525

In Table 2, the instances are grouped according to the returns volume and quality levels. These results show that these instance features have an impact on the problem resolution. Namely, formulation (14)-(28) displays a significant dispersion of the integrality gap: from 1.33% for the low volume and low quality case to 24.94% to the high volume high quality case. This might be explained by the relative weight of the lost sales penalty costs and fixed setup costs in the objective function. Namely, we note that the integrality gap mainly comes from the fact that the binary constraints on the setup variables Y_p^n are relaxed so that the value of the fixed setup costs is underestimated in the linear relaxation.

The larger the relative weight of the setup costs in the objective function, the larger the integrality gap. Now, in case the volume of returned products is high and their quality is good, a large part of the demand for remanufactured products will be satisfied so that the lost sales quantity will be close to 0. This implies that the setup costs will make up a large portion of the overall production costs, leading to a large integrality gap. On the contrary, in case the volume of returned products is low and their quality is bad, a large portion of the demand will not be satisfied, lost sales penalties will be high as compared so setup costs, leading to small integrality gaps. Note that one advantage of our cutting-plane generation algorithm is that it is capable of reducing the integrality gap in the same way for any product volume and quality level.

We note however that the instances corresponding to small lost sales quantity and displaying large LP gaps seem to be easier to solve than the instances corresponding to large lost sales quantity and displaying small LP gaps. This might be due to the fact that, over the course of the branch-and-bound search tree, the lower bound increase is slower when the weight of lost sales in the objective function is significant. Namely, in this case, making a branching decision on a binary setup variable in terms of lower bound improvement has a smaller impact on the objective function value.

6.3. Rolling horizon simulation

The aim of these experiments is to assess the practical performance of the stochastic planning model with respect to the simpler deterministic model ignoring uncertainty. This is achieved by carrying out a rolling horizon simulation similar to the one used by Brandimarte [21].

Each experiment consists in simulating the application of the first-stage planning decisions over 12 time periods and in comparing the total cost incurred when applying the planning decisions established by the deterministic or the stochastic model. Note that the cost considered in the rolling horizon simulation is not the objective function of the optimization model, but the sum over time of the true cost incurred by the application of the first-stage decision planning over the true scenario. The relative difference is computed as $100(C_D - C_S)/C_S$, where C_D and C_S are the total costs accumulated over each simulation run for the deterministic and stochastic model, respectively.

Table 3: Values of the stochastic solution for 2,3 and 4 children in the scenario tree. The instances are grouped by returns and quality levels and the values was calculated as follows: $100(C_D - C_S)/C_S$.

Instances		c=2		c=3		c=4	
R	*Q*	Ave.	MAD	Ave.	MAD	Ave.	MAD
1	1	19.9	7.7	25.4	9.6	24.6	8.9
	2	36.1	18.5	33.2	16.9	36.2	17.6
	3	43.4	41.1	47.8	42.6	65.2	54.6
2	1	26.7	14.5	31.0	15.6	34.3	18.6
	2	74.3	61.9	66.5	54.7	74.1	56.6
	3	52.0	70.5	69.2	89.7	39.8	48.5
3	1	37.1	31.8	34.0	25.8	45.5	39.8
	2	60.2	62.1	46.0	44.5	65.4	69.5
	3	33.2	55.9	46.1	68.6	50.0	70.7
Average		42.5	42.5	44.3	42.2	48.2	44.8

The instances are generated in the same way as in the previous subsection. Since running the simulation is quite costly, we considered small scenario trees with at most 255 nodes in order to gain some basic insights. More specifically, at each iteration of the rolling horizon simulation, a scenario tree with $s = 4$ stages and $b = 3$ periods per stage is generated for the stochastic model. The number of branches c at each stage is set between 2 and 4. The stochastic model is solved by our branch-and-cut algorithm and, to speed up the simulation, a time limit of 900 seconds is imposed. Since the deterministic model is easy to solve for instances with 12 time periods, we use the branch-and-bound algorithm embedded in ILOG-CPLEX to solve it with no suboptimality tolerance.

We randomly generate 100 true independent scenarios for each nominal returns level, each quality level and each scenario tree structure, resulting in a total of 2700 true scenarios. Table 3 reports the average relative increase in cost, as defined before, and its Mean Absolute Deviation (MAD) for each nominal returns level, nominal quality level and number of branches of the scenario tree of the stochastic model.

The results suggested the optimistic practical performance of the stochastic planning model as compared to the deterministic model in all cases. Namely, on average, the increase in the actual production cost observed when using deterministic model rather than the stochastic model is 45%. Moreover, for each nominal returns and quality level, the stochastic model outperforms the deterministic model, even if simple scenario trees with two branches at each stage are used. This is mainly explained by the fact that the amount of lost sales penalty costs is significantly decreased when using the stochastic model.

Moreover, we observe a clear, but not large, average improvement when increasing the number of branches. However, this improvement has to be counter-balanced by the increased CPU effort required. Of course, these results should be taken carefully, given the large mean absolute deviation. This large variability is partly due to the instance generation framework, which involves some uniform distributions with large amplitude of their intervals. It can be easily observed by comparing the mean absolute deviation of the instances with low nominal returns and quality levels, which have a relative smaller amplitude of its uniform distribution, to the high nominal and quality levels.

