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

Franco Quezada, Céline Gicquel, Safia Kedad-Sidhoum, Dong Quan Vu

# 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, CNRS, 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 multi-stage stochastic planning model with respect to a deterministic model and a two-stage stochastic planning 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 or 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 customers' 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, e.g. 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. They 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 and seek to develop an approach where the multi-stage aspect of the decision making process is explicitly considered.

Note that such multi-stage decision making processes have already been considered for stochastic production planning problems displaying features similar to our problem. Thus, Denizel et al. [6] studied a production planning problem for a company remanufacturing large-scale mailing equipment. They considered uncertainty in the quality of the returns and developed a multi-stage stochastic programming approach. Kazemi et al. [7] investigated a sawmill production planning problem with uncertainty on both the demand and the raw materials quality and also proposed a multi-stage stochastic programming approach. Production planning for remanufacturing under stochastic demand and returns was studied by Li et al. [8]. They proposed a stochastic dynamic programming based model for this problem. However, these three works all assume linear production costs, which enable them to formulate the optimization problem using only continuous decision variables. In contrast, we consider fixed production setup costs in our problem modeling, leading to the formulation of a lot-sizing problem involving a set of binary decision variables. In what follows, we thus focus on reviewing previously published works on stochastic lot-sizing for remanufacturing systems.

In recent years, stochastic lot-sizing problems for remanufacturing or 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 [9], [10] and [11], taking under consideration both stochastic demand and returns quantity. Kilic [9] and Kilic et al. [10] included customer service level constraints and developed a heuristic approach based on a static-dynamic uncertainty strategy. Naeem et al. [11] 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. [12] and Hilger et al. [13] studied a multi-item variant of the problem taking into account both stochastic demand and returns quantity. Macedo et al. [12] extended the previous works ([9], [10] and [11]) 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. [13] 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 [14] and Fang et al. [15] 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 [14] developed a two-stage stochastic programming model aiming at finding a compromise between the expected cost and the solution robustness. Fang et al. [15] 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. [15]. 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 [16], [17], [18] and [19]. Guan et al. [16] investigated an uncapacitated lot-sizing problem and extended the $(\ell, S)$ 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 Wolsey [18] studied a capacitated lot sizing problem and extended the work of Guan et al. [16] by proving that the $(Q, S_Q)$ valid inequality is dominated by a mixing inequality. Additionally, Guan et al. [17] 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. [19] 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 [20]. The number of these valid inequalities increases exponentially fast with the size of 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

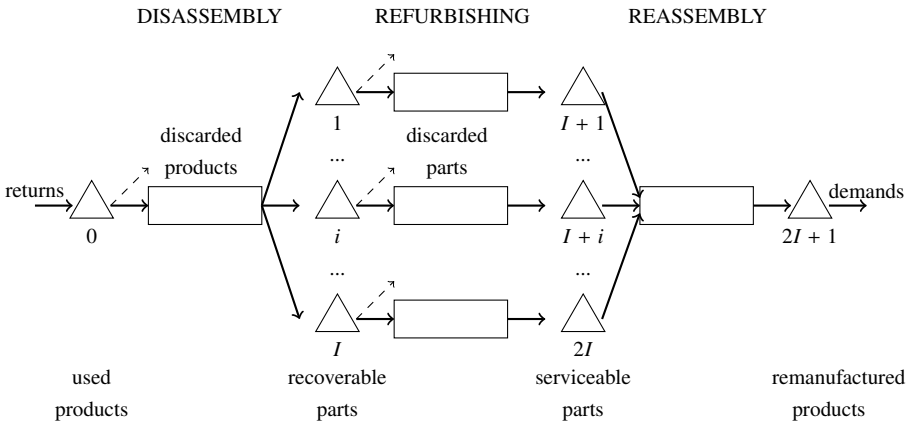


Figure 1: Illustration of studied remanufacturing system

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 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 the 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. [21] 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 $\rho^n$: note that the sum of $\rho^n$ over all nodes $n$ corresponding to a given time period $t$ is equal to 1. Each non-terminal node $n$ is the 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 $\mu$ 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 one used for 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:

- $\alpha_i$: 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 $n \in \mathcal{V}$,
- $\pi_i^n$: 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 $r^n$, $d^n$, $\pi_i^n$, $l^n$, $f_p^n$, $h_p^n$, $q^n$ 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

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

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 $r^\nu$ on the nodes $\nu$ 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 $\lambda$ in $\mathcal{L}(n)$, of the cumulated demand on the path from the node $n$ to the leaf node $\lambda$.

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

- $M_0^n = \min \left\{ \sum_{\nu \in \mathcal{P}(0,n)} r^\nu, \max_{\lambda \in \mathcal{L}(n)} \left\{ \frac{\sum_{\nu \in \mathcal{P}(n,\lambda)} d^\nu}{min_{i=1\ldots I} \pi_i^n} \right\} \right\}$

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

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. This first comes from the problem size which, for a given planning horizon length, is much larger in the stochastic case than in the deterministic case. Namely, in the stochastic case, the mixed-integer linear programming formulation involves $O(|\mathcal{V}||\mathcal{J}|)$ binary variables, $O(|\mathcal{V}||\mathcal{J}|)$ continuous variables and $O(|\mathcal{V}||\mathcal{J} + \mathcal{I}|)$ constraints. The size of the scenario tree $|\mathcal{V}|$ is in $O(c^{T+1})$ with $c$ is the number of children per node and $T$ the number of stages. The MILP formulation size thus grows exponentially fast with the number of decision stages $T$. 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 describe a reformulation of the problem that provides a way 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 [22] 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 demand $ed_i^n$ for an intermediate product can be understood as the translation of the external demand for the finished product into an independent demand for the intermediate product. For each product $i = 1...2I$, we straightforwardly define the echelon demand as $ed_i^n = \alpha_i d^n$. We note however that, in our case, it is not possible to properly define such an echelon demand for the used product $i = 0$. Namely, this demand could be defined as $ed_0^n = \frac{d^n}{min_{i \in \mathcal{I}_r} \pi_i^n}$ by considering that the amount of used product to disassemble to satisfy the external demand $d^n$ is determined by the disasssembly yield of the item $i \in \mathcal{I}_r$ which is the most difficult to recover at node $n$. However, as the disassembly yields are time-varying and stochastic, the actual amount of used product needed to satisfy the external demand $d^n$ depends on the period in which it is disassembled and might be larger or smaller than $\frac{d^n}{min_{i \in \mathcal{I}_r} \pi_i^n}$. Hence, using the echelon demand $ed_0^n$ might lead to inconsistent disassembly production decisions. We thus focus in what follows on defining echelon stock variables for products $i \in \{1, ..., 2I + 1\}$.

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 within its successors in the 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 $n \in \mathcal{V}$

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:

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

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 (27)-(28) define the domain of the decision variables.

### *3.2. Single echelon subproblems*

The introduction of echelon inventory variables leads to the elimination of the dependent demand term 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:

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

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. [20] 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. [17] 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 $c_k^\lambda$ 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 $c_k^\lambda$ to the leaf node $\lambda$, not necessarily consecutive.

