Cutting planes generation and decomposition-based approaches for solving multi-stage stochastic lot-sizing problems

Franco Quezada Valenzuela

Cutting planes generation and decomposition-based approaches for solving multi-stage stochastic lot-sizing problems

par Franco Quezada Valenzuela

Thèse de doctorat en informatique

Jury :

Kerem Akartunali	rapporteur	Professeur, University of Strathclyde
François Clautiaux	examinateur	Professeur, Université de Bordeaux
Stéphane Dauzère-Pérès	rapporteur	Professeur, Ecole des Mines de Saint-Étienne
Céline Gicquel	co-encadrante	MCF, Université Paris-Sud
Safia Kedad-Sidhoum	directrice	Professeure, Conservatoire National des Arts et Métiers
Vincent Leclere	examinateur	MCF, Ecole des Ponts ParisTech
Hande Yaman	examinatrice	Professeure, Katholieke Universiteit Leuven

Contents

Part I

Introduction and preliminaries

Chapter 1

Introduction

1.1 Context

Industrial companies now face a growing pressure to establish sustainability in their manufacturing operations by reducing both their CO2 emissions and natural resources consumption. One possible way of achieving sustainable business development in manufacturing and logistics is through the development of closed-loop supply chains and the use of reverse logistics.

Traditional forward supply chains mainly deal with material flows going from the suppliers providing raw materials up to the retailers selling finished products to customers. On the contrary, reverse logistics consists in the set of logistics and rehabilitation activities starting from used products that are no longer required by the users to products that are once again usable by customers. It encompasses a variety of activities such as collecting the used products, sorting, disassembling, remanufacturing, recycling and redistributing.

In particular, remanufacturing industrial products once they have reached their end-of-life is an interesting alternative to mitigate their environmental impact. 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 [67]. The reuse of materials and components embedded in used products contributes both to the reduction of pollution emissions and natural resource consumption.

However, as compared to classical manufacturing systems which produce end-products from virgin raw materials and new components, remanufacturing systems involve several complicating features, including 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 remanufactured products. Neglecting this aspect when managing remanufacturing operations may lead among others to a loss of economical performance through e.g. inventory shortages and lost sales.

The present work studies one of the many problems to be addressed by industrial companies when managing their remanufacturing operations: production planning. Within a remanufacturing context, production planning consists in deciding about the products to be disassembled, refurbished and reassembled, the timing and level of production as well as the resources to be used so as to meet the customers' demand for the remanufactured products in the most efficient and economical possible way. Industrial managers may be overwhelmed by the complexity of this optimization problem.

Our main objective in this thesis is thus to develop mathematical models and algorithms which could ultimately form the basis of decision support tools enabling industrial managers to efficiently plan production activities for complex remanufacturing production systems under uncertainty. To this aim, we consider a remanufacturing system involving three processing echelons: disassembly of used products into parts, refurbishing of the recovered parts and reassembly into like-new products. We investigate the problem of planning remanufacturing activities in this system over a finite and

discrete-time planning horizon when there are uncertainties on the quantity and quality of returned products, on the customers' demand, and on the various production costs.

Production planning problems are particularly challenging when the production costs to be taken into account involve fixed setup costs corresponding to setup operations (e.g. machine calibration, cleaning, tool change) which have to be carried out on a machine before production. In this case, determining the size of the production lots to be processed at each planning period may be a complex decision. Namely, as a naive perception, to reduce these fixed setup costs, whose value does not depend on the quantity of products processed, 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, taking into account both the customers' demand satisfaction and the practical limitations of the production system. In the present work, we thus propose to extend previously published stochastic lot-sizing models in order to build efficient production plans for the considered three-echelon remanufacturing system under uncertainty on the input data.

Various approaches for optimization under uncertainty have been proposed in the literature. Among them, stochastic programming has proved its effectiveness when tackling real world problems with some unknown parameters. Stochastic programming approaches rely on the fact that even if the exact value of the problem parameters cannot be perfectly known at the time the decision has to be made, some knowledge about their possible value is available in the form of a probability distribution. The basic idea of stochastic optimization models is thus to take advantage of this knowledge to somehow identify the "best possible decision" to be made. The reader is referred to [20] for an excellent introduction to this field.

Several stochastic programming approaches have been investigated to handle lot-sizing under uncertainty. They differ among others with respect to the number of stages involved in the decision process. In the context of stochastic programming, a stage in the decision process can be basically defined as a future point in time at which new information on the stochastic parameters becomes available and decisions based on this newly available information have to be made. In this thesis, we focus on the development of multi-stage stochastic programming approaches as this seems to be better suited to model stochastic production planning problems. Indeed, in practice, production planning is usually carried out within a rolling horizon framework. We thus compute a production plan for an horizon spanning T periods but only implement the decisions relative to the first $T' < T$ periods. After the end of period T', we update the inventory level and the demand forecasts and recompute a new production plan for periods $T'+1...T'+T'$ in which the decisions relative to periods $T'+1...T$ can be modified. Hence, production planning is intrinsically a multi-stage decision process in which production decisions are not made once and for all but rather adjusted over time according to the actual realizations of the uncertain parameters.

Over the last decades, multi-stage stochastic programming has received a lot of attention from scholars and practitioners. As a result, important progress has been made both on theoretical aspects and on the development of effective and efficient solution algorithms. Nonetheless, most of this work has been restricted to a linear setting. Consequently, multi-stage stochastic integer programs such as the ones obtained in our study are still very challenging to solve. Thus, a large body of the work presented in the present manuscript deals with the development of efficient and scalable solution approaches for lot-sizing and remanufacturing planning under uncertainty.

1.2 Thesis contributions

The main purpose of the work presented here is to develop mathematical models and algorithms that may enable industrial managers to efficiently plan production activities for complex remanufacturing production systems under uncertainty.

In order to achieve this, we consider a remanufacturing system involving three processing echelons. At the first echelon, the used products returned by customers are first disassembled into parts, some of which are discarded due to their usage state. The recoverable parts are then refurbished on dedicated resources. Finally, the refurbished 'serviceable' parts are re-assembled into like-new products to satisfy the customers' demand. We investigate the problem of planning remanufacturing activities in this system over a finite and discrete-time planning horizon when there are uncertainties on the quantity and quality of returned products, on the customers' demand, and on the various production costs. We propose to model this stochastic combinatorial optimization problem as a multi-stage stochastic integer program and to use a scenario tree to represent the evolution of the stochastic input process over time. This leads to the formulation of a large-size mixed-integer linear program.

The development of a multi-stage stochastic integer programming approach for such a stochastic multi-echelon multi-item lot-sizing problem arising in a remanufacturing context can be seen as a first contribution of this work. Namely, most previously published works on lot-sizing for remanufacturing considered either a deterministic setting or develop two-stage stochastic programming approaches. Moreover, most papers considering a multi-stage decision process for stochastic lot-sizing problems assume much simpler production planning settings involving only a single echelon or/and a single item. Thus, to the best of our knowledge, this is one of the first works seeking to simultaneously take into account a realistic production setting and a multi-stage stochastic decision process for planning remanufacturing activities.

Furthermore, even if the problem under study can be formulated as a mixed-integer linear program (MILP), its direct resolution by a mathematical programming solver poses some computational difficulties in practice. The first difficulty 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, the size of the obtained MILP is proportional to the number of nodes in the scenario tree, which grows exponentially fast with the number of stages and the number of realizations per stage taken into account in the scenario tree. The second difficulty is linked to the presence of many big-M type constraints in the formulation, resulting in a poor quality of the lower bounds provided by the linear relaxation of the problem and a loss of efficiency of the generic branch-and-cut algorithms embedded in mathematical programming solvers.

We thus investigate in this thesis two kinds of solution approaches for this problem.

We first focus on enhancing the performance of the generic branch-and-cut algorithms embedded in mathematical programming solvers through the use of cutting plane generation approaches. The main idea here is to improve the quality of the lower bound used by the algorithm to make branching and cutting off decisions at each node of the branch-and-bound search tree. This improvement is achieved through the dynamic addition of problem-specific valid inequalities into the MILP formulation via a cutting-plane generation algorithm run at each node of the search tree. Here, our contributions are twofold:

- We exploit and extend previously known inequalities proposed in [66] for the deterministic single-item single-echelon lot-sizing problem with lost-sales. These inequalities can be used as such or can be mixed with one another to obtain stronger inequalities expressed on sub-trees of the scenario tree. We provide an efficient cutting-plane generation strategy to identify the useful subset of inequalities to be added at each node of the search tree and develop two customized branch-and-cut algorithms. Our computational experiments show that the proposed method is capable of significantly decreasing the computation time needed to obtain guaranteed optimal solutions for small to medium-size instances of the problem.

- We try to go one step further by developing several families of new valid inequalities for our problem. These valid inequalities seek to take into account two important aspects of our problem which are overlooked by the inequalities proposed in [66]: the fact that our production

system comprises multiple echelons and the fact that the production on a resource is limited by the availability of the used products returned by customers. In view of the theoretical and numerical difficulties lying ahead, we focus on the deterministic variant of the remanufacturing planning problem under investigation. Our numerical results enable us to identify a family of valid inequalities which are efficient at strengthening the formulation of the deterministic problem and which would thus be promising candidates for further improving our customized branch-and-cut algorithms for the stochastic problem.

Second, in order to solve larger instances of our problem, we investigate a second kind of solution approach based on a decomposition of the original problem into a series of small sub-problems linked together by dynamic programming equations. Each of these sub-problems focuses on making decisions for a small subset of nodes belonging to the same scenario and the same decision stage, taking into account not only the current cost of these decisions but also their future cost which is represented by an expected cost-to-go function. The Stochastic Dual Dynamic integer Programming (SDDiP) algorithm recently published in [105] provides a way of iteratively building a piecewise linear approximation for each of the expected cost-to-to functions involved in the decomposition. Our main contributions here consist in two extensions of this algorithm to improve its numerical efficiency on lot-sizing problems:

- Considering the complexity and novelty of the SDDiP algorithm, we first study how to use it to solve the stochastic single-item uncapacitated lot-sizing (SULS) problem, which can be seen as the simplest and most investigated multi-stage stochastic lot-sizing problem. We propose an extension of this algorithm which mainly relies on a partial decomposition of the scenario tree into sub-trees and on the exploitation of existing knowledge on the polyhedral structure of the SULS problem. Our computational results show that the proposed extended SDDiP algorithm significantly outperforms the initial SDDiP algorithm in terms of solution quality on large-size instances .

- We then come back to our initial stochastic remanufacturing planning problem and investigate how the extended SDDiP algorithm may be adapted to provide good-quality solutions to this more complex lot-sizing problem.

Finally, we present an on-going exploratory work on risk-averse multi-stage stochastic lot-sizing. Namely, risk-neutral models based on the minimization of the expected costs may lead to production plans displaying a good performance on average, but providing very poor results, i.e. production costs much higher than the expected value, in some unfavorable scenarios. In case the production manager is more concerned about these potential large monetary losses than about the average performance of the production plan, it might be useful to introduce risk measures in the problem modeling. We thus study several ways of incorporating risk aversion in the multi-stage stochastic uncapacitated lot-sizing (SULS) problem and show how the risk-averse SULS can be reformulated as a mixed-integer linear program in each case. We finally provide some preliminary results seeking to assess the practical usefulness of using a risk-averse formulation rather than a risk neutral formulation.

1.3 Thesis organization

The manuscript comprises four parts.

Part I comprises two chapters: the present introductory chapter and Chapter 2. This latter introduces the necessary background on lot-sizing and multi-stage stochastic programming to understand the work presented here.

Part II presents our work focusing on improving the performance of implicit enumeration methods through cutting plane generation algorithms. Chapter 3 first provides a detailed description of the

stochastic remanufacturing planning problem which is at the core of this thesis. It then presents the proposed cutting-plane generation algorithms based on the valid inequalities introduced in [66]. Chapter 4 investigates three new sets of valid inequalities that may be used to further strengthen the MILP formulation of the deterministic variant of our remanufacturing planning problem and identifies one promising candidate family of valid inequalities.

Part III is dedicated to the development of decomposition-based methods to solve very large instances of stochastic lot-sizing problems. In Chapter 5, we turn out attention to the resolution of the stochastic uncapacitated single-echelon lot-sizing problem (SULS) through a new extension of the SDDiP algorithm. Chapter 6 focuses on adapting this extended SDDiP algorithm to our initial stochastic remanufacturing planning problem.

Part IV comprises a single chapter, Chapter 7. This one provides a preliminary report of our on-going work on risk-averse multi-stage stochastic lot-sizing.

Finally, Chapter 8 provides a general conclusion and discusses some directions for further research.

It is worth mentioning that, at the end of this manuscript, we provide a list of notations used throughout the manuscript. These are organized following the structure of the manuscript and grouped by studied problems and solution approaches. We thus invite the reader to consult it when necessary during the reading.

Chapter 2

Preliminaries

This chapter provides some fundamental knowledge about lot-sizing and multi-stage stochastic programming. It introduces the necessary background on the models and solution approaches which form the starting point of this thesis and will thus be frequently used throughout this dissertation. In order to ease the understanding, we focus our explanations on the simplest available lot-sizing model, which is the single-item Uncapacitated Lot-Sizing problem (ULS).

2.1 Deterministic lot-sizing

In the ULS, we consider a production system involving a single production resource and assume that this resource transforms raw materials into a single type of item through a single processing step. The ULS aims at planning the production of this item over a finite discrete-time planning horizon involving a set $\mathcal{T} = \{1...T\}$ of periods. Producing a positive amount in period $t \in \mathcal{T}$ incurs a fixed set-up cost f_t together with a production cost g_t per unit produced and an inventory holding cost h_t per unit held in stock between two consecutive periods. The objective is to build a production plan such that the customers' demand d_t is met in each time period t and the total costs, i.e. the sum of setup, production, and inventory holding costs over the whole planning horizon, are minimized.

This problem can be formulated as a mixed-integer linear program by introducing the following decision variables:

- X_t: quantity produced in period t for $t \in \mathcal{T}$.
- Y_t: set-up state of the resource. $Y_t = 1$ if the resource is set-up to produce the item in t, $Y_t = 0$ otherwise, for $t \in \mathcal{T}$.
- S_t: inventory level at the end of period t for $t \in \mathcal{T}$.

With this notation, the ULS can be formulated as follows:

$$
\begin{cases}
Z^* = \min \sum_{t=1}^{T} \left(f_t Y_t + g_t X_t + h_t S_t \right) & & (2.1) \\
X_t \leq M_t Y_t & \forall t \in \mathcal{T} & (2.2) \\
S_t = S_{t-1} + X_t - d_t & \forall t \in \mathcal{T} & (2.3) \\
X_t \geq 0 & \forall t \in \mathcal{T}. & (2.4) \\
S_t \geq 0 & \forall t \in \mathcal{T}. & (2.5) \\
Y_t \in \{0, 1\} & \forall t \in \mathcal{T}. & (2.6)
\end{cases}
$$

The objective function (2.1) minimizes the sum of the set-up, production and inventory holding costs over the whole planning horizon. Constraints (2.2) ensure that, if production takes place in

period t, the corresponding setup costs are incurred. Note that the value of constant M_t can be set to the value of the cumulative remaining demand to be satisfied till the end of the horizon, i.e. to $\sum_{\tau=t}^{T} d_\tau$. Constraints (2.3) are the inventory balance constraints. Together with Constraints (2.5), they ensure the timely satisfaction of the demand.

The ULS is known to be solvable in polynomial time. A simple dynamic programming algorithm was proposed by Wagner and Within [100]. It is based on the zero-inventory-ordering property, i.e. production is undertaken in a period only if the entering inventory level drops to zero, and runs in $\mathcal{O}(T^2)$ time. This time complexity was later improved to $\mathcal{O}(T \log T)$ in [3] and [99]. Moreover, Barany et al. [16] proposed a family of valid inequalities, known as the $(\ell, \mathscr{S})$ inequalities. These inequalities, when added to Constraints (2.2)-(2.5), provide a full description of the convex hull of the feasible space of ULS.

Many extensions of the ULS have been proposed since the seminal work of Wagner and Within [100] in order to improve its applicability in real-life situations by taking into account complicating features relative among others to the costs, the production resource or the demand service policy. We refer the reader to [22] and [21] for comprehensive surveys on the single-item dynamic lot-sizing problem and to [35], [57] and [24] for introductions on multi-item dynamic lot-sizing problems.

In the present work, we focus on an extension of the ULS, denoted SULS, in which the input parameters (cost and demand) are subject to uncertainty. Namely, in practice, these parameters, which are relative to the future, are often estimated through a forecasting procedure so that their value is not perfectly known at the time the production plan should be built. Hence, in many applications, it is necessary to handle lot-sizing as an optimization problem involving uncertainty.

A wide variety of approaches have been proposed to handle production planning and lot-sizing under uncertainty: see [13] for a general overview and [96] and [21] for literature reviews focusing on single-item dynamic lot-sizing problems. In what follows, we assume that an accurate probabilistic description of the random input parameters is available under the form of probability distributions and focus on the development of multi-stage stochastic programming approaches for lot-sizing under uncertainty.

2.2 Multi-stage stochastic lot-sizing

2.2.1 Nested formulation

Many practical decision problems such as lot-sizing problems involve making a sequence of decisions that react to outcomes evolving over time with only partial information about the future conditions (such as unknown demands, costs, inventory reservations ...). Multi-stage stochastic integer programming (MSIP) is a framework for sequentially making decision under uncertainty of future conditions ([95], [20]).