7. Conclusions

We considered an uncapacitated multi-item multi-echelon lot-sizing problem within a remanufacturing system involving three production echelons: disassembly, refurbishing and reassembly. We considered a stochastic environment, in which the input data of the optimization problem are subject to uncertainty and proposed a multi-stage stochastic integer programming approach relying on scenario trees to represent the uncertain information structure. This resulted in the formulation of a large-size mixed-integer linear program involving a series of big-M type constraints. We developed a branch-and-cut algorithm in order to solve the obtained MILP to optimality. This algorithm relied on a new set of tree inequalities obtained by combining valid inequalities previously known for each individual scenario of the scenario tree. The tree inequalities are used within a cutting-plane generation procedure based on a heuristic resolution of the corresponding separation problem. Computational experiments carried out on randomly generated instances show that the proposed Branch-and-Cut algorithm performs well as compared to the use of a stand-alone mathematical solver, especially for small size instances.

However, neither CPLEX solver nor the proposed branch-and-cut algorithm could solve large-size instances to optimality in a reasonable computation time. Hence, an interesting direction for further research could be to study other heuristic solution approaches in order to reduce the total computation time. Moreover, we assumed in our problem modeling uncapacitated production processes. Extending the present work in order to account for production resources with limited capacity could also be worth investigating.

8. Acknowledgements

This research was partially supported by the supercomputing infrastructure of the NLHPC (ECM-02) in which computational tests were conducted. Authors are grateful for partial support from Gaspard Monge Program for Optimization, operational research and their interactions with data science (PGMO) and the Fondation mathmatique Jacques Hadamard (FMJH).

References

[1] R. T. Lund, B. Mundial, Remanufacturing: the experience of the United States and implications for developing countries, Vol. 31, World Bank, 1984.

[2] V. D. R. Guide, V. Jayaraman, R. Srivastava, Production planning and control for remanufacturing: a state-of-the-art survey, Robotics and Computer-Integrated Manufacturing 15 (3) (1999) 221–230.

[3] V. D. R. Guide, Production planning and control for remanufacturing: industry practice and research needs, Journal of operations Management 18 (4) (2000) 467–483.

[4] M. L. Lage Junior, M. G. Filho, Production planning and control for remanufacturing: literature review and analysis, Production Planning & Control 23 (6) (2012) 419–435.

[5] M. A. Ilgin, S. M. Gupta, Environmentally conscious manufacturing and product recovery (ecmpro): a review of the state of the art, Journal of environmental management 91 (3) (2010) 563–591.

[6] O. A. Kilic, A mip-based heuristic for the stochastic economic lot sizing problem with remanufacturing, IFAC Proceedings Volumes 46 (9) (2013) 742–747.

[7] O. A. Kilic, H. Tunc, S. A. Tarim, Heuristic policies for the stochastic economic lot sizing problem with remanufacturing under service level constraints, European Journal of Operational Research.

[8] M. A. Naeem, D. J. Dias, R. Tibrewal, P.-C. Chang, M. K. Tiwari, Production planning optimization for manufacturing and remanufacturing system in stochastic environment, Journal of Intelligent Manufacturing (2013) 1–12.

[9] P. B. Macedo, D. Alem, M. Santos, M. L. Junior, A. Moreno, Hybrid manufacturing and remanufacturing lot-sizing problem with stochastic demand, return, and setup costs, The International Journal of Advanced Manufacturing Technology 82 (5-8) (2016) 1241–1257.

[10] T. Hilger, F. Sahling, H. Tempelmeier, Capacitated dynamic production and remanufacturing planning under demand and return uncertainty, OR spectrum 38 (4) (2016) 849–876.

[11] H.-F. Wang, Y.-S. Huang, A two-stage robust programming approach to demand-driven disassembly planning for a closed-loop supply chain system, International Journal of Production Research 51 (8) (2013) 2414–2432.

[12] C. Fang, X. Liu, P. M. Pardalos, J. Long, J. Pei, C. Zuo, A stochastic production planning problem in hybrid manufacturing and remanufacturing systems with resource capacity planning, Journal of Global Optimization 68 (4) (2017) 851–878.

[13] Y. Guan, S. Ahmed, G. L. Nemhauser, A. J. Miller, A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem, Mathematical Programming 105 (1) (2006) 55–84.

[14] Y. Guan, S. Ahmed, G. L. Nemhauser, Cutting planes for multistage stochastic integer programs, Operations research 57 (2) (2009) 287–298.

[15] M. Di Summa, L. A. Wolsey, Lot-sizing on a tree, Operations Research Letters 36 (1) (2008) 7–13.

[16] M. Zhang, S. Küçükyavuz, S. Goel, A branch-and-cut method for dynamic decision making under joint chance constraints, Management Science 60 (5) (2014) 1317–1333.

[17] M. Loparic, Y. Pochet, L. A. Wolsey, The uncapacitated lot-sizing problem with sales and safety stocks, Mathematical Programming 89 (3) (2001) 487–504.

[18] H.-D. Ahn, D.-H. Lee, H.-J. Kim, Solution algorithms for dynamic lot-sizing in remanufacturing systems, International Journal of Production Research 49 (22) (2011) 6729–6748.

[19] Y. Pochet, L. A. Wolsey, Production planning by mixed integer programming, Springer Science & Business Media, 2006.

[20] V. Jayaraman, Production planning for closed-loop supply chains with product recovery and reuse: an analytical approach, International Journal of Production Research 44 (5) (2006) 981–998.

[21] P. Brandimarte, Multi-item capacitated lot-sizing with demand uncertainty, International Journal of Production Research 44 (15) (2006) 2997–3022.