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*

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

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

The proof is direct following the proof in [20].

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 $\lambda$. 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:

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

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 $E_i^k / \alpha_i$ 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*

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

*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 [16] and [17]. 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 $\sum_{v \in U_{k,\lambda}} d^v$: $\sum_{v \in U_{k,\sigma_1}} d^v \leq ... \leq \sum_{v \in U_{k,\sigma_l}} d^v \leq ... \leq \sum_{v \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^v$. We set $\sum_{v \in U_{k,\sigma_0}} d^v = 0$. The following inequality*

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

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

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

*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:

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

We rewrite the inequalities (44) in a form making it easier to apply Theorem 2 in [17].

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

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 [17] to combine these $|\mathcal{L}(k)|$ path inequalities and derive a new tree inequality as follows:

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

where, the coefficient $\phi^\mu$ 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 $\sum_{\nu\in U_{k,\lambda}} d^\nu$: $\sum_{\nu\in U_{k,\sigma_0}} d^\nu \leq \sum_{\nu\in U_{k,\sigma_1}} d^\nu \leq ... \leq \sum_{\nu\in U_{k,\sigma_l}} d^\nu \leq ... \leq \sum_{\nu\in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^\nu$. We set $\sum_{\nu\in U_{k,\sigma_0}} d^\nu = 0$. The following inequality*

$$\min_{i\in\mathcal{I}_r}\Big[\frac{E^k_i}{\alpha_i}\Big] + \sum_{\nu\in U} L^\nu + \sum_{\mu\in\mathcal{V}(k)\setminus\{k\}} \phi^\mu Y^\mu_0 \geq \sum_{\nu\in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^\nu \tag{45}$$

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

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

## 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 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 proving 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 $\mathcal{P}(c^\lambda_k, \lambda)$ 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 $\mathcal{P}(c^\lambda_k, \lambda)$, which satisfy the following inequality:

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

We underline that the above strategy can be implemented in polynomial time [20], namely, the running-time of the proposed separation algorithm is $O(P^2)$, where $P$ corresponds to the number of nodes in the set $\mathcal{P}(c^\lambda_k, \lambda)$.

*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 $\lambda_{min} = argmin_{\lambda \in \mathcal{L}(k)} \tilde{E}^k_{p+I} - \alpha_p \sum_{\nu \in U_{k,\lambda}} \left[ d^\nu \left(1 - \sum_{\mu \in \mathcal{P}(c_k^\lambda, \nu)} \tilde{Y}_p^\mu \right) - \tilde{L}^\nu \right]$.

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 $\nu_{min}$, 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 $\lambda_{min}$ 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. Thus, if a cut involving leaf node $\lambda_{min}$ is generated at node $k$ at a given iteration, for all $n \in \mathcal{P}(k, \nu_{min})$, $\lambda_{min}$ is removed temporarily, i.e. for the course of the current iteration, from the leaf node set $\mathcal{L}'(n)$ considered for the search of violated valid inequalities at node $n$. It is then reintegrated into all leaf node sets at the beginning of the next iteration.

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 $S_{path}$

Initialize $S_{path} \leftarrow \emptyset$
**foreach** $p \in \mathcal{J}$ **do**
  $\mathcal{L}'(\cdot) \leftarrow \mathcal{L}(\cdot)$
  **foreach** $k \in \mathcal{V}$ **do**
    $viol_{min} \leftarrow 0$
    **foreach** $\lambda \in \mathcal{L}'(k)$ **do**
      $U_{k,\lambda} \leftarrow \emptyset$, $\nu_{last} \leftarrow k$
      **foreach** $\nu \in \mathcal{P}(c_k^{\lambda}, \lambda)$ **do**
        **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**
          $U_{k,\lambda} \leftarrow U_{k,\lambda} \cup \{\nu\}$ , $\nu_{last} \leftarrow \nu$
        **end**
      **end**
      $viol \leftarrow \tilde{E}_{p+I}^{k} - \alpha_p \sum_{\nu \in U_{k,\lambda}} \left[d^{\nu}\left(1 - \sum_{\mu \in \mathcal{P}(c_k^{\lambda}, \nu)} \tilde{Y}_p^{\mu}\right) - \tilde{L}^{\nu}\right]$
      **if** $viol < viol_{min}$ **then**
        $viol_{min} \leftarrow viol$, $\lambda_{min} \leftarrow \lambda$, $\nu_{min} \leftarrow \nu_{last}$
      **end**
    **end**
    **if** $viol_{min} < 0$ **then**
      $S_{path} \leftarrow S_{path} \cup \left\{E_{p+I}^{k} > \alpha_p \sum_{\nu \in U_{k,\lambda_{min}}} \left[d^{\nu}(1 - \sum_{\mu \in \mathcal{P}(c_k^{\lambda}, \nu)} Y_p^{\mu}) - L^{\nu}\right]\right\}$
      **foreach** $\nu \in \mathcal{P}(c_k^{\lambda_{min}}, \nu_{min})$ **do**
        $\mathcal{L}'(\nu) \leftarrow \mathcal{L}'(\nu) \setminus \{\lambda_{min}\}$
      **end**
    **end**
  **end**
**end**

### *5.3. Separation algorithm for tree inequalities*

Given a non-leaf node $k$, solving the separation problem for inequalities (43) (resp. (45)) requires to identify a subset of nodes $U \subseteq \mathcal{V}(k)$ minimizing the difference between the left-hand side and the right-hand side of inequalitie (43) (resp. (45)). 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 $v$ of $\mathcal{V}(k)$ in the set $U$ not only changes the left-hand side of the inequality by a quantity $L^v + \phi^v Y^v$, but also potentially impacts the value of the coefficient $\phi^{\mu}$ for all other nodes in $\mathcal{V}(k) \setminus \{k\}$. In addition, selecting a node $v$ of $\mathcal{V}(k)$ potentially changes the order of the sequence $\sigma$ 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 $S_{tree}$