In the context of stochastic programming, a stage in the finite horizon sequential decision process can be basically defined as a future point in time at which new information on the stochastic input parameters becomes available and decisions based on this newly available information have to be made. Note that stages do not necessarily coincide with planning periods, in particular a stage may comprise several periods. This is of particular interest for lot-sizing problems as the time discretization used by the decision-makers to plan production activities is indeed usually finer than the one used to update the demand and cost forecasts and readjust the production plan. A planning period may thus typically correspond to an 8-hours shift or a day whereas a stage may correspond to a week or a month. We thus consider a decision process involving Σ decision stages and denote by $\mathcal{S} = \{1, ..., \Sigma\}$ the set of stages. A stage may correspond to one or several consecutive planning periods. Let $\mathcal{T}^\sigma$ be the set of time periods belonging to stage $\sigma \in \mathcal{S}$: $t^F(\sigma)$ (resp. $t^L(\sigma)$) denotes the first (resp. the last) time period belonging to stage σ. Note that the sets $\{\mathcal{T}^\sigma, \sigma \in \mathcal{S}\}$ form a partition of $\mathcal{T}$.

The basic scheme of dynamic decision process in which observations and decisions are alternated has the following form

$$\begin{aligned}
&\text{Observation of random parameters for stage } 1 \rightarrow \text{ Decision for stage } 1\\
&\cdots \text{ Observation of random parameters for stage } \sigma \rightarrow \text{ Decision for stage } \sigma\\
&\qquad\qquad \cdots \text{ Observation of random parameters for stage } \Sigma \rightarrow \text{ Decision for stage } \Sigma
\end{aligned}$$

At stage σ, we thus make production decision $x_\sigma = \{(X_t, Y_t, S_t), t \in \mathcal{T}^\sigma\}$ based only on the information available at stage σ. We assume that this information is given by a discrete time stochastic process $\{\xi_\sigma\}_{\sigma=1}^{\Sigma}$ where $\xi_\sigma = \{(f_t, g_t, h_t, d_t), t \in \mathcal{T}^\sigma\}$ describes the realization of the stochastic costs and demand for all periods belonging to stage σ. Thus, the information available at stage σ consists of the past realization of the random vector up to stage σ, $\xi_{[1,\sigma]} = \{\xi_1, \ldots, \xi_\sigma\}$, and the stochastic decision x_σ at stage σ should depend only on $\xi_{[1,\sigma]}$. This property is called *non-anticipativity*.

The feasible region for the production decisions to be made at stage σ depends on the decision $x_{\sigma-1}$ made at stage $\sigma - 1$ and on the information ξ_σ available at the beginning of stage σ. Let us denote it by $F_\sigma(x_{\sigma-1}, \xi_\sigma)$. We have:

$$\begin{aligned}
F_\sigma(x_{\sigma-1}, \xi_\sigma) = \{(X_t, Y_t, S_t), t \in \mathcal{T}^\sigma \text{ s.t.} \quad & X_t \leq M_t Y_t && \forall t \in \mathcal{T}^\sigma\\
& S_t = S_{t-1} + X_t - d_t && \forall t \in \mathcal{T}^\sigma\\
& S_{t^F(\sigma)-1} = S_{t^L(\sigma-1)}\\
& X_t, S_t \geq 0, Y_t \in \{0,1\} && \forall t \in \mathcal{T}^\sigma\}
\end{aligned}$$

Note how the entering stock at stage σ, $S_{t^F(\sigma)-1}$, is imposed by the leaving stock at stage $\sigma - 1$, $S_{t^L(\sigma-1)}$. As for the cost relative to stage σ, we denote it by $C_\sigma(x_\sigma, \xi_\sigma) = \sum_{t \in \mathcal{T}^\sigma}(f_t Y_t + g_t X_t + h_t S_t)$. Finally, we use $\boldsymbol{\xi}_{[\sigma,\sigma']}$ to denote the sequence of random data vectors corresponding to stages σ through σ' and $\xi_{[\sigma,\sigma']}$ to denote a specific realization of this sequence of random vectors.

Using this notation, the decision dynamics described above leads to the following nested formulation.

$$\begin{aligned}
\min_{x_1 \in F_1} \Big\{ C(x_1, \xi_1) + \mathbb{E}_{\boldsymbol{\xi}_{[2,\Sigma]}|\xi_{[1,1]}} \Big[\min_{x_2 \in F_2(x_1,\xi_2)} \Big\{ C(x_2, \xi_2) + \cdots \\
+ \mathbb{E}_{\boldsymbol{\xi}_{[\sigma,\Sigma]}|\xi_{[1,\sigma-1]}} \Big[\min_{x_\sigma \in F_\sigma(x_{\sigma-1},\xi_\sigma)} \Big\{ C_\sigma(x_\sigma, \xi_\sigma) + \mathbb{E}_{\boldsymbol{\xi}_{[\sigma+1,\Sigma]}|\xi_{[1,\sigma]}} \Big[\cdots \\
+ \mathbb{E}_{\boldsymbol{\xi}_{[\Sigma,\Sigma]}|\xi_{[1,\Sigma-1]}} \Big[\min_{x_\Sigma \in F_\Sigma(x_{\Sigma-1},\xi_\Sigma)} \Big\{ C_\Sigma(x_\Sigma, \xi_\Sigma) \Big] \Big\} \Big] \Big\} \Big] \Big\} \Big] \Big\} \quad (2.7)
\end{aligned}$$

where $\mathbb{E}_{\boldsymbol{\xi}_{[\sigma,\Sigma]}|\xi_{[1,\sigma-1]}}$ denotes the expectation in stage σ with respect to the conditional distribution of $\boldsymbol{\xi}_{[\sigma,\Sigma]}$ given the realization $\xi_{[1,\sigma-1]}$ up to stage $\sigma - 1$. Thus, in (2.7), at each stage σ, we seek to plan production for all periods of $\mathcal{T}^\sigma$, i.e. to decide about the value of x_σ, given the realization of the stochastic input process described by $\xi_{[1,\sigma]}$. This production decisions should lie within the feasible set $F_\sigma(x_{\sigma-1}, \xi_\sigma)$. Moreover, they should minimize the sum of the immediate production costs $C_\sigma(x_\sigma, \xi_\sigma)$ and of the expected future costs computed as the expected value, over all possible realizations of $\boldsymbol{\xi}_{[\sigma+1,\Sigma]}$ given $\xi_{[1,\sigma]}$, of the random production costs up to stage Σ.

2.2.2 Uncertainty representation by a scenario tree

One way to develop computationally tractable solution approaches for Problem (2.7) consists in approximating the stochastic process $\{\xi_\sigma\}_{\sigma=1}^{\Sigma}$ by a process having finitely many realizations in the form of a scenario tree.

With a slight abuse of notation, we will refer to this scenario tree (and all other scenario sub-trees involved in the present work) by mentioning only its set of nodes $\mathcal{V}$. Each node $n \in \mathcal{V}$ corresponds to a single time period t^n and a single decision stage σ^n. Let $\mathcal{V}^t$ be the set of nodes belonging to time period t. Each node n has a unique predecessor node denoted a^n belonging to time period $t^n - 1$. By convention, the root node of the scenario tree is indexed by 1 and a^1 is set to 0. Any non-leaf node n of the tree has a set $\mathcal{C}(n)$ of immediate children. Let $\mathcal{V}(n)$ the sub-tree of $\mathcal{V}$ rooted in n and $\mathcal{L}(n)$ the set of leaf nodes belonging to $\mathcal{V}(n)$. The set of nodes on the path from node n to node m is denoted by $\mathcal{P}(n, m)$.

Each node n represents the state of the system that can be distinguished by the information unfolded up to time period t^n. More precisely, for each leaf node $\ell \in \mathcal{L}(1)$, the set of nodes $\cup_{t \in \mathcal{T}^\sigma} \mathcal{V}^t \cap \mathcal{P}(1, \ell)$ corresponds to a realization $\xi_{[1,\sigma]}$ of the stochastic input process up to stage σ. Thus, the value of the stochastic costs and demand observed at period $t \in \mathcal{T}^\sigma$ in realization $\xi_{[1,\sigma]}$ can be associated to the node $n = \mathcal{V}^t \cap \mathcal{P}(1, \ell)$ and denoted by (f^n, g^n, h^n, d^n). This node n also has a probability denoted by ρ^n which corresponds to the probability of realization $\xi_{[1,\sigma]}$. Moreover, at node $n = \mathcal{V}^{t^L(\sigma)} \cap \mathcal{P}(1, \ell)$ belonging to the last period of stage σ, there are one or several branches describing the possible outcomes of the stochastic process over stages $\sigma + 1$ to Σ, $\boldsymbol{\xi}_{[\sigma+1,\Sigma]}$, given the realization up to stage σ described by $\xi_{[1,\sigma]}$. Finally, a scenario is defined as a path $\mathcal{P}(1, \ell)$ from the root node to a leaf node $\ell \in \mathcal{L}(1)$ in the scenario tree and represents a possible outcome $\xi_{[1,\Sigma]}$ of the stochastic input parameters over the whole planning horizon.

The reader can refer to Figure 2.1 for an illustration of these notations on a small scenario tree.

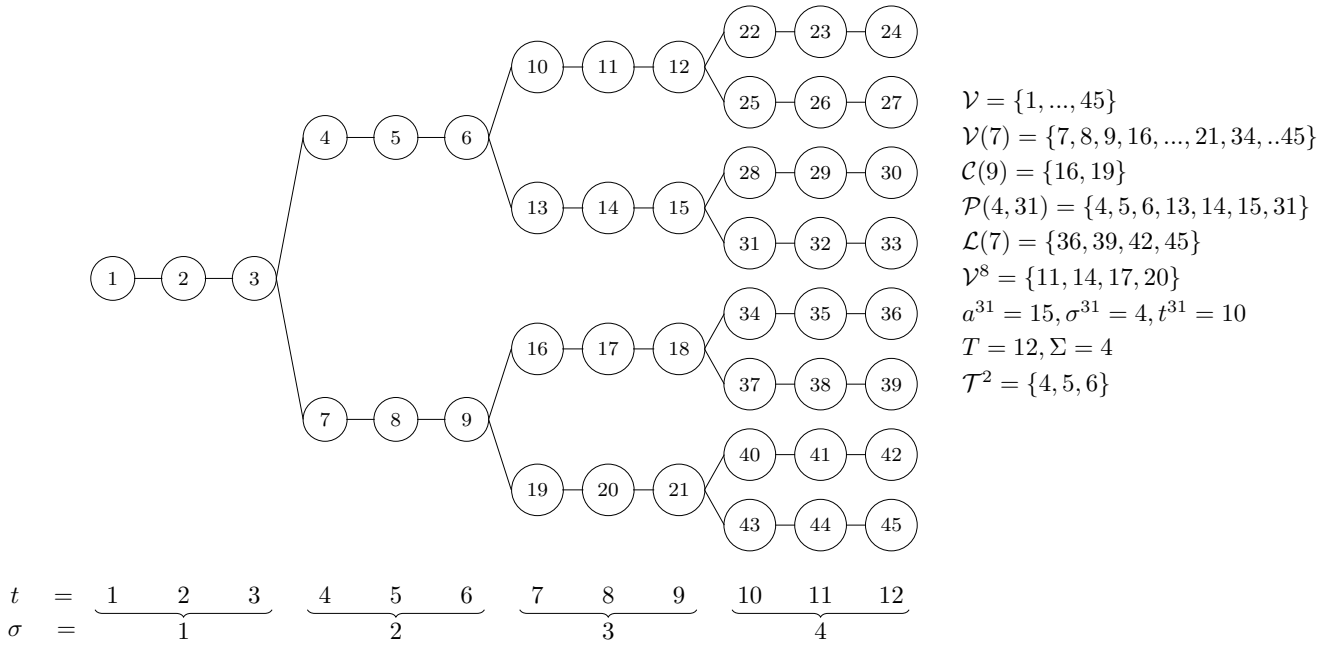


Figure 2.1: Scenario tree structure

Now recall that, thanks to the non-anticipativity property, the production decisions x_σ made at stage σ should depend only on the realization of the stochastic input process up to this stage, i.e. should depend only on $\xi_{[1,\sigma]}$. Let $\cup_{t \in \mathcal{T}^\sigma} \mathcal{V}^t \cap \mathcal{P}(1, \ell)$ be the set of nodes corresponding to realization $\xi_{[1,\sigma]}$ in $\mathcal{V}$. The production decisions made for period t in $x_\sigma(\xi_{[1,\sigma]})$ can be associated to node $n = \mathcal{V}^t \cap \mathcal{P}(1, \ell)$ and will be denoted by (X^n, Y^n, S^n).

In what follows, we briefly present two alternative reformulations of Problem (2.7) based on this representation of the stochastic process evolution by a scenario tree. The first one is an extensive mixed-integer linear programming formulation for which cutting-planes based on strong valid inequalities have been proposed. The second one is a dynamic programming formulation and is

the starting point of a solution approach called the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm based on the decomposition of the problem into a series of smaller sub-problems.

2.3 Cutting-planes generation approach

2.3.1 Extensive MILP formulation

By associating the input parameters (f^n, g^n, h^n, d^n) and decision variables (X^n, Y^n, S^n) to nodes of the scenario tree and by unnesting the different expectation terms involved in (2.7), it is possible to reformulate the SULS as an equivalent deterministic model in the form of an MILP as follows:

$$\min \sum_{n \in \mathcal{V}} \rho^n (f^n Y^n + h^n S^n + g^n X^n) \tag{2.8}$$

$$X^n \leq M^n Y^n \qquad \forall n \in \mathcal{V} \tag{2.9}$$

$$S^n + d^n = X^n + S^{a^n} \qquad \forall n \in \mathcal{V} \tag{2.10}$$

$$X^n, S^n \geq 0, Y^n \in \{0,1\} \qquad \forall n \in \mathcal{V} \tag{2.11}$$

The objective function (2.8) aims at minimizing the expected total setup, inventory holding and production costs over all nodes of the scenario tree. Constraints (2.9) link the production quantity variables to the setup variables. Note that the value of constant M^n can be set by using an upper bound on the quantity to be processed at node n, usually defined as the maximum future demand as seen from node n, i.e. $M^n = \max_{\ell \in \mathcal{L}(n)} d^{n\ell}$, where $d^{n\ell} = \sum_{m \in \mathcal{P}(n,\ell)} d^m$. Constraints (2.10) are the inventory balance constraints. Constraints (2.11) provide the decision variables domain.

Note that the size of the above formulation depends on the number of nodes in the scenario tree. As this one grows exponentially fast with respect to the number of stages, the obtained MILP can be of extremely large size (involving e.g. several millions of variables and constraints). Moreover, due to the presence of big-M type constraints (2.9), the bound provided by its linear relaxation are of rather poor quality. This negatively impacts the performance of a branch-and-cut algorithm so that the direct resolution of Problem (2.8)-(2.11) by a mathematical programming solver may thus pose some numerical difficulty.

2.3.2 Valid inequalities

One possible way of addressing this difficulty is to strengthen the MILP formulation through a set of valid inequalities so as to obtain linear programming bounds of a better quality and improve the performance of a branch-and-cut algorithm at solving the problem. In this section, we discuss two previously published classes of valid inequalities. These valid inequalities take advantage of the scenario tree structure in order to generate valid inequalities for individual scenarios (Path inequalities) or for a set of scenarios (Tree inequalities).

Path inequalities

A path inequality is essentially an inequality valid for the feasible region of a deterministic multi-period problem defined by those constraints that correspond to the nodes on the path $\mathcal{P}(1, n)$ for each $n \in \mathcal{V}$. This deterministic multi-period problem is a relaxation of Problem (2.8)-(2.11). Hence, inequalities valid for this deterministic multi-period problem are also valid for the stochastic problem.

Thus, Formulation (2.8)-(2.11) can be strengthened by applying valid inequalities known for the deterministic ULS to each path of the scenario tree. The $(\ell, \mathscr{S})$ inequalities developped by Barany et al. [16] to strengthen the linear programming relaxation of the ULS can be easily adapted to the SULS (see [46]).

Proposition 1

Given $\ell \in \mathcal{V}$ and $\mathscr{S} \subseteq \mathcal{P}(1,\ell)$, the following inequality is valid for Problem (2.8)-(2.11)*:*

$$S^0 + \sum_{n\in\mathscr{S}} X^n + \sum_{n\in\bar{\mathscr{S}}} d^{n\ell} Y^n \geq d^{1\ell} \tag{2.12}$$

with $d^{n\ell} = \sum_{m\in\mathcal{P}(n,\ell)} d^m$ and $\bar{\mathscr{S}} = \mathcal{P}(1,\ell) \setminus \mathscr{S}$.

For the deterministic ULS, the $(\ell, \mathscr{S})$ inequalities provide a full description of the convex hull of the feasible space. It is however not the case for the SULS. Guan et al. [45] thus proposed to further strengthen formulation (2.8)-(2.11) by exploiting its tree structure. To do this, they consider a set of valid inequalities, each one expressed on a path in the scenario tree, and combine them to obtain a stronger valid inequality expressed on the sub-tree of the scenario tree.