1 Initialize $S_{tree} \leftarrow \emptyset$, $\mathcal{V}' := \{v \in \mathcal{V} : |\mathcal{E}(v)| > 1\}$
2 **foreach** $p \in \mathcal{J}$ **do**
3  **foreach** $k \in \mathcal{V}'$ **do**
4  $U \leftarrow \emptyset$, $viol_{curr} \leftarrow \infty$
5  **foreach** $v \in \mathcal{V}(k)$ **do**
6  **if** $\left[d^v\left(1 - \sum_{\mu \in \mathcal{P}(c^\lambda_k, v)} \tilde{Y}^\mu_p\right) - \tilde{L}^v\right] > 0$ **then**
7  $U \leftarrow U \cup \{v\}$
8  **end**
9  **end**
10  **if** $U = \emptyset$ **then**
11  break;
12  **else**
13  Determine the ordering $\sigma$ of the leaf nodes for set $U$ and compute $\phi^\mu$ for every $\mu \in \mathcal{V}(k) \setminus \{k\}$
14  $viol_{min} \leftarrow \tilde{E}^k_{p+I} + \alpha_p \sum_{v \in U} \tilde{L}^v + \alpha_p \sum_{\mu \in \mathcal{V}(k)\setminus\{k\}} \phi^\mu \tilde{Y}^\mu_p - \alpha_p \sum_{v \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^v$
15  **while** $viol_{min} < viol_{curr}$ **do**
16  $viol_{curr} \leftarrow viol_{min}$
17  **foreach** $v \in U$ **do**
18  $U' \leftarrow U \setminus \{v\}$
19  Update the ordering $\sigma$ for set $U'$ and compute $\phi^\mu$ for every $\mu \in \mathcal{V}(k) \setminus \{k\}$ and
20  $viol \leftarrow \tilde{E}^k_{p+I} + \alpha_p \sum_{v \in U'} \tilde{L}^v + \alpha_p \sum_{\mu \in \mathcal{V}(k)\setminus\{k\}} \phi^\mu \tilde{Y}^\mu_p - \alpha_p \sum_{v \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^v$
21  **if** $viol < viol_{min}$ **then**
22  $viol_{min} \leftarrow viol$, $U \leftarrow U'$
23  **end**
24  **end**
25  **end**
26  **if** $viol_{min} < 0$ **then**
27  Update the ordering $\sigma$ for the set $U$ and compute $\phi^\mu$ for every $\mu \in \mathcal{V}(k) \setminus \{k\}$
28  $S_{tree} \leftarrow S_{tree} \cup \left\{E^k_{p+I} > \alpha_p \sum_{v \in U} L^v + \alpha_p \sum_{\mu \in \mathcal{V}(k)\setminus\{k\}} \phi^\mu Y^\mu_p - \alpha_p \sum_{v \in U_{k,\sigma_{|\mathcal{L}(k)|}}} d^v\right\}$
29  **end**
30  **end**
31  **end**
32 **end**

The intuition behind Algorithm 2 is the following. It first builds an initial set $U$ containing all nodes in $\mathcal{V}(k)$ that would be selected in the set $U_{k,\lambda}$ when looking for a violated path inequality: see lines 4-9 of Algorithm 2. If this initial set is empty, we stop: see lines 10-11. Otherwise, we try to find a tree inequality as violated as possible by removing, one by one, some nodes from set $U$: see lines 12-30. More precisely, we start by computing the amount of violation ($viol_{min}$) obtained with the initial set $U$: this requires to determine the ordering $\sigma$ of the leaf nodes in $\mathcal{L}_k$ corresponding to set $U$ and to compute coefficients $\phi^\mu$ for every $\mu \in \mathcal{V}(k)$. We then explore the neighborhood of set $U$ which consists of all subsets of $U$ obtained by removing a single node. For each considered neighbor, we compute the amount of violation of the corresponding tree inequality: this operation is particularly time-consuming due to the fact that the ordering $\sigma$ of the leaf nodes and the coefficients $\phi$ need to be recomputed for each neighbor. Note that a first improvement strategy is used to explore the neighborhood of the current set, i.e. we update the current set $U$ as soon as a better neighbor set $U'$ is found. Finally, the algorithm stops when no neighbor set $U'$ has a violation value lower than the one of the current set $U$.

## 6. Computational results and discussion

We develop two branch-and-cut algorithms for solving problem (1)-(13). These algorithms rely on the cutting-plane generation algorithms proposed in Section 5 to add valid inequalities to the Echelon Stock reformulation discussed in Section 3. We provide in this Section the results of 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 algorithms by comparing them 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 [21] and [23] 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 [21].

- The demands 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 $h_0^n$ were 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 $\epsilon$ from $DU(80, 100)$.

For the proportion of recoverable parts $\pi_i^n$, $i \in \mathcal{I}_r$, 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 [23].

- 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 $I \in \{5, 10\}$, 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. Figure 2 displays a scenario tree with $(b, s, c) = (3, 3, 2)$. 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 2880 instances.

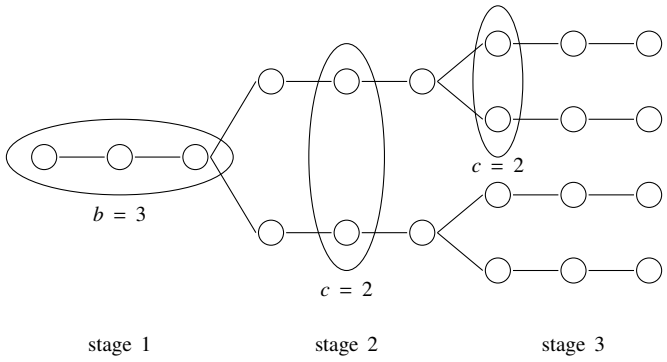


Figure 2: Scenario tree structure

### *6.2. Results*

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

1. The standard branch-and-cut algorithm embedded in the mathematical programming solver CPLEX with the solver default settings.
2. BC1: a customized branch-and-cut algorithm using only Algorithm 1 to add path inequalities at the root node of the branch-and-bound search tree.
3. BC2: a customized branch-and-cut algorithm in which Algorithms 1 and 2 are used to add path and tree inequalities at the root node of the branch-and-bound search tree and UserConstraints callbacks based on Algorithm 2 are used to add tree valid inequalities to the formulation during the course of the branch-and-bound search tree. Note that Algorithm 2 is not only capable to find valid tree inequalities, but also valid path inequalities.

We note that even though the natural formulation (1)-(13) and the echelon stock formulation (14)-(28) have the same linear relaxation, preliminary experiments show that CPLEX performs slightly better using the echelon stock formulation than the natural formulation on the considered set of instances. We thus only report the results obtained using this latter formulation.

All related linear programs and mixed-integer linear program were solved by CPLEX 12.8 with the solver default settings. The algorithms were implemented in C++ using the Concert Technology environment. All tests were run on the computing infrastructure of the Laboratoire d'Informatique de Paris VI (LIP6), which consists in a cluster of Intel Xeon Processors X5690. We set the cluster to use two 3.46GHz cores and 12GB RAM to solve each instance. We imposed a time limit of 900 seconds. The corresponding results are displayed in Tables 1 and 3 for instances with $I = 5$ and Tables 2 and 4 for instances with $I = 10$.

For each set of instances, we report five 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.

Table 1: Comparison between default CPLEX configuration and the customized branch-and-cut algorithms for instances with $I = 5$. Instances are grouped according to the number of nodes in the scenario tree.

| Instances | CPLEX default | | | | BC1 (Path) | | | | | BC2 (Path and Tree) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nodes | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts |
| 126 | 10.25 | 0.02 | 338.04 | 68 | 1.83 | 0.01 | 200.50 | 79 | 1190 | 1.58 | 0.01 | 160.52 | 80 | 1350 |
| 189 | 13.35 | 0.07 | 716.55 | 26 | 1.77 | 0.03 | 429.89 | 58 | 2146 | 1.54 | 0.02 | 339.91 | 66 | 2315 |
| 242 | 10.68 | 0.13 | 856.61 | 7 | 2.34 | 0.06 | 640.80 | 37 | 1790 | 2.00 | 0.05 | 550.86 | 48 | 2109 |
| 254 | 12.25 | 0.16 | 877.31 | 5 | 2.05 | 0.07 | 693.37 | 33 | 2439 | 1.74 | 0.05 | 589.49 | 46 | 2778 |
| 255 | 8.72 | 0.12 | 810.42 | 12 | 2.30 | 0.06 | 629.43 | 36 | 1631 | 1.84 | 0.04 | 488.53 | 51 | 2385 |
| 255 | 10.68 | 0.15 | 891.51 | 1 | 1.89 | 0.07 | 801.11 | 16 | 2209 | 1.66 | 0.05 | 660.45 | 37 | 2427 |
| 363 | 11.78 | 0.23 | 900.07 | 0 | 1.90 | 0.12 | 863.67 | 5 | 3476 | 1.63 | 0.11 | 855.92 | 8 | 3820 |
| 381 | 12.91 | 0.28 | 900.24 | 0 | 1.59 | 0.14 | 888.23 | 3 | 4470 | 1.36 | 0.11 | 874.01 | 4 | 4837 |
| Average | 11.33 | 0.15 | 786.34 | 119 | 1.96 | 0.07 | 643.37 | 267 | 2418 | 1.67 | 0.05 | 564.96 | 340 | 2752 |
| 468 | 10.34 | 0.34 | 900.17 | 0 | 1.96 | 0.19 | 898.79 | 1 | 3778 | 1.70 | 0.18 | 900.69 | 0 | 4159 |
| 510 | 12.02 | 0.38 | 900.27 | 0 | 1.89 | 0.20 | 900.74 | 0 | 4963 | 1.58 | 0.21 | 899.74 | 1 | 5693 |
| 511 | 9.61 | 0.29 | 900.18 | 0 | 2.50 | 0.21 | 890.58 | 3 | 3332 | 2.02 | 0.21 | 897.06 | 2 | 4858 |
| 682 | 9.47 | 0.40 | 900.20 | 0 | 2.18 | 0.33 | 898.19 | 1 | 4430 | 1.87 | 0.31 | 900.73 | 0 | 5220 |
| 728 | 9.88 | 0.45 | 900.32 | 0 | 2.02 | 0.37 | 900.88 | 0 | 5421 | 1.73 | 0.37 | 900.74 | 0 | 6375 |
| 765 | 13.37 | 0.47 | 900.43 | 0 | 1.54 | 0.34 | 900.97 | 0 | 9043 | 1.30 | 0.32 | 900.90 | 0 | 9773 |
| 777 | 10.90 | 0.58 | 900.25 | 0 | 2.13 | 0.40 | 900.69 | 0 | 6019 | 1.88 | 0.42 | 901.07 | 0 | 6581 |
| 1022 | 12.75 | 0.68 | 900.48 | 0 | 1.87 | 0.42 | 901.01 | 0 | 10058 | 1.61 | 0.47 | 901.05 | 0 | 11537 |
| Average | 11.04 | 0.45 | 900.29 | 0 | 2.01 | 0.31 | 898.98 | 5 | 5880 | 1.71 | 0.31 | 900.24 | 3 | 6774 |

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. "$\#Opt$" is the number of instances solved to optimality within the time limit.
5. "Cuts" reports the average number of cuts added to the formulation.

Tables 1 and 2 display 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).

First, results from Table 1 indicate that, for small-size instances with $I = 5$, the two proposed branch-and-cut algorithms perform much better than CPLEX solver. This can be seen by the fact that the number of instances solved to optimality is more than twice the number solved by CPLEX when using Algorithm BC1 and almost three times the number solved by CPLEX when using Algorithm BC2. Moreover, the average residual gap is decreased from 0.15% with CPLEX to 0.07% with BC1 and 0.05% with BC2 and the average computation time is decreased from 786s with CPLEX to 643s with BC1 and 565s with BC2. This is mainly explained by the effectiveness of the cutting-plane generation at tightening the LP relaxation gap at the root node, as shown by the results provided in columns $Gap_{LP}$. Namely, a significant tightening of the LP relaxation is achieved via our approach since the average LP gap is reduced from over 11% to less than 2%. Moreover, we note that on these instances, the addition of tree inequalities using Algorithm 2 during the course of the branch-and-cut algorithm proves useful to improve its performance both in terms of solution quality and computation time. As for the medium-size instances, we note that none of the three algorithms could solve them to optimality within 900 seconds. However, we observe that the average residual gap is smaller with the proposed branch-and-cut algorithms than with CPLEX solver. Namely, it is reduced from 0.45% to 0.31%.

Second, by looking at Table 2, we observe that the performance of the three solution approaches decreases when the number of items increases. This is mainly explained by the fact that the formulation size strongly increases with $I$. However, the proposed branch-and-cut algorithms are still able to provide optimal solutions for around 15% of the small size instances whereas CPLEX finds optimal solutions for less than 2% of the considered instances within the imposed time limit. Regarding medium size instances, we note that algorithms BC1 and BC2 lead to larger residual gaps than CPLEX. This might be due to the fact that, even if the best lower bounds are improved by the cutting-plane generation algorithms, the residual gap remains large due to the poor quality of the best upper bounds found within the time limit.

Table 2: Comparison between default CPLEX configuration and the customized branch-and-cut algorithms for instances with $I = 10$. Instances are grouped according to the number of nodes in the scenario tree.