Tree inequalities

Proposition 2

Given a subset $\mathscr{O} = \{n_1, ..., n_O\} \subseteq \mathcal{V}$ which is partially ordered such that $0 = d^{1n_0} \leq d^{1n_1} \leq ... \leq d^{1n_O}$ and a subset $\mathscr{S}_\mathscr{O}$ of nodes belonging to $\cup_{o=1,...,O}\mathcal{P}(1,n_O)$, the following inequality is valid for Problem (2.8)-(2.11)*:*

$$S^0 + \sum_{n\in\mathscr{S}_\mathscr{O}} X^n + \sum_{n\in\bar{\mathscr{S}}_\mathscr{O}} \Delta_n(\mathscr{O}) Y^n \geq d^{1n_O} \tag{2.13}$$

with $\Delta_n(\mathscr{O}) = \sum_{m_o\in\mathscr{O}\cap\mathcal{V}(n)} (d^{1m_o} - d^{1m_{o-1}})$.

Note that inequalities (2.12) can be seen as a special case of inequalities (2.13) in which $\mathscr{O}$ comprises a single node ℓ. Guan et al. [44] showed that a particular case of inequalities (2.13) suffices to fully describe the convex hull of the two-period SULS. However, this is not the case anymore when more than two planning periods are involved in the problem.

In practice, the number of inequalities (2.12) and (2.13) is too large to allow adding all of them a priori to the MILP formulation. Guan et al. [45] proposed to add them to the formulation using a cutting-plane generation algorithm, which resulted in the development of a customized branch-and-cut algorithm. The reader is referred to [45] for more detail about this algorithm.

2.4 Decomposition-based approach

Implicit enumeration methods, such as branch-and-cut algorithms, do not generally scale up well with the size of the scenario tree. Decomposition methods, such as Benders' decomposition, are thus an attractive alternative to tackle instances with large-size scenario trees. In particular, the Stochastic Dual Dynamic Programming (SDDP) approach proposed by Pereira and Pinto [75] has been widely used to solve large-size multi-stage stochastic linear programs. This approach relies on a dynamic programming formulation of the stochastic problem. In this formulation, the overall problem is decomposed into a series of single-node sub-problems in which the future costs of the decision made at node n are represented by an expected cost-to-go function. In a linear setting, the expected cost-to-go functions are piecewise linear convex and can thus be under-approximated through a set of supporting hyperplanes. Recently, Zou et al. [105] proposed a new extension called Stochastic Dual Dynamic integer Programming (SDDiP) of this method in order to solve multi-stage stochastic integer programs with binary state decision variables and non-convex expected cost-to-go functions. One of their main contributions was to introduce a new class of cutting-planes, called Lagrangian

cuts, which satisfies the validity, tightness and finiteness conditions ensuring the convergence of the algorithm.

In what follows, we briefly present how the SDDiP algorithm can be used to solve the SULS. Note that, for the sake of simplicity, we assume throughout this section that there is only one time period t at each stage, i.e., $T = \Sigma$.

2.4.1 Dynamic programming formulation

An alternative to the extensive formulation of the stochastic ULS discussed above is a dynamic programming formulation involving nested expected cost-to-go functions. This approach decomposes the original problem into a series of single-node sub-problems which are linked together by dynamic programming equations.

More precisely, the sub-problem related to node n focuses on defining the production plan for node n based on the entering stock level, S^{a^n}, imposed by its parent node a^n in the scenario tree. Its objective value comprises two terms: a term related to the setup, production and inventory holding costs incurred at node n and a term called the expected cost-to-go function which represents the expected future costs, over all $m \in \mathcal{C}(n)$, incurred by the production decisions made at node n.

The sub-problem for the root node $n = 0$ is expressed as follows:

$$\min(f^0 Y^0 + h^0 S^0 + g^0 X^0) + \sum_{m \in \mathcal{C}(0)} \rho^{0m} \mathscr{Q}^m(S^0) \tag{2.14}$$

subject to:

$$X^0 \leq M^0 Y^0 \tag{2.15}$$
$$S^0 + d^0 = X^0 \tag{2.16}$$
$$X^0, S^0 \geq 0 \tag{2.17}$$
$$Y^0 \in \{0, 1\} \tag{2.18}$$

Note that we assume, without loss of generality, that the entering stock at the root node is zero.

For each node $n \in \mathcal{V} \setminus \{0\}$, the sub-problem is formulated as:

$$\mathscr{Q}^n(S^{a^n}) = \min(f^n Y^n + h^n S^n + g^n X^n) + \sum_{m \in \mathcal{C}(n)} \rho^{nm} \mathscr{Q}^m(S^n) \tag{2.19}$$

subject to:

$$X^n \leq M^n Y^n \tag{2.20}$$
$$S^n + d^n = X^n + S^{a^n} \tag{2.21}$$
$$X^n, S^n \geq 0 \tag{2.22}$$
$$Y^n \in \{0, 1\} \tag{2.23}$$

Here $\mathscr{Q}^n(S^{a^n})$ represents the optimal objective value at node n as a function of the entering stock level S^{a^n}. The expected cost-to-go function at node n is defined as $\mathcal{Q}^n(\cdot) = \sum_{m \in \mathcal{C}(n)} \rho^{nm} \mathscr{Q}^m(\cdot)$. Note that for all leaf nodes, i.e. for all $n \in \mathcal{V}^\Sigma$, $\mathcal{Q}^n(\cdot) \equiv 0$.

2.4.2 Stochastic Dual Dynamic integer Programming for SULS

The main idea of this algorithm is to solve the problem by solving a sequence of single-node sub-problems in which the expected cost-to-go function $\mathcal{Q}^n(\cdot)$ is approximated by a piece-wise linear function. Note that a key assumption for developing this algorithm is that the scenario tree satisfies the *stage-wise independence property*, i.e., for any two nodes n and n' in $\mathcal{V}^\sigma$ the set of children nodes

$\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. In this case, the expected cost-to-go functions depend only on the stage rather than on the nodes, i.e., we have $\mathcal{Q}^n(\cdot) \equiv \mathcal{Q}^\sigma(\cdot)$ for all $n \in \mathcal{V}^\sigma$. As a result, only one expected cost-to-go function has to be approximated per stage and the cuts generated at different nodes n belonging to $\mathcal{V}^\sigma$ are added to a single set of cuts defining the piece-wise linear approximation of function $\mathcal{Q}^\sigma(\cdot)$.

Each iteration of the SDDiP algorithm comprises a sampling step, a forward step and a backward step. In the sampling step, a subset of scenarios is sampled from the scenario tree. In the forward step, the algorithm then proceeds stage-wise from $\sigma = 1$ to Σ by solving, at each node of the sampled scenarios, a dynamic programming equation with an approximate expected cost-to-go function. At the end of this step, the state decision variables are stored and a statistical upper-bound of the problem is computed as the weighted average over all sampled scenarios. In the backward step, we proceed stage-wise from the last stage Σ to the root node and solve at each node a suitable relaxation of the forward problem. The algorithm then adds supporting hyperplanes to the approximate cost-to-go functions of the previous stage. Finally, the nodal problem solved at the root node provides a lower bound of the overall problem. The algorithm stops when the upper and lower bound are close enough, according to a convergence criteria.

In the stochastic ULS, the state variables are the inventory variables, S^n, which are defined as continuous decision variables. Hence, in order to be able to apply the SDDiP algorithm to this problem, we resort to a binary approximation of the state variables. This binarization is obtained by replacing the continuous variable S^n by a set of binary variables $U^{n,\beta}$ such that $S^n = \sum_{\beta \in \mathcal{B}} 2^\beta U^{n,\beta}$, where $\mathcal{B} = \{1, \dots, B\}$. Here $U^{n,\beta} = 1$ if coefficient 2^β is used to compute the value of S^n, 0 otherwise. Moreover, in order to generate the cuts during the backward step of the algorithm, we introduce local copies of the binary state variables. More precisely, $\hat{U}^{n,\beta}$ is an auxiliary decision variable representing the value of the state variable at the parent node of n, i.e. it is a local copy at node n of the state variable $U^{a^n,\beta}$. This leads to the following reformulation of the nodal sub-problem for node $n \in \mathcal{V}$:

$$\mathscr{Q}^n(U^{a^n}) = \min(f^n Y^n + h^n S^n + g^n X^n) + \sum_{m \in \mathcal{C}(n)} \rho^{nm} \mathscr{Q}^m(U^n) \tag{2.24}$$

subject to:

$$X^n \leq M^n Y^n \tag{2.25}$$

$$\sum\nolimits_{\beta \in \mathcal{B}} 2^\beta U^{n,\beta} + d^n = X^n + \sum\nolimits_{\beta \in \mathcal{B}} 2^\beta \hat{U}^{n,\beta} \tag{2.26}$$

$$\hat{U}^{n,\beta} = U^{a^n,\beta} \qquad \forall \beta \in \mathcal{B} \tag{2.27}$$

$$X^n, S^n, \hat{U}^n \geq 0; Y^n \in \{0,1\} \tag{2.28}$$

$$U^{n,\beta} \in \{0,1\} \qquad \forall \beta \in \mathcal{B} \tag{2.29}$$

where U^n denotes the vector of binary variables $U^n = (U^{n,0}, ..., U^{n,\beta}, ..., U^{n,B})$.

We now describe in more detail the 3 steps comprised within an iteration υ of the SDDiP algorithm.

Sampling step

In the sampling step, a number of W scenarios, i.e. a number W of paths from the root node to the leaf nodes, are randomly selected. Let $\Omega_\upsilon = \{\omega_\upsilon^w, ..., \omega_\upsilon^W\}$ be the set of sampled scenarios and ω_υ^w be the set of nodes belonging to scenario w at iteration υ.

Forward step

At iteration υ, the forward step proceeds stage-wise from $\sigma = 1$ to Σ and solves the dynamic programming recursion (2.24)-(2.29) with an approximate expected cost-to-go function for each node

in the sampled set Ω_υ.

Let $\psi_\upsilon^\sigma(U^n)$ be the approximation of the expected cost-to-go function available at iteration υ for stage σ. It is defined by the set of supporting hyperplanes generated until iteration υ. We thus have:

$$\psi_\upsilon^\sigma(U^n) = \min\{\theta^\sigma : \theta^\sigma \geq \sum_{m\in\mathcal{C}(n)} \rho^{nm}(\nu_u^m + (\pi_u^m)^\intercal U^n) \qquad \forall u = \{1, ..., \upsilon - 1\}\} \tag{2.30}$$

where ν_u^m and π_u^m are the coefficients of the cuts generated at iteration $u < \upsilon$.

At each sampled node n, we thus solve the following nodal sub-problem denoted by $P_\upsilon^n(U_\upsilon^{a^n}, \psi_\upsilon^\sigma)$. Here, $U_\upsilon^{a^n}$ denotes the binary approximation vector providing the level of ending stock in the solution of the nodal sub-problem at the parent node a^n during iteration υ of the algorithm.

$$\underline{\mathscr{Q}}_\upsilon^n(U_\upsilon^{a^n}) = \min(f^n Y^n + h^n S^n + g^n X^n) + \psi_\upsilon^\sigma(U^n) \tag{2.31}$$

subject to (2.25)-(2.29).

Note that this sub-problem is a small-size mixed-integer linear program than can be solved by a standard mathematical programming solver.

The forward step ends when the nodal sub-problems for all nodes in Ω_υ have been solved. Its output is thus a state variable solution U_υ^n for each node in Ω_υ.

Backward step

The aim of the backward step is to update the approximate expected cost-to-go function $\psi_\upsilon^\sigma(\cdot)$ for each stage σ. This step starts from the last stage Σ and goes back to stage 2. Note that the last stage nodal sub-problems do not have an expected cost-to-go function, therefore $\psi_\upsilon^\Sigma \equiv 0$ for all iterations υ. At each stage $\sigma = \Sigma, ..., 2$, the algorithm solves a suitable relaxation for each node $n \in \mathcal{V}^\sigma$. This enables the algorithm to collect the cut coefficients $\{\nu_\upsilon^n, \pi_\upsilon^n\}$ and generate a new linear inequality to strengthen the approximation of $\psi_{\upsilon+1}^{\sigma-1}(\cdot)$. The backward step continues iteratively until the approximation of the expected cost-to-go function at stage $\sigma = 1$ is updated.

Since the linear cuts (2.30) are under-approximations of the true expected cost-to-go function $\mathcal{Q}^\sigma(\cdot)$, the optimal value of the forward problem at the root node provides a lower bound of the optimal value (2.24).

Stopping criteria

We consider three stopping criteria that are used in the literature. At each iteration υ, a statistical upper bound (UB) is computed by the forward step and a lower bound (LB) of the optimal value is generated at the root node at the end of the backward step. A first stopping criterion imposes the termination of the algorithm when the gap $\frac{|UB_\upsilon - LB_\upsilon|}{LB_\upsilon}$ is lower than a convergence threshold ϵ. A second stopping criterion stops the algorithm when the lower bound becomes stable during a fixed number of iterations. Finally, a limit on the number of iterations is also enforced. In what follows, we detail three different families used to strengthen the approximation of the expected cost-to-go function.

Integer Optimality Cuts

Let U_υ^n be a solution of problem $P_\upsilon^n(U_\upsilon^{a^n}, \psi_\upsilon^\sigma)$ solved during iteration υ at sampled node n in the forward step. To generate an integer optimality cut at node n, we solve, for each $m \in \mathcal{C}(n)$, problem $P_\upsilon^m(U_\upsilon^n, \psi_{\upsilon+1}^\sigma)$, i.e. the original nodal subproblem with an updated approximation $\psi_{\upsilon+1}^\sigma$. Let $\nu_{\upsilon+1}^m$ be its optimal objective value and $\bar{\nu}_{\upsilon+1}^n = \sum_{m\in\mathcal{C}(n)} \rho^{nm}\nu_{\upsilon+1}^m$. The integer optimality cut generated at iteration υ in the backward step takes the following form:

$$\theta^\sigma \geq (\bar{\nu}_{\upsilon+1}^n)\Big(\sum_{\beta=0}^{B}(U_\upsilon^{n,\beta} - 1)U^{n,\beta} + \sum_{\beta=0}^{B}(U^{n,\beta} - 1)U_\upsilon^{n,\beta}\Big) + \bar{\nu}_{\upsilon+1}^n \tag{2.32}$$

where $U^{n,\beta}$ corresponds to the β-th entry of the binary approximation of S^n.

Lagrangian Cuts

Zou et al. [105] introduced a Lagrangian cut family. They proved that these cuts display the validity, tightness and finiteness conditions ensuring the theoretical convergence of the algorithm to the optimal solution. A Lagrangian cut is generated at node n in the backward step at iteration υ by considering the Lagrangian relaxation of each problem $P_\upsilon^m(U_\upsilon^n, \psi_\upsilon^\sigma)$, $m \in \mathcal{C}(n)$, in which the copy constraints (2.27) have been dualized. Each corresponding Lagrangian dual problem is then solved to optimality. The generated Lagrangian cut takes the form (2.30), where ν_υ^m corresponds to the optimal value of the Lagrangian dual problem and π_υ^m to the Lagrangian dual optimal solution.

Strengthened Benders' cuts

This family of cuts have been deduced by the observation that for any fixed dual solution $\pi^n = \{\pi^{n,\beta} : \beta \in \mathcal{B}\}$, solving the Lagrangian relaxation created by relaxing the set of constraints $\hat{U}^{n,\beta} = U_\upsilon^{a^n,\beta}$ yields a valid cut. A strengthened Benders' cut is generated at node n in the backward step at iteration υ by solving the linear relaxation of problem $P_\upsilon^m(U_\upsilon^n, \psi_\upsilon^\sigma)$, $m \in \mathcal{C}(n)$. The value of each coefficients $\pi_\upsilon^{m,\beta}$ is set to the dual value of the copy constraint $\hat{U}^{m,\beta} = U^{n,\beta}$ in this linear relaxation. The value of ν_υ^m is obtained by solving the Lagrangian relaxation of problem $P_\upsilon^m(U_\upsilon^n, \psi_\upsilon^\sigma)$ in which all copy constraints (2.30) have been dualized and the Lagrangian multiplier of constraint $\hat{U}^{m,\beta} = U^{n,\beta}$ set to $\pi_\upsilon^{m,\beta}$.

As a synthesis, the main steps of the SDDiP algorithm applied to the stochastic ULS are summarized in Algorithm 1.

2.5 Conclusion

We provided in this chapter some background on the multi-stage stochastic uncapacitated lot-sizing problem in order to ease the reading of the manuscript. We also presented an introduction to two solution approaches that may be used for this problem. The first one relies on a mixed-integer linear programming formulation and corresponds to a customized branch-and-cut algorithm based on path and tree valid inequalities. The second one, called the SDDiP algorithm, uses a dynamic programming formulation as a starting point and basically consists in fully decomposing the problem into a set of small single-node sub-problems. In Chapter 5, we propose a solution approach which combines the SDDiP algorithm with the polyhedral approach, in order to further improve the quality of the solutions obtained for large-size instances. Note that the proposed algorithm, called the extSDDiP algorithm, relies on a partial decomposition of the problem.

Moreover, as mentioned in the introduction, we also study in this thesis a more realistic multi-stage stochastic lot-sizing problem aiming at planning production for a multi-echelon multi-item remanufacturing system. For this problem, we first discuss in Chapter 3 the development of customized branch-and-cut algorithms similar to the one discussed in Section 2.3.2. These algorithms are based on path and tree inequalities obtained from inequalities previously known for the deterministic uncapacitated single-item single-echelon lot-sizing problem with lost sales. In order to further strengthen the MILP formulation of the problem, we then investigate in Chapter 4 new valid inequalities which seek to take into account the multi-echelon aspect of the production system and the fact that the production on a resource is limited by the availability of the returned products. However, in view of the theoretical and numerical difficulties lying ahead, we focus on the deterministic variant of the remanufacturing planning problem investigated in Chapter 3. Namely, before seeking to extend valid inequalities to a multi-stage stochastic problem expressed on a scenario tree, it is necessary to identify valid inequalities which are efficient at strengthening its deterministic counterpart and

Algorithm 1: SDDiP algorithm

Initialize $LB \leftarrow -\infty, UB \leftarrow +\infty, \upsilon \leftarrow 1$
while *no stopping criterion is satisfied* **do**
  Randomly select W scenarios $\Omega_\upsilon = \{\omega_\upsilon^1, ..., \omega_\upsilon^W\}$.
  for $w = 1, ..., W$ **do**
    for $n \in \omega_\upsilon^w$ **do**
      Solve $P_\upsilon^n(U^{a^n}, \psi_\upsilon^{\sigma^n})$
      Collect $(X_\upsilon^n, Y_\upsilon^n, S_\upsilon^n, U_\upsilon^n)$
    end
    $\upsilon^w \leftarrow \sum_{n \in \omega_\upsilon^w}(f^n Y_\upsilon^n + h^n S_\upsilon^n + g^n X_\upsilon^n)$
  end
  $\hat{\mu} \leftarrow \sum_{w=1}^W \upsilon^w$ and $\hat{\sigma}^2 \leftarrow \frac{1}{W-1}\sum_{w=1}^W(\upsilon^w - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2}\frac{\hat{\sigma}}{\sqrt{W}}$
  for $\sigma = \Sigma - 1, ..., 1$ **do**
    for $w = 1, ..., W$ **do**
      for $n \in \mathcal{V}^\sigma \cap \omega_\upsilon^w$ **do**
        Generate an integer optimality cut, a Lagrangian cut and a strengthened Benders' cut considering all nodes $m \in \mathcal{C}(n)$
      end
    end
    Add the generated cuts to ψ_υ^σ to get $\psi_{\upsilon+1}^\sigma$.
  end
  $LB \leftarrow P_\upsilon^0(0, \psi_{\upsilon+1}^1)$
  $\upsilon \leftarrow \upsilon + 1$
end

to study how to solve the corresponding separation problem. Finally, we present in Chapter 6 an adaptation of the extSDDiP algorithm introduced in Chapter 5 for the SULS to solve the stochastic remanufacturing planning problem.

Part II

Cutting planes generation approaches

Chapter 3

Stochastic multi-echelon lot-sizing problem with remanufacturing and lost sales

This chapter first introduces the stochastic remanufacturing planning problem which is at the core of this thesis. We describe the studied remanufacturing system which comprises three production echelons: disassembly, refurbishing and reassembly and model the problem of planning production on this system as an uncapacitated multi-item multi-echelon lot-sizing problem with returns and lost sales. We then seek to explicitly take into account uncertainties on the input data of this optimization problem. To this aim, we propose a multi-stage stochastic integer programming approach relying on scenario trees to represent the uncertain information structure. Finally, we develop customized branch-and-cut algorithms in order to solve the resulting mixed-integer linear program to optimality. These branch-and-cut algorithms are based 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 run at each node of the branch-and-cut search tree. Our computational experiments show that the proposed method is capable of significantly decreasing the computation time needed to obtain guaranteed optimal solutions. Additionally, 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.

3.1 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 [67]. 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 set-up operations such as tool changes, machine calibration or machine installation incurring fixed set-up costs. As a naive perception, to reduce these set-up 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 set-up 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 set-up 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 [49] and [48]. Lage and Filho [64] provided a recent literature review about production planning and control for remanufacturing systems. They analyzed whether the gap identified by Guide [48] 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 [56] 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 [56] 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. [33] 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. [60] 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. [65]. 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 set-up 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.

3.1.1 Related work

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 [61], [62] and [74], taking under consideration both stochastic demand and returns quantity. Kilic [61] and Kilic et al. [62] included customer service level constraints and developed a heuristic approach based on a static-dynamic uncertainty strategy. Naeem et al. [74] 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. [68] and Hilger et al. [52] studied a multi-item variant of the problem taking into account both stochastic demand and returns quantity. Macedo et al. [68] extended the previous works ([61], [62] and [74]) by considering also stochastic set-up costs and proposed a two-stage stochastic programming model that assumes production and set-up as first-stage decision variables and inventory, disposal, and backlogging as second-stage decision variables. Hilger et al. [52] 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 [101] and Fang et al. [38] 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 [101] developed a two-stage stochastic programming model aiming at finding a compromise between the expected cost and the solution robustness. Fang et al. [38] 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. [38]. 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 [46], [45], [34] and [102]. Guan et al. [46] investigated an uncapacitated lot-sizing problem and extended the (ℓ, S) valid inequalities known for the deterministic variant to a general facet-defining class called (Q, S_Q) for the stochastic variant. Later, Di Summa and Wolsey [34] studied a capacitated lot sizing problem and extended the work of Guan et al. [46] by proving that the (Q, S_Q) valid inequality is dominated by a mixing inequality. Additionally, Guan et al. [45] 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. [102] 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.

3.1.2 Contributions

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 [66]. 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 3.2 formally describes the problem and proposes a mixed integer linear programming model. In Section 3.3, 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 3.4 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 3.5. Section 3.6 reports the results of computational experiments and discusses the performance of our branch-and-cut algorithm. Finally, Section 3.7 gives the conclusions with possible issues for further research.

3.2 Problem description and mathematical formulation

3.2.1 System description

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

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

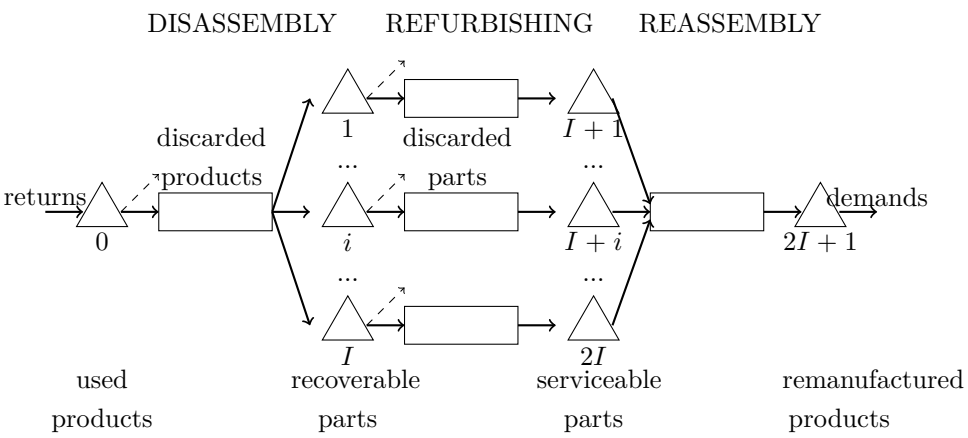


Figure 3.1: Illustration of studied remanufacturing system

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 set-up 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. [5] 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.

3.2.2 Uncertainty

As mentioned in above, 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. With a slight abuse of notation, we will refer to this scenario tree by mentioning only its set of nodes $\mathcal{V}$. Each node $n \in \mathcal{V}$ corresponds to a single time period t^n and a single-stage σ^n and it represents the state of the system that can be distinguished by the information unfolded up to time period t^n. Let $\mathcal{C}(n)$ be the set of immediate children of node n, $\mathcal{V}(n)$ the sub-tree of $\mathcal{V}$ rooted in n and $\mathcal{L}(n)$ the set of leaf nodes belonging to $\mathcal{V}(n)$. We refer the reader to Section 2.2.2 for a detail description of a scenario tree.

We use the following notations for the problem formulation:

- I: number of part/items 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:

- ϖ_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 $n \in \mathcal{V}$,
- δ_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: set-up 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 $g^n = \sum_{i=1}^{I} q_i^n(1-\delta_i^n)\varpi_i$ 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, δ_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 $n \in \mathcal{V}$.

3.2.3 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 $n \in \mathcal{V}$,
- $Y_p^n \in \{0, 1\}$: set-up 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.

$$\min \sum_{n\in\mathcal{V}} \rho^n \Big(\sum_{p\in\mathcal{J}} f_p^n Y_p^n + \sum_{i\in\mathcal{I}} h_i^n S_i^n + l^n L^n + \sum_{i\in\mathcal{I}_r\cup\{0\}} q_i^n Q_i^n + g^n X_0^n \Big) \tag{3.1}$$

subject to

$$X_p^n \leq M_p^n Y_p^n \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \qquad (3.2)$$
$$S_0^n = S_0^{a^n} + r^n - X_0^n - Q_0^n \qquad \forall n \in \mathcal{V} \qquad (3.3)$$
$$S_i^n = S_i^{a^n} + \delta_i^n \varpi_i X_0^n - X_i^n - Q_i^n \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} \qquad (3.4)$$
$$S_i^n = S_i^{a^n} + X_{i-I}^n - \varpi_i X_{I+1}^n \qquad \forall i \in I_s, \forall n \in \mathcal{V} \qquad (3.5)$$
$$S_{2I+1}^n = S_{2I+1}^{a^n} + X_{I+1}^n - d^n + L^n \qquad \forall n \in \mathcal{V} \qquad (3.6)$$
$$S_i^0 = 0 \qquad \forall i \in \mathcal{I} \qquad (3.7)$$
$$S_i^n \geq 0 \qquad \forall i \in \mathcal{I}, \forall n \in \mathcal{V} \qquad (3.8)$$
$$Q_i^n \geq 0 \qquad \forall i \in \mathcal{I}_r \cup \{0\}, \forall n \in \mathcal{V} \qquad (3.9)$$
$$L^n \geq 0 \qquad \forall n \in \mathcal{V} \qquad (3.10)$$
$$X_p^n \geq 0, Y_p^n \in \{0,1\} \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \qquad (3.11)$$

The objective function (3.1) aims at minimizing the expected total cost, over all nodes of the scenario tree. This cost is the sum of the expected set-up, inventory holding, lost sales and disposal costs. Constraints (3.2) link the production quantity variables to the set-up variables. Constraints (3.3)-(3.6) are the inventory balance constraints. Constraints (3.3) (resp. (3.4) and (3.5)) involve a term corresponding to a dependent demand X_0^n (resp. X_i^n and $\varpi_i X_{I+1}^n$) whereas Constraints (3.6) 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 (3.8)-(3.11) 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^ν on the nodes ν 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 $\mathcal{P}(n,\ell)$. It is the maximum, over all leaf nodes ℓ in $\mathcal{L}(n)$, of the cumulated demand on the path from the node n to the leaf node ℓ.

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

- $M_0^n = \min\left\{ \sum_{\nu \in \mathcal{P}(0,n)} r^\nu, \max_{\ell \in \mathcal{L}(n)} \left\{ \frac{\sum_{\nu \in \mathcal{P}(n,\ell)} d^\nu}{min_{i=1\ldots I} \delta_i^n} \right\} \right\}$
- $M_p^n = \min\left\{ \sum_{\nu \in \mathcal{P}(0,n)} (\varpi_p r^\nu \max_{\mu \in \mathcal{P}(\nu,n)} \delta_p^\mu), \max_{\ell \in \mathcal{L}(n)} \left\{ \sum_{\nu \in \mathcal{P}(n,\ell)} \varpi_p d^\nu \right\} \right\}$, for $p = 1, ..., I$.
- $M_{I+1}^n = \min\left\{ \min_{p \in \{1,\ldots,I\}} \left\{ \sum_{\nu \in \mathcal{P}(0,n)} (r^\nu \max_{\mu \in \mathcal{P}(\nu,n)} \delta_p^\mu) \right\}, \max_{\ell \in \mathcal{L}(n)} \left\{ \sum_{\nu \in \mathcal{P}(n,\ell)} d^\nu \right\} \right\}$

Even if the problem (3.1)-(3.11) 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 $\mathcal{O}(|\mathcal{V}||\mathcal{J}|)$ binary variables, $\mathcal{O}(|\mathcal{V}||\mathcal{J}|)$ continuous variables and $\mathcal{O}(|\mathcal{V}||\mathcal{J}+\mathcal{I}|)$ constraints. The size of the scenario tree $|\mathcal{V}|$ is in $\mathcal{O}(c^{T+1})$ 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.

3.3 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 [79] 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.

3.3.1 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 $ed_i^n = \varpi_i d^n$. 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 $ed_0^n = \frac{d^n}{min_{i \in \mathcal{I}_r} \delta_i^n}$ 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 $\frac{d^n}{min_{i \in \mathcal{I}_r} \delta_i^n}$. 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:

- $E_i^n = S_i^n + E_{I+i}^n = S_i^n + S_{I+i}^n + \varpi_i S_{2I+1}^n$, for $i \in \mathcal{I}_r$, for $n \in \mathcal{V}$
- $E_i^n = S_i^n + \varpi_i E_{2I+1}^n = S_i^n + \varpi_i S_{2I+1}^n$, 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 $n \in \mathcal{V}$
- $eh_{2I+1}^n = h_{2I+1}^n - \sum_{i \in \mathcal{I}_r} \varpi_i h_i^n$, for $n \in \mathcal{V}$

This leads to the following mixed-integer linear programming formulation:

$$Z^* = \min \sum_{n \in \mathcal{V}} \rho^n \Big(\sum_{p \in \mathcal{J}} f_p^n Y_p^n + h_0^n S_0^n + \sum_{i \in \mathcal{I} \setminus \{0\}} eh_i^n E_i^n + l^n L^n + \sum_{i \in \mathcal{I}_r \cup \{0\}} q_i^n Q_i^n + g^n X_0^n \Big) \tag{3.12}$$

subject to:

$$X_p^n \leq M_p^n Y_p^n \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \tag{3.13}$$
$$S_0^n = S_0^{a^n} + r^n - X_0^n - Q_0^n \qquad \forall n \in \mathcal{V} \tag{3.14}$$
$$E_i^n = E_i^{a^n} + \delta_i^n \varpi_i X_0^n - \varpi_i d^n + \varpi_i L^n - Q_i^n \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} \tag{3.15}$$
$$E_i^n = E_i^{a^n} + X_{i-I}^n - \varpi_i d^n + \varpi_i L^n \qquad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{V} \tag{3.16}$$
$$E_{2I+1}^n = E_{2I+1}^{a^n} + X_{I+1}^n - d^n + L^n \qquad \forall n \in \mathcal{V} \tag{3.17}$$
$$E_i^n - E_{I+i}^n \geq 0 \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} \tag{3.18}$$
$$E_i^n - \varpi_i E_{2I+1}^n \geq 0 \qquad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{V} \tag{3.19}$$
$$E_{2I+1}^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{3.20}$$
$$S_0^0 = 0 \tag{3.21}$$
$$E_i^0 = 0 \qquad \forall i \in \mathcal{I} \setminus \{0\} \tag{3.22}$$
$$S_0^n \geq 0, L^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{3.23}$$
$$Q_i^n \geq 0 \qquad \forall i \in \mathcal{I}_r \cup \{0\}, \forall n \in \mathcal{V} \tag{3.24}$$
$$X_p^n \geq 0, Y_p^n \in \{0,1\} \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{V} \tag{3.25}$$

As in the previous formulation, the objective function (3.12) aims at minimizing the expected cost, over all nodes of the scenario tree. Constraints (3.13) are defined as Constraints (3.2) of the (3.1)-(3.11) formulation. Constraints (3.14)-(3.17) are inventory balance constraints. Constraints (3.14) use the classical inventory variables, whereas Constraints (3.15)-(3.17) make use of the echelon inventory variables. Contrary to Constraints (3.4)-(3.5) of the natural formulation, Constraints (3.15)-(3.17) do not involve a dependent demand term but only an external demand term. Constraints (3.18)-(3.20) 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 (3.23)-(3.25) define the domain of the decision variables.

3.3.2 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 (3.1)-(3.11) formulation. This induces that the constraint matrix of (3.12)-(3.25) displays a specific structure: it can be decomposed in a series of single-echelon single-resource lot-sizing subproblems coupled by the linking constraints (3.14), (3.18)-(3.20). The single-echelon subproblems are defined as follows.

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

$$Z_p^* = \min \sum_{n \in \mathcal{V}} \rho^n \Big(f_p^n Y_p^n + eh_{p+I}^n E_{p+I}^n + l^n L^n \Big) \tag{3.26}$$

subject to:

$$X_p^n \leq M_p^n Y_p^n \qquad \forall n \in \mathcal{V} \tag{3.27}$$
$$E_{p+I}^n = E_{p+I}^{a^n} + X_p^n - \varpi_p d^n + \varpi_p L^n \qquad \forall n \in \mathcal{V} \tag{3.28}$$
$$E_{p+I}^0 = X_p^0 - \varpi_p d^0 + \varpi_p L^0 \tag{3.29}$$
$$E_{p+I}^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{3.30}$$
$$L^n \geq 0, X_p^n \geq 0, Y_p^n \in \{0,1\} \qquad \forall n \in \mathcal{V} \tag{3.31}$$

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

$$Z_0^* = \min \sum_{n \in \mathcal{V}} \rho^n \Big(f_0^n Y_0^n + \sum_{i=1}^{I} eh_i^n E_i^n + \sum_{i=1}^{I} q_i^n Q_i^n + l^n L^n + g^n X_0^n \Big) \tag{3.32}$$

subject to:

$$X_0^n \leq M_0^n Y_0^n \qquad \forall n \in \mathcal{V} \tag{3.33}$$
$$E_i^n = E_i^{a^n} + \delta_i^n \varpi_i X_0^n - \varpi_i d^n + \varpi_i L^n - Q_i^n \qquad \forall n \in \mathcal{V} \tag{3.34}$$
$$E_i^0 = \delta_i^0 \varpi_i X_0^0 - \varpi_i d^0 + \varpi_i L^0 - Q_i^0 \tag{3.35}$$
$$E_i^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{3.36}$$
$$L^n \geq 0, X_0^n \geq 0, Q_i^n \geq 0, Y_0^n \in \{0,1\} \qquad \forall n \in \mathcal{V} \tag{3.37}$$

Each subproblem (3.26)-(3.31) or (3.32)-(3.37) 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. [66] who proposed a family of valid inequalities called (k, U) inequalities, to strengthen the linear relaxation. We discuss in Section 3.4 how these inequalities known for the deterministic variant of the problem can be used to solve the stochastic problem expressed on a scenario tree.