| Instances | CPLEX default | | | | BC1 (Path) | | | | | BC2 (Path and Tree) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nodes | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts |
| 126 | 7.71 | 0.14 | 871.23 | 7 | 1.4 | 0.07 | 571.63 | 46 | 2042 | 1.24 | 0.07 | 541.35 | 48 | 2288 |
| 189 | 9.56 | 0.27 | 895.13 | 1 | 1.23 | 0.15 | 818.65 | 14 | 3778 | 1.09 | 0.13 | 805.72 | 15 | 4030 |
| 242 | 6.81 | 0.26 | 900.37 | 0 | 1.48 | 0.19 | 880.81 | 4 | 3057 | 1.29 | 0.19 | 871.13 | 7 | 3540 |
| 254 | 8.04 | 0.36 | 900.18 | 0 | 1.37 | 0.25 | 863.01 | 8 | 4185 | 1.21 | 0.24 | 849.03 | 11 | 4652 |
| 255 | 6.15 | 0.24 | 897.71 | 1 | 1.65 | 0.17 | 815.49 | 14 | 2761 | 1.38 | 0.15 | 784.94 | 18 | 3900 |
| 255 | 7.45 | 0.27 | 900.13 | 0 | 1.32 | 0.19 | 892.65 | 2 | 3801 | 1.18 | 0.17 | 888.76 | 3 | 4120 |
| 363 | 8.81 | 0.46 | 906.08 | 0 | 1.42 | 0.35 | 900.57 | 0 | 5985 | 1.28 | 0.31 | 900.67 | 0 | 6183 |
| 381 | 8.86 | 0.41 | 906.39 | 0 | 1.07 | 0.31 | 900.63 | 0 | 7698 | 0.96 | 0.29 | 900.25 | 1 | 7985 |
| Average | 7.92 | 0.30 | 897.49 | 9 | 1.37 | 0.21 | 830.43 | 88 | 4163 | 1.20 | 0.19 | 817.73 | 103 | 4587 |
| 468 | 7.62 | 0.51 | 900.20 | 0 | 1.93 | 0.91 | 900.70 | 0 | 6528 | 1.36 | 0.39 | 900.70 | 0 | 6683 |
| 510 | 8.34 | 0.79 | 900.24 | 0 | 1.69 | 0.83 | 900.74 | 0 | 8528 | 1.43 | 0.67 | 900.66 | 0 | 9083 |
| 511 | 6.52 | 0.44 | 900.78 | 0 | 1.62 | 0.34 | 900.68 | 0 | 5545 | 1.44 | 0.33 | 900.81 | 0 | 6742 |
| 682 | 6.74 | 0.66 | 900.25 | 0 | 2.69 | 1.60 | 900.79 | 0 | 7509 | 1.65 | 0.65 | 900.72 | 0 | 7904 |
| 728 | 10.01 | 3.20 | 903.11 | 0 | 7.03 | 6.02 | 900.87 | 0 | 9254 | 4.15 | 3.25 | 900.81 | 0 | 9827 |
| 765 | 14.01 | 5.47 | 900.65 | 0 | 14.62 | 14.05 | 900.90 | 0 | 15556 | 9.42 | 8.91 | 900.96 | 0 | 16128 |
| 777 | 9.93 | 3.51 | 905.34 | 0 | 8.83 | 7.96 | 900.85 | 0 | 10369 | 5.32 | 4.51 | 900.85 | 0 | 10578 |
| 1022 | 18.13 | 10.33 | 900.42 | 0 | 14.64 | 13.91 | 900.96 | 0 | 17131 | 14.74 | 14.14 | 900.89 | 0 | 18282 |
| Average | 10.17 | 3.11 | 901.38 | 0 | 6.63 | 5.70 | 900.81 | 0 | 10052 | 4.94 | 4.11 | 900.80 | 0 | 10653 |

In Tables 3 and 4, the instances are grouped according to the nominal returns 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 LP gap: from 1.21% (resp. 4.77%) for the case of a low volume and a poor quality of the returned products to 25.47% (resp. 15.82%) for the case of a large volume and a good quality of the returned products, for the instances with $I = 5$ (resp. $I = 10$). 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 zero. 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 to setup costs, leading to small integrality gaps. We would like to point out 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

Table 3: Comparison between default CPLEX configuration and the customized branch-and-cut algorithm for instances with $I = 5$. The instances are grouped according to the nominal returns and quality levels.

| Instances | | CPLEX default | | | | BC1 (Path) | | | | | BC2 (Path and Tree) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $R$ | $Q$ | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts |
| 1 | 1 | 1.21 | 0.22 | 894.20 | 2 | 0.62 | 0.23 | 893.37 | 2 | 4908 | 0.64 | 0.25 | 890.65 | 3 | 5010 |
| | 2 | 3.71 | 0.26 | 876.34 | 6 | 1.13 | 0.23 | 858.92 | 10 | 4401 | 1.08 | 0.22 | 839.89 | 14 | 4850 |
| | 3 | 10.89 | 0.30 | 851.54 | 12 | 2.25 | 0.18 | 770.57 | 31 | 4133 | 2.01 | 0.17 | 731.14 | 43 | 4799 |
| Average | | 5.27 | 0.26 | 874.03 | 20 | 1.33 | 0.22 | 840.95 | 43 | 4481 | 1.25 | 0.21 | 820.56 | 60 | 4886 |
| 2 | 1 | 3.38 | 0.20 | 886.32 | 5 | 0.97 | 0.17 | 856.49 | 14 | 4329 | 0.93 | 0.18 | 833.98 | 17 | 4778 |
| | 2 | 12.66 | 0.27 | 807.24 | 23 | 2.20 | 0.13 | 697.91 | 45 | 3913 | 1.85 | 0.13 | 656.96 | 53 | 4688 |
| | 3 | 22.71 | 0.47 | 839.15 | 13 | 3.21 | 0.23 | 766.87 | 32 | 3979 | 2.54 | 0.21 | 709.88 | 46 | 4785 |
| Average | | 12.91 | 0.31 | 844.24 | 41 | 2.13 | 0.18 | 773.75 | 91 | 4074 | 1.77 | 0.17 | 733.61 | 116 | 4750 |
| 3 | 1 | 6.37 | 0.10 | 758.60 | 30 | 1.25 | 0.06 | 628.28 | 62 | 3863 | 1.09 | 0.06 | 589.14 | 67 | 4566 |
| | 2 | 14.28 | 0.31 | 829.69 | 17 | 2.26 | 0.16 | 703.93 | 44 | 3858 | 1.84 | 0.15 | 656.43 | 52 | 4637 |
| | 3 | 25.47 | 0.54 | 846.76 | 11 | 3.96 | 0.3 | 764.26 | 32 | 3962 | 3.22 | 0.27 | 685.38 | 48 | 4758 |
| Average | | 15.37 | 0.32 | 811.68 | 58 | 2.49 | 0.17 | 698.82 | 138 | 3894 | 2.05 | 0.16 | 643.65 | 167 | 4654 |

Table 4: Comparison between default CPLEX configuration and the customized branch-and-cut algorithms for instances with $I = 10$. The instances are grouped according to the nominal returns and quality levels.