3.4 Valid inequalities

In this section, we provide $(k,\mathcal{U})$ inequalities for each single-echelon subproblem described in Section 3.3.2. We first exploit these $(k,\mathcal{U})$ inequalities considering their application to each individual scenario, i.e. to each individual path from a non-terminal node n to a leaf node $\ell \in \mathcal{L}(n)$ in the scenario tree $\mathcal{V}$. Next, we extend them to a more general class of inequalities. This is done by exploiting the scheme proposed by Guan et al. [45] 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 $(k,\mathcal{U})$ inequality applied to an individual scenario as a *path inequality* and to a $(k,\mathcal{U})$ inequality applied to a subtree as a *tree inequality*.

3.4.1 Path inequalities

We first recall the relevant notation introduced in Section 3.2.2. Each node k of the scenario tree $\mathcal{V}$, except for the root node, has a unique parent, and each non-terminal node k is the root of a subtree $\mathcal{V}(k)$, with $\mathcal{V}(1) = \mathcal{V}$. 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^ℓ be the immediate successor of node k belonging to the set $\mathcal{P}(k,\ell)$, for $\ell \in \mathcal{L}(k)$. Let $\mathcal{U}_{k,\ell} \subseteq \mathcal{P}(c_k^\ell, \ell)$ be a subset of nodes belonging to the path from the node c_k^ℓ to the leaf node ℓ, not necessarily consecutive.

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

Proposition 3

Let $k \in \mathcal{V}$ and $\ell \in \mathcal{L}(k)$. Let $\mathcal{U}_{k,\ell} \subset \mathcal{P}(c_k^\ell, \ell)$. The following $(k,\mathcal{U})$ inequality

$$E_{p+I}^k \geq \varpi_p \sum_{\mu \in \mathcal{U}_{k,\ell}} \left[d^\mu \Big(1 - \sum_{m \in \mathcal{P}(c_k^\ell, \mu)} Y_p^m \Big) - L^\mu \right] \tag{3.38}$$

is valid for the problem (3.12)-(3.25).

The proof is direct following the proof in [66].

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 ℓ. For a node $\mu \in \mathcal{U}_{k,\ell}$, if $\sum_{m \in \mathcal{P}(c_k^\ell, \mu)} Y_p^m \geq 1$, the demand of node μ, $\varpi_p d^\mu$, can be satisfied by a production in one of the nodes $m \in \mathcal{P}(c_k^\ell, \mu)$ and does not have to be in stock at the node k. But if $\sum_{m \in \mathcal{P}(c_k^\ell, \mu)} Y_p^m = 0$, the demand $\varpi_p d^\mu$ cannot be produced in any node $m \in \mathcal{P}(c_k^\ell, \mu)$ meaning that the portion of this demand which will be satisfied by a production $\varpi_p(d^\mu - L^\mu)$, 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:

$$E_i^k \geq \varpi_i \sum_{\mu \in \mathcal{U}_{k,\ell}} \left[d^\mu \Big(1 - \sum_{m \in \mathcal{P}(c_k^\ell, \mu)} Y_0^m \Big) - L^\mu \right]$$

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 / ϖ_i is minimum. This leads to the following proposition.

Proposition 4

Let $k \in \mathcal{V}$ and $\ell \in \mathcal{L}(k)$. Let $\mathcal{U}_{k,\ell} \subset \mathcal{P}(c_k^\ell, \ell)$. The following $(k,\mathcal{U})$ inequality

$$\min_{i \in \mathcal{I}_r} \left[\frac{E_i^k}{\varpi_i} \right] \geq \sum_{\mu \in \mathcal{U}_{k,\ell}} \left[d^\mu \Big(1 - \sum_{m \in \mathcal{P}(c_k^\ell, \mu)} Y_0^m \Big) - L^\mu \right] \tag{3.39}$$

is valid for the problem (3.12)-(3.25).

3.4.2 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 [46] and [45]. 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 $\mathcal{V}(k)$ and $\mathcal{U} = \cup_{\ell \in \mathcal{L}(k)} \mathcal{U}_{k,\ell}$ 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 5

Let $k \in \mathcal{V}$ and $\mathcal{U} \subset \mathcal{V}(k)$. Let $\mathcal{s} = \{\mathcal{s}_1, ..., \mathcal{s}_{|\mathcal{L}(k)|}\}$ be a sequence of leaf nodes belonging to $\mathcal{L}(k)$ in increasing order of cumulative demand $\sum_{\mu \in \mathcal{U}_{k,\ell}} d^\mu$: $\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_1}} d^\mu \leq ... \leq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\iota}} d^\mu \leq ... \leq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_{|\mathcal{L}(k)|}}} d^\mu$. We set $\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_0}} d^\mu = 0$. The following inequality

$$E^k_{p+I} + \varpi_p \sum_{\mu \in U} L^\mu + \varpi_p \sum_{m \in \mathcal{V}(k) \setminus \{k\}} \phi^m Y^m_p \geq \varpi_p \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_{|\mathcal{L}(k)|}}} d^\mu \tag{3.40}$$

is valid for problem (3.12)-(3.25), with

$$\phi^m = \min \Big\{ \max_{\ell \in \mathcal{L}(m)} \{ \sum_{\mu \in \mathcal{U}_{a^m,\ell}} d^\mu \}, \sum_{\iota = 1...|\mathcal{L}(k)| \text{ s.t } \mathcal{s}_\iota \in \mathcal{L}(m)} \Big(\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\iota}} d^\mu - \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\iota - 1}} d^\mu \Big) \Big\}$$

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

$$E^k \geq \sum_{\mu \in \mathcal{U}_{k,\ell}} \Big[d^\mu (1 - \sum_{m \in \mathcal{P}(c^\ell_k, \mu)} Y^m) - L^\mu \Big] \tag{3.41}$$

We rewrite the inequalities (3.41) in a form making it easier to apply Theorem 2 in [45].

$$E^k + \sum_{\mu \in \mathcal{U}_{k,\ell}} L^\mu + \sum_{m \in \mathcal{P}(c^\ell_k, \ell)} (\sum_{\mu \in \mathcal{U}_{a^m,\ell}} d^\mu) Y^m \geq \sum_{\mu \in \mathcal{U}_{k,\ell}} d^\mu$$

Let $\mathcal{s} = \{\mathcal{s}_1, ..., \mathcal{s}_{|\mathcal{L}(k)|}\}$ be a sequence of leaf nodes in increasing order of cumulative demand $\sum_{\mu \in \mathcal{U}_{k,\ell}} d^\mu$: $\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_1}} d^\mu \leq ... \leq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\iota}} d^\mu \leq ... \leq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_{|\mathcal{L}(k)|}}} d^\mu$. We set $\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_0}} d^\mu = 0$. Now, we use Theorem 2 in [45] to combine these $|\mathcal{L}(k)|$ path inequalities and derive a new tree inequality as follows:

$$E^k + \sum_{\mu \in \cup_{\ell \in \mathcal{L}(k)} \mathcal{U}_{k,\ell}} L^\mu + \sum_{m \in \mathcal{V} \setminus \{k\}} \phi^m Y^m \geq (\sum_{\mu \in \mathcal{U}_{k,|\mathcal{L}(k)|}} d^\mu)$$

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

$$\phi^m = \min \Big\{ \max_{\ell \in \mathcal{L}(m)} \{ \sum_{\mu \in \mathcal{U}_{a^m,\ell}} d^\mu \}, \sum_{\ell \in \mathcal{L}(m)} \Big(\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\ell}} d^\mu - \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\ell - 1}} d^\mu \Big) \Big\}$$

with $\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_0}} d^\mu$ set to 0. □

The same scheme can be applied starting with the path inequalities (3.39) for the disassembly process $p = 0$. This leads to the following proposition:

Proposition 6

Let $k \in \mathcal{V}$ and $\mathcal{U} \subset \mathcal{V}(k)$. Let $\mathcal{s} = \{\mathcal{s}_1, ..., \mathcal{s}_{|\mathcal{L}(k)|}\}$ be a sequence of leaf nodes belonging to $\mathcal{L}(k)$ in the increasing order of the cumulative demand $\sum_{\mu \in \mathcal{U}_{k,\ell}} d^\mu$: $\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_0}} d^\mu \leq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_1}} d^\mu \leq ... \leq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\iota}} d^\mu \leq ... \leq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_{|\mathcal{L}(k)|}}} d^\mu$. We set $\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_0}} d^\mu = 0$. The following inequality

$$\min_{i \in \mathcal{I}_r} \left\lceil \frac{E_i^k}{\varpi_i} \right\rceil + \sum_{\mu \in \mathcal{U}} L^\mu + \sum_{m \in \mathcal{V}(k) \setminus \{k\}} \phi^m Y_0^m \geq \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_{|\mathcal{L}(k)|}}} d^\mu \tag{3.42}$$

is valid for problem (3.12)-(3.25), with

$$\phi^m = \min \Big\{ \max_{\ell \in \mathcal{L}(m)} \{ \sum_{\mu \in \mathcal{U}_{a^m,\ell}} d^\mu \}, \sum_{\iota = 1...|\mathcal{L}(k)| \ s.t \ \mathcal{s}_\iota \in \mathcal{L}(m)} \Big(\sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\iota}} d^\mu - \sum_{\mu \in \mathcal{U}_{k,\mathcal{s}_\iota - 1}} d^\mu \Big) \Big\}$$

3.5 Cutting-plane generation

The number of valid inequalities (3.38), (3.39), (3.40) and (3.42) 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 (3.12)-(3.25).

3.5.1 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 (3.38)-(3.39) if it exists or proving that no such inequality exists. For a given process p, node $k \in \mathcal{V}$ and leaf node $\ell \in \mathcal{L}(k)$, finding the most violated inequality corresponds to identifying the set $\mathcal{U}_{k,\ell}$ maximizing the right-hand side of the inequality. We note that the value of the term corresponding to a node $\mu \in \mathcal{U}_{k,\ell}$ in the right-hand side of (3.38)-(3.39) does not depend on the other nodes belonging to $\mathcal{U}_{k,\ell}$. Hence, each node of $\mathcal{P}(c_k^\ell, \ell)$ 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 $\mathcal{U}_{k,\ell}$, if not, it is discarded. This leads to the following exact separation algorithm for inequalities (3.38)-(3.39):

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

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

We underline that the above strategy can be implemented in polynomial time [66], namely, the running-time of the proposed separation algorithm is $\mathcal{O}(|\mathcal{P}|^2)$, where $|\mathcal{P}|$ corresponds to the number of nodes in the set $\mathcal{P}(c_k^\ell, \ell)$.

3.5.2 Cutting-plane generation for path inequalities

Our preliminary computational experiments showed that adding all the violated inequalities of class (3.38)-(3.39) 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 set-up 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 $\ell \in \mathcal{L}(k)$ providing the largest violation of the path inequality, i.e. to the leaf node $\ell_{min} = \arg\ \min_{\ell \in \mathcal{L}(k)} \tilde{E}^k_{p+I} - \varpi_p \sum_{\mu \in \mathcal{U}_{k,\ell}} \left[d^\mu \left(1 - \sum_{m \in \mathcal{P}(c_k^\ell, \mu)} \tilde{Y}_p^m \right) - \tilde{L}^\mu \right]$.

The second idea aims at avoiding the addition of inequalities involving similar subsets of set-up 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 μ_{min}, the last node of the path $\mathcal{P}(c_k^{\ell_{min}}, \ell_{min})$ added to the set $\mathcal{U}_{k,\ell_{min}}$. The inequality added to the formulation involves a subset of set-up variables Y_p^n corresponding to nodes n belonging to the subpath $\mathcal{P}(c_k^{\ell_{min}}, \mu_{min})$.

Algorithm 2: Cutting-plane generation for path inequalities

Data: instances parameters and linear relaxation solution $(\tilde{E}, \tilde{Y}, \tilde{L})$
Result: set of path inequalities χ_{path}

Initialize $\chi_{path} \leftarrow \emptyset$
for $p \in \mathcal{J}$ **do**
  $\mathcal{L}'(\cdot) \leftarrow \mathcal{L}(\cdot)$
  for $k \in \mathcal{V}$ **do**
    $viol_{min} \leftarrow 0$
    for $\ell \in \mathcal{L}'(k)$ **do**
      $\mathcal{U}_{k,\ell} \leftarrow \emptyset$, $\mu_{last} \leftarrow k$
      for $\mu \in \mathcal{P}(c_k^\ell, \ell)$ **do**
        if $\left[d^\mu \left(1 - \sum_{m \in \mathcal{P}(c_k^\ell, \mu)} \tilde{Y}_p^m \right) - \tilde{L}^\mu \right] > 0$ **then**
          $\mathcal{U}_{k,\ell} \leftarrow \mathcal{U}_{k,\ell} \cup \{\mu\}$, $\mu_{last} \leftarrow \mu$
        end
      end
      $viol \leftarrow \tilde{E}^k_{p+I} - \varpi_p \sum_{\mu \in \mathcal{U}_{k,\ell}} \left[d^\mu \left(1 - \sum_{m \in \mathcal{P}(c_k^\ell, \mu)} \tilde{Y}_p^m \right) - \tilde{L}^\mu \right]$
      if $viol < viol_{min}$ **then**
        $viol_{min} \leftarrow viol$, $\ell_{min} \leftarrow \ell$, $\mu_{min} \leftarrow \mu_{last}$
      end
    end
    if $viol_{min} < 0$ **then**
      $\chi_{path} \leftarrow \chi_{path} \cup \left\{ E^k_{p+I} > \varpi_p \sum_{\mu \in \mathcal{U}_{k,\ell_{min}}} \left[d^\mu \left(1 - \sum_{m \in \mathcal{P}(c_k^\ell, \mu)} Y_p^m \right) - L^\mu \right] \right\}$
      for $\mu \in \mathcal{P}(c_k^{\ell_{min}}, \mu_{min})$ **do**
        $\mathcal{L}'(\mu) \leftarrow \mathcal{L}'(\mu) \setminus \{\ell_{min}\}$
      end
    end
  end
end

As the valid inequalities generated when considering the leaf node ℓ_{min} at nodes $n \in \mathcal{P}(c_k^{\ell_{min}}, \mu_{min})$ are likely to involve the same set-up 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 ℓ_{min} is generated at node k at a given iteration, for all $n \in \mathcal{P}(k, \mu_{min})$, ℓ_{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 in Algorithm 2.

3.5.3 Separation algorithm for tree inequalities

Given a non-leaf node k, solving the separation problem for inequalities (3.40) (resp. (3.42)) requires to identify a subset of nodes $\mathcal{U} \subseteq \mathcal{V}(k)$ minimizing the difference between the left-hand side and the right-hand side of inequalitie (3.40) (resp. (3.42)). 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 $\mathcal{U}$ 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 ϕ^m 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 $\mathfrak{s}$ 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 in Algorithm 3.

The intuition behind Algorithm 3 is the following. It first builds an initial set $\mathcal{U}$ containing all nodes in $\mathcal{V}(k)$ that would be selected in the set $\mathcal{U}_{k,\ell}$ when looking for a violated path inequality: see lines 4-9 of Algorithm 3. 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 $\mathcal{U}$: see lines 12-30. More precisely, we start by computing the amount of violation ($viol_{min}$) obtained with the initial set $\mathcal{U}$: this requires to determine the ordering $\mathfrak{s}$ of the leaf nodes in $\mathcal{L}_k$ corresponding to set $\mathcal{U}$ and to compute coefficients ϕ^m for every $m \in \mathcal{V}(k)$. We then explore the neighborhood of set $\mathcal{U}$ which consists of all subsets of $\mathcal{U}$ 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 $\mathfrak{s}$ of the leaf nodes and the coefficients ϕ 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 $\mathcal{U}$ as soon as a better neighbor set $\mathcal{U}'$ is found. Finally, the algorithm stops when no neighbor set $\mathcal{U}'$ has a violation value lower than the one of the current set $\mathcal{U}$.

3.6 Computational experiments

We develop two branch-and-cut algorithms for solving problem (3.1)-(3.11). These algorithms rely on the cutting-plane generation algorithms proposed in Section 3.5 to add valid inequalities to the Echelon Stock reformulation discussed in Section 3.3. 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 [5] and [58] before discussing the detailed results of our computational experiments.

Algorithm 3: Cutting-plane generation for tree inequalities

Data: instances parameters and linear relaxation solution $(\tilde{E}, \tilde{Y}, \tilde{L})$
Result: set of tree inequalities χ_{tree}

Initialize $\chi_{tree} \leftarrow \emptyset$, $\mathcal{V}' = \{\mu \in \mathcal{V} : |\mathcal{E}(\mu)| > 1\}$
for $p \in \mathcal{J}$ **do**
  for $k \in \mathcal{V}'$ **do**
    $\mathcal{U} \leftarrow \emptyset$, $viol_{curr} \leftarrow \infty$
    for $\mu \in \mathcal{V}(k)$ **do**
      if $\left[d^{\mu}\left(1 - \sum_{m \in \mathcal{P}(c_k^{\ell}, \mu)} \tilde{Y}_p^m\right) - \tilde{L}^{\mu}\right] > 0$ **then**
        $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mu\}$
      end
    end
    if $\mathcal{U} = \emptyset$ **then**
      break;
    else
      Determine the ordering $\mathfrak{s}$ of the leaf nodes for set $\mathcal{U}$ and compute ϕ^m for every $m \in \mathcal{V}(k) \setminus \{k\}$
      $viol_{min} \leftarrow \tilde{E}_{p+I}^k + \varpi_p \sum_{\mu \in U} \tilde{L}^{\mu} + \varpi_p \sum_{m \in \mathcal{V}(k) \setminus \{k\}} \phi^m \tilde{Y}_p^m - \varpi_p \sum_{\mu \in \mathcal{U}_{k,\mathfrak{s}|\mathcal{L}(k)|}} d^{\mu}$
      while $viol_{min} < viol_{curr}$ **do**
        $viol_{curr} \leftarrow viol_{min}$
        for $\mu \in \mathcal{U}$ **do**
          $\mathcal{U}' \leftarrow \mathcal{U} \setminus \{\mu\}$
          Update the ordering $\mathfrak{s}$ for set $\mathcal{U}'$ and compute ϕ^m for every $m \in \mathcal{V}(k) \setminus \{k\}$ and
          $viol \leftarrow \tilde{E}_{p+I}^k + \varpi_p \sum_{\mu \in \mathcal{U}'} \tilde{L}^{\mu} + \varpi_p \sum_{m \in \mathcal{V}(k) \setminus \{k\}} \phi^m \tilde{Y}_p^m - \varpi_p \sum_{\mu \in \mathcal{U}_{k,\mathfrak{s}|\mathcal{L}(k)|}} d^{\mu}$
          if $viol < viol_{min}$ **then**
            $viol_{min} \leftarrow viol, \mathcal{U} \leftarrow \mathcal{U}'$
          end
        end
      end
      if $viol_{min} < 0$ **then**
        Update the ordering $\mathfrak{s}$ for the set $\mathcal{U}$ and compute ϕ^m for every $m \in \mathcal{V}(k) \setminus \{k\}$
        $\chi_{tree} \leftarrow \chi_{tree} \cup \left\{E_{p+I}^k > \varpi_p \sum_{\mu \in U} L^{\mu} + \varpi_p \sum_{m \in \mathcal{V}(k) \setminus \{k\}} \phi^m Y_p^m - \varpi_p \sum_{\mu \in \mathcal{U}_{k,\mathfrak{s}|\mathcal{L}(k)|}} d^{\mu}\right\}$
      end
    end
  end
end

3.6.1 Instances generation

The following test data were randomly generated based on the instances generation scheme provided in [5].

- 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 $\varpi_0 = \varpi_I + 1 = 1$, and for each $p = 1, ..., I, \varpi_p$ 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 $\sum_{i=1}^{I} \varpi_i h_{I+i}^n + \varepsilon$ where we generated ϵ from $DU(80, 100)$.

For the proportion of recoverable parts δ_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 [58].

- Low nominal quality level (Q1):, $\delta_i^n \sim DU[8, 25]/100$

- Medium nominal quality level (Q2):, $\delta_i^n \sim DU[11, 58]/100$

- High nominal quality level (Q3):, $\delta_i^n \sim DU[21, 79]/100$

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, $q_i^n = h_i^n * \frac{T}{\epsilon}$, where $\epsilon \sim DU[2, T]$.

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

Regarding the scenario tree structure, we used only balanced trees with Σ stages, a constant number $b = |\mathcal{T}^\sigma|$, for all $\sigma \in \mathcal{S}$, of time periods per stage and a constant number $R = R^\sigma$, for all $\sigma \in \mathcal{S}$, of equiprobable realizations per stage.

We set the number of parts in a product to $I \in \{5, 10\}$, the length of a decision stage to a constant number $b = |\mathcal{T}^\sigma|$ such that $b \in \{1, 2, 3\}$ periods, for all $\sigma \in \mathcal{S}$ and the number of stages to $\Sigma \in \{4, 5, 6, 7, 8, 9\}$. 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 3.2 displays a scenario tree with $(b, \Sigma, c) = (3, 3, 2)$. 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.

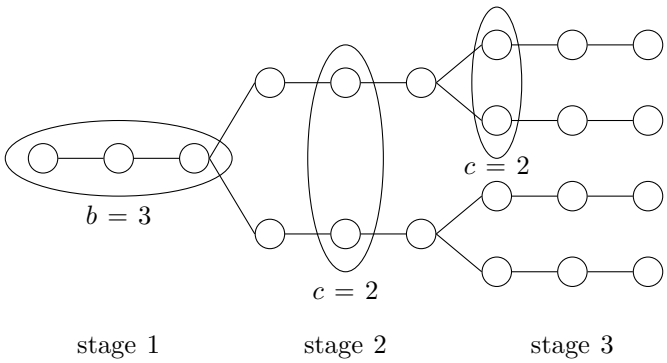


Figure 3.2: Scenario tree structure

3.6.2 Results

Each instance was solved using the echelon stock formulation (3.12)-(3.25) 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 2 and 3 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 3 are used to add tree valid inequalities to the formulation during the course of the branch-and-bound search tree. Note that Algorithm 3 is not only capable to find valid tree inequalities, but also valid path inequalities.

We note that even though the natural formulation (3.1)-(3.11) and the echelon stock formulation (3.12)-(3.25) 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 3.1 and 3.3 for instances with $I = 5$ and Tables 3.2 and 3.4 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.

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 3.1 and 3.2 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 3.1 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 3.2, 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.

In Tables 3.3 and 3.4, 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.

Table 3.1: 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

Table 3.2: 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

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 set-up costs in the objective function. Namely, we note that the integrality gap mainly comes from the fact that the binary constraints on the set-up variables Y_p^n are relaxed so that the value of the fixed set-up costs is underestimated in the linear relaxation. The larger the relative weight of the set-up 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 set-up costs will make up a large portion of the overall production costs, leading to

Table 3.3: 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 3.4: 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

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 set-up 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 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 set-up 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 3.5. 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.

Table 3.5: 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

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. [61], [62], [74], [52], [101] and [38]. 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 3.8. 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.

3.6.3 Value of the stochastic solution

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 assumptions as the ones used by Macedo *et al.* [68] 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.

Rolling horizon simulation

This assessment is achieved by carrying out a rolling horizon simulation similar to the one used by Brandimarte [23]. 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.

Table 3.6: 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

The instances are generated in the same way as in the previous subsection. Since running the simulation is quite costly, we considered small scenario trees with at most 255 nodes in order to gain some basic insights. More specifically, at each iteration of the rolling horizon simulation, a scenario tree with $s = 4$ stages and $b = 3$ periods per stage is generated for the stochastic model. The number of branches c at each stage is set between 2 and 4. The stochastic model is solved by 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 3.6 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 3.6 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.

3.7 Conclusion and perspectives

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.

3.8 Appendix: Additional computational results on large instances with only two sources of uncertainty

In Subsection 3.6.2, 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. [61], [62], [74], [52], [101] and [38]. 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.

3.8.1 Instances generation

We randomly generate instances following the scheme presented in Subsection 3.6.1. The stochastic parameters, i.e. the demand and returns quantity, are generated as described in Subsection 3.6.1. All the other parameters may be time varying but are assumed to be deterministically known. Hence, contrary to what is done in Subsection 3.6.1, 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 3.6.1.

We set the number of parts in a product to $I = 5$ and considered 3 scenario tree structures: $(b, \Sigma, c) = (1, 7, 3)$, $(b, \Sigma, c) = (1, 6, 4)$ and $(b, \Sigma, c) = (5, 6, 3)$ 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.

3.8.2 Results

Each instance was solved using formulation (3.12)-(3.25) by the three alternative branch-and-cut methods detailed in Subsection 3.6.2. The corresponding results are provided in Table 3.7. 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 3.7: 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

Chapter 4

New valid inequalities for lot-sizing problems with remanufacturing and lost sales

This chapter is devoted to the development of new valid inequalities aiming at further strengthening the MILP formulation of the deterministic variant of the problem studied in Chapter 3. We first extend the single-echelon $(k,\mathcal{U})$ inequalities introduced in Chapter 3 to a multi-echelon setting. Second, we introduce a new class of single-echelon inequalities that exploit some specific features of our problem, in particular the fact that the quantity processed on a resource at a given period is limited by the availability of its input product, either stored as such or as one of its ancestor in the bill-of-material. We then further exploit this idea and use it to strengthen the single-echelon and multi-echelon $(k,\mathcal{U})$ inequalities. Finally, extensive computational results are carried out to assess the effectiveness of customized branch-and-cut algorithms based on the proposed valid inequalities.

4.1 Introduction

In Chapter 3, we investigated a stochastic multi-echelon lot-sizing problem with lost sales in order to optimize the production plan of a remanufacturing system. We proposed to solve the resulting mixed-integer linear program through customized branch-and-cut algorithms based on new valid inequalities. Although our numerical results showed that the proposed algorithms were able to solve to near-optimality medium-size instances of the problem, some difficulties could be observed for the largest studied instances. In particular, the results reported in Table 3.1 showed that for the largest studied instances, the value of the average integrality gap Gap_{LP} obtained with the formulation strengthened by the proposed inequalities could be larger than 14% in some cases. This translated in numerical difficulties to provide good quality solutions for the corresponding instances.

Intuitively, this might be due to the fact that the valid inequalities presented in Chapter 3 focused only on strengthening the formulation of the single-echelon uncapacitated lot-sizing sub-problems embedded in the general remanufacturing planning problem. Yet, this problem is intrinsically a multi-echelon lot-sizing problem. Moreover, even if the production resources are assumed to be uncapacitated in the model, the amount of products that can be processed on a resource at a given period is limited among others by the amount of used products returned up to this period. Taking into account these two aspects of the problem in the valid inequalities used to strengthen its MILP formulation might contribute to further reduce its integrality gap and decrease the computational effort needed to solve large instances.

We thus investigate in the present chapter the development of new valid inequalities for this problem. However, in view of the theoretical and numerical difficulties lying ahead, we focus on the deterministic variant of the remanufacturing planning problem investigated in Chapter 3. Namely,

before seeking to extend valid inequalities to a multi-stage stochastic problem expressed on a scenario tree, it is necessary to identify valid inequalities which are efficient at strengthening its deterministic counterpart and to study how to solve the corresponding separation problem.

Throughout the last decade, several works sought to strengthen the MILP formulation of single-echelon lot-sizing problems involving remanufacturing, either through extended reformulations or through valid inequalities. The authors of [86] discussed several MILP formulations of the uncapacitated single-item single-echelon lot-sizing problem with remanufacturing and introduced new valid inequalities by adapting the previously known (ℓ, S, WW) proposed in [78] to their problem. The inequalities developed in [86] are based on the assumption that all returned products are either processed or kept in stock and do not consider the possibility that some of the returns may be discarded in case of an unbalance between returns quantity and demand. They can therefore not be directly used for the problem under study here. Similarly, Cunha et al. [29] proposed a multi-commodity reformulation and a new set of valid inequalities for this problem. In particular, they further strengthened the (ℓ, S, WW) inequalities presented in [86] by considering that the amount of finished products remanufactured in a given period t is limited by the cumulative quantity of returned products brought back up to t. Then, Ali et al. [10] enriched the previous works by highlighting a theoretical property with regards to the equivalence of the shortest path and facility location reformulations. They also carried out a polyhedral analysis of a related sub-problem based on the single node fixed-charge network problem, proving the validity of several flow cover inequalities and their facet-defining conditions as well. Akartunali and Arulselvan [6] studied both the uncapacitated and capacitated variants of this single-item single-echelon lot-sizing problem. They showed that the uncapacitated problem cannot have a fully polynomial time approximation scheme (FPTAS) and provided a pseudo-polynomial algorithm to solve the problem. They also provided valid inequalities based on the flow-cover inequalities for the problem with a limited production capacity. Finally, we refer the reader to [21] for a recent survey on single-item lot-sizing problems with remanufacturing.

Polyhedral approaches for multi-echelon lot-sizing problems not taking into account a remanufacturing option have been recently proposed by a few works. Authors of [18] and [40] considered a multi-echelon production planning problem involving a complex assembly product structure. They used the concept of echelon inventory to construct a new class of valid inequalities based on the well-known (l, S) valid inequalities. Unfortunately, due to the 'classical' assembly structure of our production system, the valid inequalities introduced in [18] and [40] become the classical (l, S)-inequalities in our setting. Zhang et al. [103] studied a multi-echelon uncapacitated lot-sizing problem with a serial product structure in which there is a demand for both the end product and the intermediate products. They proposed a family of two-echelon valid inequalities for this problem, which can be separated in polynomial-time. These inequalities are a particular case of a more general class of valid inequalities called dicut collection inequalities introduced in [85].

Thus, to the best of our knowledge, the problem of developing valid inequalities explicitly taking into account the multi-echelon aspects of lot-sizing with remanufacturing and/or lost sales has not yet been studied. The present work aims at partially closing this gap by proposing new valid inequalities for this variant of lot-sizing problem. Our contributions are thus threefold. We first extend the single-echelon $(k,\mathcal{U})$ inequalities introduced in Chapter 3 to a multi-echelon setting. Second, we introduce a new class of single-echelon inequalities that exploit some specific features of our problem, in particular the fact that the quantity processed on a resource at a given period is limited by the availability of its input product, either stored as such or as one of its ancestor in the bill-of-material. We then further exploit this idea and use it to strengthen the single-echelon and multi-echelon $(k,\mathcal{U})$ inequalities. Finally, extensive computational results are carried out to assess the effectiveness of customized branch-and-cut algorithms based on the proposed valid inequalities.

The remainder of this chapter is organized as follows. We first introduce in Section 4.2.2 two MILP formulations for the deterministic variant of the production planning problem addressed in Chapter 3. We then introduce a new class of multi-echelon $(k,\mathcal{U})$ inequalities in Section 4.3 and new

exponential class of single-echelon valid inequalities in Section 4.4. Then, $(k,\mathcal{U})$ valid inequalities are further strengthened in Section 4.5. Finally, computational results are presented in Section 4.7.

4.2 Mathematical formulations

We consider the remanufacturing system introduced in Subsection 3.2.1 and seek to plan production for this system over a finite planning horizon involving a set $\mathcal{T} = \{1, ..., T\}$ of periods. Contrary to the approach used in Chapter 3, we assume a deterministic environment and thus consider a single possible realization of each input parameter of the production planning problem. We use the same notation as in Subsection 3.2.2 for these input parameters as well as for the decision variables. The only difference is that they are indexed by the corresponding time period t rather than by the corresponding scenario node n.

4.2.1 Initial formulation

We first briefly provide the initial MILP formulation using natural inventory variables S_p^t.

$$\min \sum_{t\in\mathcal{T}} \Big(\sum_{p\in\mathcal{J}} f_p^t Y_p^t + \sum_{i\in\mathcal{I}} h_i^t S_i^t + l^t L^t + \sum_{i\in\mathcal{I}_r\cup\{0\}} q_i^t Q_i^t + g^t X_0^t \Big) \tag{4.1}$$

subject to

$$X_p^t \leq M_p^t Y_p^t \qquad \forall p \in \mathcal{J}, \forall t \in \mathcal{T} \tag{4.2}$$
$$S_0^t = S_0^{t-1} + r^t - X_0^t - Q_0^t \qquad \forall t \in \mathcal{T} \tag{4.3}$$
$$S_i^t = S_i^{t-1} + \delta_i^t \varpi_i X_0^t - X_i^t - Q_i^t \qquad \forall i \in \mathcal{I}_r, \forall t \in \mathcal{T} \tag{4.4}$$
$$S_i^t = S_i^{t-1} + X_{i-I}^t - \varpi_i X_{I+1}^t \qquad \forall i \in I_s, \forall t \in \mathcal{T} \tag{4.5}$$
$$S_{2I+1}^t = S_{2I+1}^{t-1} + X_{I+1}^t - d^t + L^t \qquad \forall t \in \mathcal{T} \tag{4.6}$$
$$S_i^0 = 0 \qquad \forall i \in \mathcal{I} \tag{4.7}$$
$$S_i^t \geq 0 \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.8}$$
$$Q_i^t \geq 0 \qquad \forall i \in \mathcal{I}_r \cup \{0\}, \forall t \in \mathcal{T} \tag{4.9}$$
$$L^t \geq 0 \qquad \forall t \in \mathcal{T} \tag{4.10}$$
$$X_p^t \geq 0, Y_p^t \in \{0,1\} \qquad \forall p \in \mathcal{J}, \forall t \in \mathcal{T} \tag{4.11}$$

The objective function (4.1) aims at minimizing the total remanufacturing cost over the whole planning horizon. This cost comprises the set-up, inventory holding, lost sales and disposal costs. Constraints (4.2) link the production quantity variables to the set-up variables. Note that the value of each constant M_p^t can be computed as described in Section 3.2. Constraints (4.3)-(4.6) are the inventory balance constraints. Without loss of generality, we assume that the initial inventory, S_i^0, is set to 0 for each item $i \in \mathcal{I}$: see Constraints (4.7). Finally, Constraints (4.8)-(4.11) provide the domain of the decision variables.