| Instances | | CPLEX default | | | | BC1 (Path) | | | | | BC2 (Path and Tree) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $R$ | $Q$ | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts |
| 1 | 1 | 4.77 | 3.81 | 903.00 | 0 | 6.11 | 5.81 | 900.70 | 0 | 7846 | 5.63 | 5.32 | 900.65 | 0 | 7946 |
| | 2 | 8.20 | 5.00 | 900.15 | 0 | 10.76 | 10.20 | 900.64 | 0 | 7615 | 8.17 | 7.61 | 900.66 | 0 | 8009 |
| | 3 | 8.87 | 0.67 | 900.18 | 0 | 3.64 | 2.36 | 896.34 | 2 | 7205 | 2.77 | 1.57 | 893.75 | 3 | 7758 |
| Average | | 7.28 | 3.16 | 901.11 | 0 | 6.84 | 6.12 | 899.23 | 2 | 7555 | 5.52 | 4.83 | 898.36 | 3 | 7905 |
| 2 | 1 | 4.93 | 2.49 | 912.03 | 0 | 7.11 | 6.67 | 900.64 | 0 | 7313 | 3.98 | 3.56 | 900.73 | 0 | 7728 |
| | 2 | 8.33 | 0.39 | 888.57 | 4 | 1.35 | 0.24 | 818.24 | 22 | 6791 | 1.19 | 0.22 | 805.78 | 26 | 7404 |
| | 3 | 15.87 | 0.72 | 900.27 | 0 | 2.19 | 0.4 | 861.45 | 11 | 6924 | 1.85 | 0.35 | 854.26 | 13 | 7562 |
| Average | | 9.71 | 1.20 | 900.29 | 4 | 3.55 | 2.44 | 860.11 | 33 | 7009 | 2.34 | 1.38 | 853.59 | 39 | 7564 |
| 3 | 1 | 4.86 | 0.97 | 897.64 | 1 | 0.80 | 0.16 | 844.12 | 15 | 6707 | 0.72 | 0.14 | 832.63 | 19 | 7301 |
| | 2 | 9.78 | 0.52 | 894.84 | 2 | 1.67 | 0.43 | 822.14 | 23 | 6710 | 1.32 | 0.27 | 807.64 | 26 | 7366 |
| | 3 | 15.83 | 0.80 | 896.65 | 2 | 2.36 | 0.34 | 846.3 | 15 | 6860 | 2.02 | 0.32 | 837.28 | 16 | 7507 |
| Average | | 10.14 | 0.76 | 896.38 | 5 | 1.61 | 0.31 | 837.52 | 53 | 6759 | 1.35 | 0.24 | 825.85 | 61 | 7391 |

instances corresponding to a small amount of lost sales and a large LP gap seem to be easier to solve than the instances corresponding to a large amount of lost sales and a small LP gap. 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 the lost sales penalty in the objective function is large. Namely, in this case, making a branching decision on a binary setup variable has a smaller impact on the objective function value in terms of lower bound improvement.

We note that the proposed branch-and-cut methods do not perform better than default CPLEX for the sets of instances with more than 682 nodes and $I = 10$, leading to larger residual gaps. For a better understanding of the behavior of the branch-and-cut methods BC1 and BC2, we carried out additional computational experiments over the same sets of instances after the automatic generation of cuts by default CPLEX was turned off. The results are reported in Table 5. They show that the proposed methods perform as well as default CPLEX for instances with less than 682 nodes, showing slightly larger residual gaps. Nonetheless, for instances with more than 682 nodes, the proposed methods outperform default CPLEX providing much smaller residual gaps. In general, the method BC2 (resp. BC1) improves the residual gaps from 3.11% to 0.84% (resp. 1.16%) on average and, in particular, for instances with 1022 nodes, the method is able to reduce the gap from 10.33% to 1.22% (resp. 2.61%). Additionally, we test the methods on even larger scenario trees with 1093 and 1365 nodes. The results show that the method BC2 (resp. BC1) is able to decrease the gap from 11.02% to 1.11% (resp. 1.62%) on average and for instances with 1365 nodes, the methods reduce the gap from over 13% to less than 1% (resp 1.52%) on average. A possible explanation of the improvement of the residual gaps is that turning off the default CPLEX cuts generation leads to a decrease of the size of the linear programs solved at each node of the search tree. This allows more nodes to be explored by the branch-and-bound algorithm and results in finding upper bounds of better quality. All these results confirm the usefulness of the proposed methods in tackling large instances.

Finally, it is worth mentioning that, in the computational experiments reported in this section, many sources of uncertainty are considered whereas in most previously published works, only one or two sources of uncertainty were taken into account: see e.g. [9], [10], [11], [13], [14] and [15]. We thus carried out additional computational experiments over instances involving large scenario trees in order to assess the performance of the proposed methods when only the demand and returns quantity are stochastic. The corresponding results are reported in Appendix A. They suggest the good performance of the methods as compared to the mathematical programming solver CPLEX 12.8. In particular, they show that the proposed algorithms are able to solve to optimality instances involving scenario trees with more than 1800 nodes in a reasonable computation time.

### *6.3. Rolling horizon simulation*

In this subsection, we seek to assess the practical performance of the multi-stage stochastic programming model by comparing it with two simpler production planning models: a deterministic model which completely ignores uncertainty and a two-stage stochastic programming model which considers uncertainty but does not allow to dynamically adjust production decisions over time. To build the two-stage stochastic programming model, we made the same

Table 5: Comparison between default CPLEX configuration and the customized branch-and-cut algorithms without automatically generated cuts embedded in CPLEX for instances with $I = 10$. Instances are grouped according to the number of nodes in the scenario tree.

| Instances | CPLEX default | | | | BC1 (Path) | | | | | BC2 (Path and Tree) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nodes | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts | $Gap_{LP}$ | $Gap_{MIP}$ | Time | #Opt | Cuts |
| 126 | 7.71 | 0.14 | 871.23 | 7 | 1.40 | 0.23 | 900.63 | 0 | 2042 | 1.24 | 0.15 | 866.06 | 8 | 2289 |
| 189 | 9.56 | 0.27 | 895.13 | 1 | 1.22 | 0.32 | 900.66 | 0 | 3778 | 1.09 | 0.24 | 892.70 | 2 | 4030 |
| 242 | 6.81 | 0.26 | 900.37 | 0 | 1.48 | 0.49 | 900.54 | 0 | 3057 | 1.29 | 0.36 | 900.60 | 0 | 3539 |
| 254 | 8.04 | 0.36 | 900.18 | 0 | 1.37 | 0.52 | 900.66 | 0 | 4184 | 1.21 | 0.40 | 900.68 | 0 | 4653 |
| 255 | 6.15 | 0.24 | 897.71 | 1 | 1.66 | 0.67 | 900.68 | 0 | 2761 | 1.40 | 0.44 | 900.63 | 0 | 3900 |
| 255 | 7.45 | 0.27 | 900.13 | 0 | 1.33 | 0.46 | 900.66 | 0 | 3801 | 1.19 | 0.36 | 900.56 | 0 | 4119 |
| 363 | 8.81 | 0.46 | 906.08 | 0 | 1.45 | 0.74 | 900.67 | 0 | 5984 | 1.31 | 0.62 | 900.64 | 0 | 6182 |
| 381 | 8.86 | 0.41 | 906.39 | 0 | 1.09 | 0.60 | 900.75 | 0 | 7699 | 0.97 | 0.50 | 900.75 | 0 | 7985 |
| Average | 7.92 | 0.30 | 897.49 | 9 | 1.37 | 0.50 | 900.66 | 0 | 4163 | 1.21 | 0.38 | 895.33 | 10 | 4587 |
| 468 | 7.62 | 0.51 | 900.20 | 0 | 1.51 | 0.85 | 900.78 | 0 | 6528 | 1.40 | 0.71 | 900.70 | 0 | 6682 |
| 510 | 8.34 | 0.79 | 900.24 | 0 | 1.35 | 0.88 | 900.71 | 0 | 8527 | 1.21 | 0.75 | 900.62 | 0 | 9082 |
| 511 | 6.52 | 0.44 | 900.78 | 0 | 1.69 | 0.96 | 900.77 | 0 | 5545 | 1.47 | 0.78 | 900.64 | 0 | 6741 |
| 682 | 6.74 | 0.66 | 900.25 | 0 | 1.61 | 0.94 | 900.85 | 0 | 7509 | 1.50 | 0.87 | 900.75 | 0 | 7904 |
| 728 | 10.01 | 3.20 | 903.11 | 0 | 1.63 | 0.98 | 900.87 | 0 | 9253 | 1.47 | 0.86 | 900.98 | 0 | 9826 |
| 765 | 14.01 | 5.47 | 900.65 | 0 | 1.53 | 1.19 | 901.11 | 0 | 15549 | 1.08 | 0.73 | 901.13 | 0 | 16121 |
| 777 | 9.93 | 3.51 | 905.34 | 0 | 1.47 | 0.88 | 900.86 | 0 | 10369 | 1.40 | 0.83 | 900.89 | 0 | 10578 |
| 1022 | 18.13 | 10.33 | 900.42 | 0 | 2.96 | 2.61 | 900.99 | 0 | 17129 | 1.55 | 1.22 | 901.53 | 0 | 18279 |
| Average | 10.17 | 3.11 | 901.38 | 0 | 1.72 | 1.16 | 900.87 | 0 | 10051 | 1.38 | 0.84 | 900.91 | 0 | 10652 |
| 1093 | 13.45 | 8.45 | 900.48 | 0 | 2.46 | 1.72 | 900.98 | 0 | 7950 | 1.94 | 1.24 | 900.82 | 0 | 9831 |
| 1365 | 17.90 | 13.59 | 900.51 | 0 | 2.26 | 1.52 | 900.80 | 0 | 7690 | 1.74 | 0.98 | 900.89 | 0 | 9479 |
| Average | 15.67 | 11.02 | 900.49 | 0 | 2.36 | 1.62 | 900.89 | 0 | 7820 | 1.84 | 1.11 | 900.85 | 0 | 9655 |