4.2.2 Echelon stock reformulation

We also provide a reformulation of the problem using the echelon stock concept. 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. We thus denote by E_i^t the echelon stock level of item $i \in \mathcal{I} \setminus \{0\}$ at the end of period t and define it using

the same relationships as the ones used in Subsection 3.3.1. Replacing variables S_i^t by variables E_i^t in Problem (4.1)-(4.11) leads to the following MILP formulation:

$$\min \sum_{t\in\mathcal{T}} \Big(\sum_{p\in\mathcal{J}} f_p^t Y_p^t + h_i^t S_0^t + \sum_{i\in\mathcal{I}\setminus\{0\}} eh_i^t E_i^t + l^t L^t + \sum_{i\in\mathcal{I}_r\cup\{0\}} q_i^t Q_i^t + g_0^t X_0^t \Big) \tag{4.12}$$

subject to

$$X_p^t \leq M_p^t Y_p^t \qquad \forall p \in \mathcal{J}, \forall t \in \mathcal{T} \tag{4.13}$$
$$S_0^t = S_0^{t-1} + r^t - X_0^t - Q_0^t \qquad \forall t \in \mathcal{T} \tag{4.14}$$
$$E_i^t = E_i^{t-1} + \pi_i^t \varpi_i X_0^t - \varpi_i (d^t - L^t) - Q_i^t \qquad \forall i \in \mathcal{I}_r, \forall t \in \mathcal{T} \tag{4.15}$$
$$E_i^t = E_i^{t-1} + X_{i-I}^t - \varpi_i (d^t - L^t) \qquad \forall i \in I_s, \forall t \in \mathcal{T} \tag{4.16}$$
$$E_{2I+1}^t = E_{2I+1}^{t-1} + X_{I+1}^t - d^t + L^t \qquad \forall t \in \mathcal{T} \tag{4.17}$$
$$S_0^0 = 0 \tag{4.18}$$
$$E_i^0 = 0 \qquad \forall i \in \mathcal{I} \setminus \{0\} \tag{4.19}$$
$$E_i^t - E_{I+i}^t \geq 0 \qquad \forall i \in \mathcal{I}_r, \forall t \in \mathcal{T} \tag{4.20}$$
$$E_i^t - \varpi_i E_{2I+1}^t \geq 0 \qquad \forall i \in \mathcal{I}_s, \forall t \in \mathcal{T} \tag{4.21}$$
$$E_i^t \geq 0 \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.22}$$
$$Q_i^t \geq 0 \qquad \forall i \in \mathcal{I}_r \cup \{0\}, \forall t \in \mathcal{T} \tag{4.23}$$
$$S_0^t, L^t \geq 0 \qquad \forall t \in \mathcal{T} \tag{4.24}$$
$$X_p^t \geq 0, Y_p^t \in \{0,1\} \qquad \forall p \in \mathcal{J}, \forall t \in \mathcal{T} \tag{4.25}$$

The objective function (4.12) aims at minimizing the total cost over the whole planning horizon. Constraints (4.13) link the production quantity variables to the setup variables. Constraints (4.14)-(4.17) are the inventory balance constraints. Constraints (4.14) use the classical inventory variables, whereas Constraints (4.15)-(4.17) make use of the echelon inventory variables. Constraints (4.18)-(4.19) translate the fact that the initial inventory of each item is assumed to be equal to 0. Constraints (4.20)-(4.21) ensure consistency between the echelon inventory at the different echelons of the bill-of-material and guarantee that the physical inventory of each product, S_i^t, remains non-negative for all $i \in \mathcal{I}$. Finally, Constraints (4.22)-(4.25) define the domain of the decision variables.

As explained in Subsection 3.3.2, the use of the echelon stock reformulation (4.12)-(4.25) enables us to decompose the initial problem into a series of single-echelon sub-problems by relaxing the linking constraints (4.20)-(4.21). We refer the reader to [1, 19, 2] for a deeper discussion on single-item lot-sizing problems with lost sales. Each of these sub-problems is an uncapacitated single-echelon single-item lot-sizing problem with lost sales, the formulation of which can be strengthened by the $(k,\mathcal{U})$ valid inequalities proposed in [66]. However, this decomposition into single-echelon uncapacitated sub-problems overlooks the fact that the initial problem is intrisically a multi-echelon lot-sizing problem and that the production on each process at a given period is limited by the amount of used products returned up to this period. In what follows, we investigate several classes of valid inequalities in which these aspects of the problem are explicitly taken into account.

4.3 Two-echelon $(k,\mathcal{U})$-inequalities

We first investigate the extension of the single-echelon $(k,\mathcal{U})$ inequalities introduced in [66] for a single-echelon uncapacitated lot-sizing problem with lost sales to a two-echelon setting.

Theorem 1

Let $1 \leq k \leq T$ be a period in the planning horizon and $\mathcal{U} \subseteq \{k+1, ..., T\}$ be a subset of periods. Let $t^* = \max\{\ell \in \mathcal{U}\}$ denote the last period belonging to $\mathcal{U}$. We partition $\mathcal{U}$ into 2 subsets: $\mathcal{U}_1 \subseteq \mathcal{U}$ and $\mathcal{U}_0 = \mathcal{U} \setminus \mathcal{U}_1$.

The following inequalities are valid for Problem (4.12)-(4.25) *for any refurbishing process $p \in \{1, \ldots, I\}$:*

$$\varpi_p^{-1} E_{p+I}^k + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_0) Y_p^t + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_1) Y_{I+1}^t \geq \sum_{t\in\mathcal{U}} (d^t - L^t) \tag{4.26}$$

$$\varpi_p^{-1} E_p^k + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_0) Y_0^t + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_1) Y_p^t \geq \sum_{t\in\mathcal{U}} (d^t - L^t) \tag{4.27}$$

Moreover, the following inequality is valid for Problem (4.12)-(4.25)*:*

$$\min_{i\in\mathcal{I}_r} \varpi_i^{-1} E_i^k + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_0) Y_0^t + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_1) Y_{I+1}^t \geq \sum_{t\in\mathcal{U}} (d^t - L^t) \tag{4.28}$$

with

$$\phi^t(\tilde{\mathcal{U}}) = \sum_{\ell\in\tilde{\mathcal{U}}:\ell\geq t} d^\ell$$

Proof: Let (X, Y, S, E, L, Q) be a feasible solution of (4.12) – (4.25). We show that this solution complies with inequalities (4.26) for any period k, any subset $\mathcal{U}$ and any partition of $\mathcal{U}$ into two subsets $\mathcal{U}_1$ and $\mathcal{U}_0$.

Let $\tau = \min\{\ell \in \{k+1, \ldots, t^*\} : Y_p^\ell = 1\}$ be the first production period of the refurbishing process p on interval $[k+1, t^*]$. By convention, $\tau = t^* + 1$ if no production occurs on process p during this interval.

Similarly, let $\bar{\tau} = \min\{\ell \in \{k+1, \ldots, t^*\} : Y_{I+1}^\ell = 1\}$ be the first production period of the reassembly process $I+1$ on interval $[k+1, t^*]$. By convention, $\bar{\tau} = t^* + 1$ if no production occurs on process $I+1$ during this interval.

- Case (1): $\tau = t^* + 1$

  There is no production on refurbishing process p over periods $k+1$ to t^*. We thus have $X_p^t = 0$ for all t in $[k+1, t^*]$. By summing up the inventory balance constraints (4.16) corresponding to periods $k+1$ to t^*, we have:

  $$\varpi_p^{-1} E_{p+I}^k \geq \sum_{t=k+1\ldots t^*} (d^t - L^t) \geq \sum_{t\in\mathcal{U}} (d^t - L^t)$$

- Case (2): $\bar{\tau} = t^* + 1$

  There is no production on reassembly process $I+1$ over periods $k+1$ to t^*. We thus have $X_{I+1}^t = 0$ for all t in $[k+1, t^*]$. By summing up the inventory balance constraints (4.17) corresponding to periods $k+1$ to t^*, we have:

  $$E_{2I+1}^k \geq \sum_{t=k+1\ldots t^*} (d^t - L^t) \geq \sum_{t\in\mathcal{U}} (d^t - L^t)$$

  By Constraints (4.21), we have $E_{p+I}^k \geq \varpi_p E_{2I+1}^k$, so that:

  $$\varpi_p^{-1} E_{p+I}^k \geq E_{2I+1}^k \geq \sum_{t\in\mathcal{U}} (d^t - L^t)$$

- Case (3): $\bar{\tau} \leq \tau < t^* + 1$

  Similar to Case (1), there is no production on process p on interval $[k+1, \tau-1]$ so that, by summing up Constraints (4.16) over $k+1$ to $\tau - 1$, we obtain:

  $$\varpi_p^{-1} E_{p+I}^k \geq \sum_{t \in \mathcal{U}: t \leq \tau - 1} (d^t - L^t)$$

  Moreover, by definition of periods $\bar{\tau}$ and τ and by using $\bar{\tau} \leq \tau$, we have:

  $$\begin{aligned}
  \varpi_p^{-1} E_{p+I}^k + \sum_{k < t \leq t^*} \phi^t(\mathcal{U}_0) Y_p^t + \sum_{k < t \leq t^*} \phi^t(\mathcal{U}_1) Y_{I+1}^t &\geq \sum_{t \in \mathcal{U}: \leq \tau - 1} (d^t - L^t) + \phi^\tau(\mathcal{U}_0) Y_p^\tau + \phi^{\bar{\tau}}(\mathcal{U}_1) Y_{I+1}^{\bar{\tau}} \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \tau - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}_0: t \geq \tau} d^t + \sum_{t \in \mathcal{U}_1: t \geq \bar{\tau}} d^t \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \tau - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}_0: t \geq \tau} d^t + \sum_{t \in \mathcal{U}_1: t \geq \tau} d^t \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \tau - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}: t \geq \tau} d^t \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \tau - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}: t \geq \tau} (d^t - L^t) \\
  &\geq \sum_{t \in \mathcal{U}} (d^t - L^t)
  \end{aligned}$$

- Case (4): $\tau \leq \bar{\tau} < t^* + 1$

  Similar to Case (2), there is no production on reassembly process $I+1$ on interval $[k+1, \bar{\tau} - 1]$ so that, by summing up Constraints (4.17) over $k+1$ to $\bar{\tau} - 1$ and using Constraints (4.21), we obtain:

  $$\varpi_p^{-1} E_{p+I}^k \geq E_{2I+1}^k \geq \sum_{t \in \mathcal{U}: t \leq \bar{\tau} - 1} (d^t - L^t)$$

  Moreover, by definition of periods $\bar{\tau}$ and τ and by using $\tau \leq \bar{\tau}$, we have:

  $$\begin{aligned}
  \varpi_p^{-1} E_{p+I}^k + \sum_{k < t \leq t^*} \phi^t(\mathcal{U}_0) Y_p^t + \sum_{k < t \leq t^*} \phi^t(\mathcal{U}_1) Y_{I+1}^t &\geq \sum_{t \in \mathcal{U}: \leq \bar{\tau} - 1} (d^t - L^t) + \phi^\tau(\mathcal{U}_0) Y_p^\tau + \phi^{\bar{\tau}}(\mathcal{U}_1) Y_{I+1}^{\bar{\tau}} \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \bar{\tau} - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}_0: t \geq \tau} d^t + \sum_{t \in \mathcal{U}_1: t \geq \bar{\tau}} d^t \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \bar{\tau} - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}_0: t \geq \bar{\tau}} d^t + \sum_{t \in \mathcal{U}_1: t \geq \bar{\tau}} d^t \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \bar{\tau} - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}: t \geq \bar{\tau}} d^t \\
  &\geq \sum_{t \in \mathcal{U}: t \leq \bar{\tau} - 1} (d^t - L^t) + \sum_{t \in \mathcal{U}: t \geq \bar{\tau}} (d^t - L^t) \\
  &\geq \sum_{t \in \mathcal{U}} (d^t - L^t)
  \end{aligned}$$

This concludes the proof for inequalities (4.26). Note that proving the validity of inequalities (4.27) and (4.28) can be done using the same arguments and steps of the above proof. □

In order to get a more intuitive interpretation of these valid inequalities and ease the resolution of the corresponding separation algorithm, we note that e.g. valid inequalities (4.26) may be rewritten as:

$$\varpi_p^{-1} E_{p+I}^k \geq \sum_{t \in \mathcal{U}_0} [d^t (1 - \sum_{\tau = k+1}^{t} Y_p^\tau) - L^t] + \sum_{t \in \mathcal{U}_1} [d^t (1 - \sum_{\tau = k+1}^{t} Y_{I+1}^\tau) - L^t] \tag{4.29}$$

The intuition underlying these inequalities can be understood as follows. We consider the echelon stock of serviceable component $p+I$ at the end of period k and look at the future demand for the end products. For a demand d^t arising in a period t belonging to $\mathcal{U}_0$, if $\sum_{\tau=k+1}^{t} Y_p^\tau \geq 1$, it means that some refurbishing for component $p+I$ will happen over the interval $[k+1,t]$ so that it will be possible to refurbish the $\varpi_p(d^t - L^t)$ recovered components necessary to satisfy the demand at period t. But if $\sum_{\tau=k+1}^{t} Y_p^\tau = 0$, no serviceable component $p+I$ might be obtained over the interval $[k+1,t]$. In this case, the $\varpi_p(d^t - L^t)$ recovered components necessary to satisfy the demand at period t should already be in the echelon stock E_{p+I}^k at the end of period k. As for a demand d^t arising in a period t belonging to $\mathcal{U}_1$, we note that, if $\sum_{\tau=k+1}^{t} Y_{I+1}^\tau = 0$, it will not be possible to reassemble any remanufactured products between $k+1$ and t. The $(d^t - L^t)$ units of remanufactured products needed to satisfy the demand in t must thus already be in stock at the end of period k., i.e. must be in S_{2I+1}^k. As $E_{p+I}^k = S_{2I+1}^k + \varpi_p S_{p+I}^k$, this means that $\varpi_p(d^t - L^t)$ must already be in the echelon stock E_{p+I}^k at the end of period k.

Separation algorithm

Regarding the separation problem for these two-echelon valid inequalities, we note that it can be solved in $\mathcal{O}(T^2)$. Namely, to find e.g. the valid inequality (4.26) most violated by the current solution$(\tilde{X}, \tilde{Y}, \tilde{S}, \tilde{E}, \tilde{L}, \tilde{Q})$ of the linear relaxation of Problem (4.12)-(4.25), we need to find, for each period $k \in \mathcal{T}$, the subsets $\mathcal{U}_0$ and $\mathcal{U}_1$ maximizing the right-hand side of inequality (4.29).

We thus consider a exact separation algorithm in our computational experiments, which can be summarized as follows. For a given process p:

Algorithm 4: Separation algorithm for two-echelon $(k,\mathcal{U})$

```
for k ∈ 𝒯 do
    for t = k+1, ..., T do
        compute m = min{d^t(1 − Σ_{τ=k+1}^{t} Ỹ_p^τ) − L̃^t; d^t(1 − Σ_{τ=k+1}^{t} Ỹ_{I+1}^τ) − L̃^t}
        if m ≤ 0 then
            t ∉ 𝒰
        end
        else
            if m = d^t(1 − Σ_{τ=k+1}^{t} Ỹ_p^τ) − L̃^t then
                t ∈ 𝒰_0
            end
        end
        else
            t ∈ 𝒰_1
        end
    end
end
```

Remark 1

We note that most of the single-echelon $(k,\mathcal{U})$ inequalities (3.38)-(3.39) investigated in Chapter 3 can be seen as special cases of the two-echelon inequalities (4.26)-(4.28). More precisely:

- *for each refurbishing process $p \in \{1,...,I\}$, the single-echelon inequality (3.38) relative to period k and set $\mathcal{U}$ corresponds to the two-echelon valid inequality (4.26) relative to period k, set $\mathcal{U}$ and subset $\mathcal{U}_1 = \emptyset$.*
- *similarly, for the disassembly process, the single-echelon inequality (3.39) relative to period*

k and set $\mathcal{U}$ corresponds to the two-echelon valid inequality (4.28) *relative to period k, set $\mathcal{U}$ and subset $\mathcal{U}_1 = \emptyset$.*

- *however, for the reassembly process $p = I + 1$, the single-echelon inequality* (3.38) *relative to period k and set $\mathcal{U}$ involves the echelon stock E^k_{2I+1} whereas the two-echelon valid inequality* (4.26) *relative to period k, set $\mathcal{U}$ and subset $\mathcal{U}_1 = \emptyset$ involves the echelon stock $\varpi_p^{-1}E^k_p$, which is such that $\varpi_p^{-1}E^k_p \leq E^k_{2I+1}$. This implies that the single-echelon inequality* (3.38) *might be stronger than the corresponding two-echelon valid inequality* (4.26) *and thus cannot be considered as a special case of a two-echelon valid inequality* (4.26).

Finally, we note that it is possible to extend the two-echelon valid inequalities (4.27) to obtain three-echelon valid inequalities simultaneously involving set-up variables relative to the disassembly, refurbishing and reassembly processes.