assumptions as the ones used by Macedo *et al.* [12] for planning a hybrid manufacturing/remanufacturing system under stochastic demand, returns and set-up costs. More precisely, we considered the production decisions (production quantity and set-up) as first-stage variables and all other decisions (inventory level, lost sales and discarded quantities) as second-stage variables.

This assessment is achieved by carrying out a rolling horizon simulation similar to the one used by Brandimarte [24]. Each experiment consists in simulating the application of the first-stage planning decisions over 12 time periods and in recording the total cost incurred when applying the planning decisions established by the deterministic or the stochastic models. Note that the cost considered in this 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 planning decisions over a true scenario. We then compute the relative difference $100(C - C_{MS})/C_{MS}$, where $C$ is the total costs accumulated over each simulation run for the deterministic or the two-stage stochastic programming model, and $C_{MS}$ corresponds to the total costs accumulated over each simulation run for the multi-stage stochastic programming model.

The instances are generated in the same way as in the previous subsection. Since running the simulation is

Table 6: 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 were calculated as follows: $100(C - C_{MS})/C_{MS}$.

| | | Deterministic model | | | | | | Two-stage stochastic model | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instances | | c=2 | | c=3 | | c=4 | | c=2 | | c=3 | | c=4 | |
| $R$ | $Q$ | Ave. | MAD | Ave. | MAD | Ave. | MAD | Ave. | MAD | Ave. | MAD | Ave. | MAD |
| 1 | 1 | 19.9 | 7.7 | 25.4 | 9.6 | 24.6 | 8.9 | -1.39 | 15.07 | -1.90 | 21.68 | -2.97 | 28.27 |
| | 2 | 36.1 | 18.5 | 33.2 | 16.9 | 36.2 | 17.6 | 16.13 | 16.35 | 15.82 | 13.77 | 20.29 | 19.30 |
| | 3 | 43.4 | 41.1 | 47.8 | 42.6 | 65.2 | 54.6 | 20.79 | 25.41 | 40.18 | 35.22 | 44.45 | 33.44 |
| 2 | 1 | 26.7 | 14.5 | 31.0 | 15.6 | 34.3 | 18.6 | 5.93 | 13.62 | 11.59 | 17.17 | 9.50 | 21.00 |
| | 2 | 74.3 | 61.9 | 66.5 | 54.7 | 74.1 | 56.6 | 12.10 | 21.56 | 21.87 | 24.48 | 28.79 | 24.08 |
| | 3 | 52.0 | 70.5 | 69.2 | 89.7 | 39.8 | 48.5 | 9.65 | 15.93 | 15.27 | 15.94 | 22.72 | 18.41 |
| 3 | 1 | 37.1 | 31.8 | 34.0 | 25.8 | 45.5 | 39.8 | 21.76 | 20.26 | 29.39 | 21.10 | 27.36 | 21.89 |
| | 2 | 60.2 | 62.1 | 46.0 | 44.5 | 65.4 | 69.5 | 15.00 | 18.58 | 23.33 | 19.90 | 46.12 | 29.39 |
| | 3 | 33.2 | 55.9 | 46.1 | 68.6 | 50.0 | 70.7 | 7.94 | 14.10 | 18.78 | 17.41 | 27.16 | 19.83 |
| Average | | 42.5 | 42.5 | 44.3 | 42.2 | 48.2 | 44.8 | 11.99 | 17.88 | 19.37 | 20.74 | 24.82 | 23.96 |

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 algorithm BC2 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 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 6 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.

Results from Table 6 suggest that the actual practical performance of the production plan obtained by the multi-stage stochastic model might be significantly better than the one of the production plan provided by a deterministic model. Namely, the average increase in the actual production cost observed when using the deterministic model instead of the multi-stage 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 children per end-of-stage node are used. We note that this cost decrease mainly comes from the decrease in the amount of lost sales penalty costs obtained when using the multi-stage stochastic model. Regarding the two-stage stochastic model, the results also suggest an overall better performance of the multi-stage stochastic model. Indeed, the average increase in the actual production cost observed when using the two-stage stochastic model instead of the multi-stage stochastic model is almost 19%. We note however an exception for the instances with the lowest returns and quality level for which the two-stage stochastic model slightly outperforms the multi-stage stochastic model. A reasonable explanation for this is that, with low returns of bad quality, there will be a large amount of lost sales and the positive impact on the costs of the additional flexibility to adjust production decisions over time offered by the multi-stage stochastic model is diminished.

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 relies 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. Although the proposed branch-and-cut methods provide small residual gaps for large-size scenario tree instances, they were not able to solve them to optimality within the imposed time limit. 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 preliminary 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 Mathématique Jacques Hadamard (FMJH). The authors would also like to thank three anonymous referees for their detailed reviews that helped to improve an initial version of this paper.

## References

## 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] M. Denizel, M. Ferguson, et al., Multiperiod remanufacturing planning with uncertain quality of inputs, IEEE Transactions on Engineering Management 57 (3) (2010) 394–404.
[7] M. Kazemi Zanjani, M. Nourelfath, D. Ait-Kadi, A multi-stage stochastic programming approach for production planning with uncertainty in the quality of raw materials and demand, International Journal of Production Research 48 (16) (2010) 4701–4723.
[8] C. Li, F. Liu, H. Cao, Q. Wang, A stochastic dynamic programming based model for uncertain production planning of re-manufacturing system, International Journal of Production Research 47 (13) (2009) 3657–3668.
[9] O. A. Kilic, A mip-based heuristic for the stochastic economic lot sizing problem with remanufacturing, IFAC Proceedings Volumes 46 (9) (2013) 742–747.
[10] 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 267 (3) (2018) 1102–1109.
[11] 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.
[12] 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.
[13] T. Hilger, F. Sahling, H. Tempelmeier, Capacitated dynamic production and remanufacturing planning under demand and return uncertainty, OR spectrum 38 (4) (2016) 849–876.
[14] 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.
[15] 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.
[16] 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.
[17] Y. Guan, S. Ahmed, G. L. Nemhauser, Cutting planes for multistage stochastic integer programs, Operations research 57 (2) (2009) 287–298.
[18] M. Di Summa, L. A. Wolsey, Lot-sizing on a tree, Operations Research Letters 36 (1) (2008) 7–13.
[19] 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.
[20] M. Loparic, Y. Pochet, L. A. Wolsey, The uncapacitated lot-sizing problem with sales and safety stocks, Mathematical Programming 89 (3) (2001) 487–504.
[21] 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.
[22] Y. Pochet, L. A. Wolsey, Production planning by mixed integer programming, Springer Science & Business Media, 2006.
[23] 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.
[24] P. Brandimarte, Multi-item capacitated lot-sizing with demand uncertainty, International Journal of Production Research 44 (15) (2006) 2997–3022.

## Appendix A. Additional computational results on large instances with only two sources of uncertainty

In Subsection 6.2, we study the performance of the proposed algorithms over a set of instances where many sources of uncertainty, namely the returns quality and quantity, the demand and the costs, are taken into account. However, most of the works carried out so far considered only one or two sources of uncertainty and focused on a stochastic demand and/or a stochastic returns quantity: see e.g. [9], [10], [11], [13], [14] and [15]. In this Appendix, we thus seek to assess the effectiveness of the proposed branch-and-cut algorithms over a set of instances in which only two sources of uncertainty, namely the demand and the returns quantity, are taken into account. In particular, we would like to evaluate whether reducing the level of stochasticity in the problem parameters could enable us to solve instances involving larger scenario trees.

In what follows, we introduce the settings used to randomly generate this second set of instances and present the results of the related computational experiments.

### *Appendix A.1. Instances generation*

We randomly generate instances following the scheme presented in Subsection 6.1. The stochastic parameters, i.e. the demand and returns quantity, are generated as described in Subsection 6.1. All the other parameters may be time varying but are assumed to be deterministically known. Hence, contrary to what is done in Subsection 6.1, these parameters now have the same value for each node of the scenario tree belonging to the same time period. This value is randomly generated from the same discrete uniform distributions as the ones used in Subsection 6.1.

We set the number of parts in a product to $I = 5$ and considered 3 scenario tree structures: $(b, s, c) = ...$, $(b, s, c) = ...$ and $(b, s, c) = ...$leading to trees involving respectively $|\mathcal{V}| = 1093$, $|\mathcal{V}| = 1365$ and $|\mathcal{V}| = 1820$ nodes. For each combination of scenario tree structure and used product quantity level, we randomly generated 30 instances, resulting in a total of 270 instances.

### *Appendix A.2. Results*

Each instance was solved using formulation (14)-(28) by the three alternative branch-and-cut methods detailed in Subsection 6.2. The corresponding results are provided in Table A.7. They suggest that the proposed algorithms also outperform CPLEX for this second set of large instances. This can be seen by the fact that the number of instances solved to optimality is much higher than the one solved by CPLEX when using Algorithm BC1 or BC2. Specifically, Algorithm BC2 (resp. BC1) can solve almost 25% (resp. 17%) of the tested instances to optimality whereas CPLEX can solve less than 1.2% of these instances to optimality. Moreover, the average residual gap is decreased from 0.56% with CPLEX to 0.28% with BC1 and 0.19% with BC2 and the average computation time is decreased from 892.73 with CPLEX to 808.17 with BC1 and 745.25 with BC2. Moreover, we note that Algorithm BC2 performs at least as well as Algorithm BC1 over all instance sets, in terms of both reducing the computation time and obtaining optimal solutions.

Table A.7: Performance of branch-and-cut algorithm over larger instances. The instances are grouped according to the number of nodes in the scenario tree and the nominal returns levels.

| Instances | | CPLEX default | | | | BC1 (Path) | | | | BC2 (Path and Tree) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Nodes* | *R* | $Gap_{LP}$ | $Gap_{MIP}$ | *Time* | *#Opt* | $Gap_{LP}$ | $Gap_{MIP}$ | *Time* | *#Opt* | $Gap_{LP}$ | $Gap_{MIP}$ | *Time* | *#Opt* |
| 1093 | 1 | 8.05 | 0.21 | 900.18 | 0 | 2.41 | 0.09 | 841.87 | 6 | 2.05 | 0.06 | 696.38 | 9 |
| | 2 | 23.90 | 0.55 | 900.21 | 0 | 6.07 | 0.28 | 708.68 | 8 | 4.82 | 0.22 | 639.78 | 12 |
| | 3 | 26.74 | 0.68 | 858.23 | 2 | 7.33 | 0.27 | 756.66 | 6 | 6.12 | 0.20 | 710.99 | 8 |
| 1365 | 1 | 6.23 | 0.11 | 900.14 | 0 | 1.88 | 0.04 | 758.91 | 9 | 1.65 | 0.03 | 709.35 | 9 |
| | 2 | 25.66 | 0.83 | 871.56 | 1 | 7.60 | 0.37 | 778.85 | 7 | 6.11 | 0.23 | 725.53 | 7 |
| | 3 | 28.82 | 1.14 | 900.24 | 0 | 10.15 | 0.74 | 897.16 | 1 | 8.48 | 0.50 | 831.44 | 4 |
| 1820 | 1 | 19.29 | 0.59 | 900.19 | 0 | 2.18 | 0.48 | 901.64 | 0 | 1.91 | 0.28 | 901.20 | 0 |
| | 2 | 34.62 | 0.49 | 901.82 | 0 | 2.11 | 0.10 | 810.30 | 4 | 1.79 | 0.07 | 712.60 | 10 |
| | 3 | 34.04 | 0.43 | 902.00 | 0 | 2.03 | 0.12 | 819.63 | 4 | 1.70 | 0.10 | 780.27 | 6 |
| Average | | 23.04 | 0.56 | 892.73 | 3 | 4.64 | 0.28 | 808.17 | 45 | 3.78 | 0.19 | 745.28 | 65 |