Corollary 1

Let $1 \leq k \leq T$ be a period in the planning horizon.
Let $\mathcal{U} \subseteq \{k+1, ..., T\}$ be a subset of periods. $t^* = \max\{\ell \in \mathcal{U}\}$ denotes the last period belonging to $\mathcal{U}$. We partition $\mathcal{U}$ into 3 disjoint subsets $\mathcal{U}_0$, $\mathcal{U}_1$ and $\mathcal{U}_2$ such that $\mathcal{U}_0 \cup \mathcal{U}_1 \cup \mathcal{U}_2 = \mathcal{U}$.

For any refurbishing process $p \in \{1, ...I\}$, the following inequality is valid for Problem (4.12)-(4.25)*:*

$$\varpi_p^{-1}E^k_p + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_0)Y^t_0 + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_1)Y^t_p + \sum_{k<t\leq t^*} \phi^t(\mathcal{U}_2)Y^t_{I+1} \geq \sum_{t\in\mathcal{U}}(d^t - L^t) \tag{4.30}$$

with $\phi^t(\tilde{\mathcal{U}}) = \sum_{\ell\in\tilde{\mathcal{U}}:\ell\geq t} d^\ell$.

Proof: The proof of Corollary 1 can be done by using the same arguments and steps as the ones used in the proof of Theorem 1. □

Remark 2

It is worth noticing that inequalities (4.26)-(4.28) *are closely related to the valid inequalities (13.13) presented in Chapter 13 of [79] for a multi-echelon lot-sizing problem with a serial product structure and no lost sales. We refer the reader to [79] for numerical and graphical examples of this type of inequalities. Thus, inequalities* (4.26)-(4.28) *can be reformulated by following the spirit of inequalities (13.13) in [79]. In this case, given the demand at time period t, rather than assuming d^t is satisfied individually for each process, we consider that d^t can be partially satisfied by each process. This idea leads to the following reformulation of inequalities* (4.26),

$$\varpi_p^{-1}E^k_{p+I} \geq \sum_{t\in\mathcal{U}}[d^t(1 - \sum_{\tau=k+1}^{\theta_t} Y^\tau_p - \sum_{\tau=\theta_t+1}^{t} Y^\tau_{I+1}) - L^t] \tag{4.31}$$

Thus, inequalities (4.31) *entail to find a time period θ_t for each $t \in \mathcal{U}$, which defines a sequence of set-up variables of process p and $I + 1$ that aims at satisfying demand d^t. As a results, inequalities* (4.26) *can be seeing as a particular case of inequalities* (4.31). *However, the separation of inequalities* (4.31) *is more time-consuming than the separation of inequalities* (4.26), *due to the former entails to find the sequence of set-up variables that maximize the right-hand side of* (4.31).

On the other hand, we might consider the case when $\theta_t = \theta$ for each $t \in \mathcal{U}$, this leads to the

following inequalities.

$$\varpi_p^{-1} E_{p+I}^k \geq \sum_{t \in \mathcal{U}} [d^t (1 - \sum_{\tau=k+1}^{\theta} Y_p^\tau - \sum_{\tau=\theta+1}^{t} Y_{I+1}^\tau) - L^t] \tag{4.32}$$

However, inequalities (4.32) *do not contain all inequalities* (4.26)*. It is easy to find an example to illustrate this fact. Let us consider the time period k and the demand for the next three time periods $k+1, k+2, k+3$. Given a linear relaxation with the following values for Y_p, $Y_p^{k+1} = 0.3, Y_p^{k+2} = 0.2, Y_p^{k+3} = 0.1$ and for Y_{I+1}, $Y_{I+1}^{k+1} = 0.1, Y_{I+1}^{k+2} = 0.3, Y_{I+1}^{k+3} = 0.3$, we can generate the following two inequalities for $\theta = \{k+1, k+2\}$ and $\mathcal{U} = \{k+1, k+2, k+3\}$,*

for $\theta = k+1$,

$$\varpi_p^{-1} E_{p+I}^k \geq [d^{k+1}(1 - Y_p^{k+1}) - L^{k+1}] + [d^{k+2}(1 - Y_p^{k+1} - Y_{I+1}^{k+2}) - L^{k+2}] + [d^{k+3}(1 - Y_p^{k+1} - Y_{I+1}^{k+2} - Y_{I+1}^{k+3}) - L^{k+3}]$$

$$\Rightarrow \varpi_p^{-1} E_{p+I}^k \geq [d^{k+1}(1 - 0.3) - L^{k+1}] + [d^{k+2}(1 - 0.3 - 0.3) - L^{k+2}] + [d^{k+3}(1 - 0.3 - 0.3 - 0.3) - L^{k+3}]$$

for $\theta = k+2$,

$$\varpi_p^{-1} E_{p+I}^k \geq [d^{k+1}(1 - Y_p^{k+1}) - L^{k+1}] + [d^{k+2}(1 - Y_p^{k+1} - Y_p^{k+2}) - L^{k+2}] + [d^{k+3}(1 - Y_p^{k+1} - Y_p^{k+2} - Y_{I+1}^{k+3}) - L^{k+3}]$$

$$\Rightarrow \varpi_p^{-1} E_{p+I}^k \geq [d^{k+1}(1 - 0.3) - L^{k+1}] + [d^{k+2}(1 - 0.3 - 0.2) - L^{k+2}] + [d^{k+3}(1 - 0.3 - 0.2 - 0.3) - L^{k+3}]$$

and the following inequality (4.26) *can be generated by setting $\mathcal{U}_1 = \{k+2\}$,*

$$\varpi_p^{-1} E_{p+I}^k \geq [d^{k+1}(1 - Y_p^{k+1}) - L^{k+1}] + [d^{k+2}(1 - Y_{I+1}^{k+1} - Y_{I+1}^{k+2}) - L^{k+2}] + [d^{k+3}(1 - Y_p^{k+1} - Y_p^{k+2} - Y_p^{k+3}) - L^{k+3}]$$

$$\Rightarrow \varpi_p^{-1} E_{p+I}^k \geq [d^{k+1}(1 - 0.3) - L^{k+1}] + [d^{k+2}(1 - 0.1 - 0.3) - L^{k+2}] + [d^{k+3}(1 - 0.3 - 0.2 - 0.1) - L^{k+3}]$$

Clearly, the last inequalities are stronger than the previous ones and cannot be generated for any possible value of θ. Note that it is also possible to find a stronger inequality (4.32) *that cannot be generated by any set $\mathcal{U}_1$, given the same set $\mathcal{U}$.*

4.4 Single-echelon $(\ell, \mathcal{U})$-returns inequalities

The valid inequalities introduced so far are primarily focused on the demand satisfaction and do not consider a key aspect in remanufacturing systems, which is the availability of returned products in the production system. We thus investigate a family of new single-echelon inequalities which seek to better take into account the fact that the production quantity that can be processed on a resource is limited by the availability of its input product. More precisely, the quantity that can be processed on a resource p in period t is limited by the amount of used products which has been returned up to period t but not yet processed on resource p before this period. We thus introduce, for each

production echelon and each production resource $p \in \{0, ..., I+1\}$, a set of valid inequalities stating that the total quantity processed on p over a given time interval $[\ell, t]$ is limited by the sum of the amount of used products already returned and not yet transformed on p at the end of period $\ell - 1$ and of the amount of used products returned during interval $[\ell, t]$.

We start by defining the corresponding $(\ell, \mathcal{U})$-returns inequalities for the disassembly process $p = 0$.

Theorem 2

Let $1 \leq \ell \leq T$ be a period in the planning horizon and $\mathcal{U} \subseteq \{\ell, ..., T\}$ be a subset of periods. Then, the following inequality is valid for Problem (4.1)-(4.11)*:*

$$S_0^{\ell-1} + \sum_{t \in \mathcal{U}} \phi^t Y_0^t \geq \sum_{t \in \mathcal{U}} X_0^t \tag{4.33}$$

with $\phi^t = \sum_{\ell \leq \nu \leq t} r^\nu$

Proof: Let (X, Y, S, Q, L) be a feasible solution of Problem (4.1)-(4.11). We show that this solution complies with inequalities (4.33) for any period ℓ and any subset $\mathcal{U}$. This is done by induction on the index of the last period t that may belong to subset $\mathcal{U}$.

Base step: $t = \ell$ and $\mathcal{U} \subset \{\ell\}$.

We consider two cases:

- $\mathcal{U} = \emptyset$
  In this case, $\sum_{\tau \in \mathcal{U}} X_0^\tau = 0$ whereas $S_0^{\ell-1} + \sum_{\tau \in \mathcal{U}} \phi^\tau Y_0^\tau \geq S_0^{\ell-1} \geq 0$. Inequality (4.33) is trivially respected.
- $\mathcal{U} = \{\ell\}$
  - if $Y_0^\ell = 0$
  There is no production in period ℓ so that $\sum_{\tau \in \mathcal{U}} X_0^\tau = X_0^\ell = 0$ whereas $S_0^{\ell-1} + \sum_{\tau \in \mathcal{U}} \phi^\tau Y_0^\tau \geq S_0^{\ell-1} \geq 0$. Inequality (4.33) is trivially respected.
  - if $Y_0^\ell = 1$
  In this case, the inventory balance constraint (4.3) for period ℓ gives:

$$\begin{aligned} S_0^{\ell-1} + r^\ell - X_0^\ell - Q_0^\ell &\geq S_0^\ell \\ S_0^{\ell-1} + r^\ell - X_0^\ell &\geq 0 \qquad \text{by non-negativity of variables } S_0^\ell \text{ and } Q_0^\ell. \end{aligned}$$

  By noting that, under the current assumptions, $\phi^\ell Y_0^\ell = r^\ell$, we have:

$$S_0^{\ell-1} + \phi^\ell Y_0^\ell \geq X_0^\ell$$

  which corresponds to the expression of Inequality (4.33) for $\mathcal{U} = \{\ell\}$.
  Inequality (4.33) is thus valid for any $\mathcal{U} \subset \{\ell\}$.

Induction step: Assume that Inequality (4.33) is valid for any $\mathcal{U} \subset \{1, ..., t\}$. We show that, under this assumption, Inequality (4.33) is valid for any $\mathcal{U} \subset \{1, ..., t+1\}$.

We again consider two cases:

- $t + 1 \notin \mathcal{U}$.
  Using the induction hypothesis, Inequality (4.33) is valid for $\mathcal{U}$.
- $t + 1 \in \mathcal{U}$.
  - if $Y_0^{t+1} = 0$
  There is no production in period $t + 1$ so that $\sum_{\tau \in \mathcal{U}} X^\tau = \sum_{\tau \in \mathcal{U} \setminus \{t+1\}} X^\tau$ whereas $S_0^{\ell-1} + \sum_{\tau \in \mathcal{U}} \phi^\tau Y_0^\tau = S_0^{\ell-1} + \sum_{\tau \in \mathcal{U} \setminus \{t+1\}} \phi^\tau Y_0^\tau$.

Using the induction hypothesis, we have: $S_0^{\ell-1} + \sum_{\tau\in\mathcal{U}\setminus\{t+1\}} \phi^\tau Y_0^\tau \geq \sum_{\tau\in\mathcal{U}\setminus\{t+1\}} X_0^\tau$ so that Inequality (4.33) holds.

- if $Y_0^{t+1} = 1$

Summing up the inventory balance constraints (4.3) over periods ℓ to $t+1$ and using the non-negativity of variables S_0^ℓ, Q_0^τ and X_0^τ gives:

$$S_0^{\ell-1} + \sum_{\tau=\ell}^{t+1} r_\tau - \sum_{\tau=\ell}^{t+1} X_0^\tau - \sum_{\tau=\ell}^{t+1} Q_0^\tau \geq S_0^\ell$$

$$S_0^{\ell-1} + \sum_{\tau=\ell}^{t+1} r_\tau - \sum_{\tau=\ell}^{t+1} X_0^\tau \geq 0$$

$$S_0^{\ell-1} + \sum_{\tau=\ell}^{t+1} r_\tau - \sum_{\tau\in\mathcal{U}} X_0^\tau \geq 0$$

By noting that under the current assumption, $\phi^{t+1}Y_0^{t+1} = \sum_{\tau=\ell}^{t+1} r_\tau$ and adding the non-negative terms $\sum_{\tau\in\mathcal{U}\setminus\{t+1\}} \phi^\tau Y_0^\tau$ to the left-hand side of the previous inequality, we obtain:

$$S_0^{\ell-1} + \sum_{\tau\in\mathcal{U}} \phi^\tau Y_0^\tau - \sum_{\tau\in\mathcal{U}} X_0^\tau \geq 0$$

This shows that if Inequality (4.33) is valid for any $\mathcal{U} \subset \{1,...,t\}$, it is also valid for any $\mathcal{U} \subset \{1,...,t+1\}$, which concludes the proof. □

We now seek to generalize Inequalities (4.33) and consider $(\ell,\mathcal{U})$-returns inequalities for the refurbishing processes. Let $\hat{\delta}_i^{\ell,t} = \max_{\ell\leq v\leq t} \delta_i^v$ denote the maximum value of δ_i^v between the time period ℓ and t, i.e. the bet disassembly yield than can be obtained for item i over interval $[\ell,t]$.

Theorem 3

Let $1 \leq \ell \leq T$ be a period in the planning horizon. Let $\mathcal{U} \subseteq \{\ell,...,T\}$ be a subset of periods and $t^* = \max\{\tau : \tau \in \mathcal{U}\}$ the last period belonging to $\mathcal{U}$. Then, for each refurbishing process $p \in \{1,...,I\}$, the following inequality is valid for Problem (4.1)-(4.11):

$$\varpi_p \hat{\delta}_p^{\ell,t^*} S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{t\in\mathcal{U}} \phi^t Y_p^t \geq \sum_{t\in\mathcal{U}} X_p^t \tag{4.34}$$

with $\phi^t = \sum_{\ell\leq\nu\leq t} (r^\nu \hat{\delta}_p^{\nu t})$.

Proof: Let (X,Y,S,Q,L) be a feasible solution of Problem (4.1)-(4.11). We show that this solution complies with inequalities (4.34) for any period ℓ and any subset $\mathcal{U}$. This is done by induction on the index of the last period t that may belong to subset $\mathcal{U}$.

Base step: $t = \ell$ and $\mathcal{U} \subset \{\ell\}$.

We consider two cases:

- $\mathcal{U} = \emptyset$

  In this case, $\sum_{t\in\mathcal{U}} X_p^t = 0$ whereas $\varpi_p \delta_p^\ell S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{t\in\mathcal{U}} \phi^t Y_p^t \geq S_p^{\ell-1} \geq 0$. Inequality (4.34) is trivially respected.

- $\mathcal{U} = \{\ell\}$

  - if $Y_p^\ell = 0$

  There is no production in period ℓ so that $\sum_{t\in\mathcal{U}} X_p^t = 0$ whereas $\varpi_p \delta_p^\ell S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{t\in\mathcal{U}} \phi^t Y_p^t \geq 0$. Inequality (4.34) is trivially respected.

- if $Y_p^\ell = 1$

By carrying out the linear combination $\varpi_p \delta_p^\ell$ $(4.3)_\ell + (4.4)_\ell$ of the inventory balance equations corresponding to period ℓ and using the non-negativity of variables Q_0^ℓ, Q_p^ℓ, S_0^ℓ and S_p^ℓ, we obtain:

$$\varpi_p \delta_p^\ell [S_0^{\ell-1} + r^l - X_0^\ell - Q_0^\ell] + S_p^{\ell-1} + \varpi_p \delta_p^\ell X_0^\ell - X_p^\ell - Q_p^\ell \geq \varpi_p \delta_p^\ell S_0^\ell + S_p^\ell$$
$$\varpi_p \delta_p^\ell S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \delta_p^\ell r^l - X_p^\ell \geq 0$$

We note that, under the current assumptions, $\phi^\ell Y_p^\ell = \delta_p^\ell r^l$ so that we have:

$$\varpi_p \delta_p^\ell S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \phi^\ell Y_p^\ell \geq X_p^\ell$$

Inequality (4.34) is thus valid for any $\mathcal{U} \subset \{\ell\}$.

Induction step: Assume that Inequality (4.34) is valid for any $\mathcal{U} \subset \{1, ..., t\}$. We show that, under this assumption, Inequality (4.34) is valid for any $\mathcal{U} \subset \{1, ..., t+1\}$.

We consider two cases:

- $t+1 \notin \mathcal{U}$

  Using the induction hypothesis, Inequality (4.34) is valid for $\mathcal{U}$.

- $t+1 \in \mathcal{U}$

  - if $Y_p^{t+1} = 0$

  We have:

$$\varpi_p \hat{\delta}_p^{\ell,t+1} S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{t \in \mathcal{U}} \phi^t Y_p^t - \sum_{t \in \mathcal{U}} X_p^t \tag{4.35}$$
$$= \varpi_p \hat{\delta}_p^{\ell,t+1} S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{t \in \mathcal{U} \setminus \{t+1\}} \phi^t Y_p^t - \sum_{t \in \mathcal{U} \setminus \{t+1\}} X_p^t \tag{4.36}$$
$$\geq \varpi_p \hat{\delta}_p^{\ell,t} S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{t \in \mathcal{U} \setminus \{t+1\}} \phi^t Y_p^t - \sum_{t \in \mathcal{U} \setminus \{t+1\}} X_p^t \tag{4.37}$$
$$\geq 0 \tag{4.38}$$

Equality (4.36) holds because there is no production in period $t+1$ so that $\sum_{t \in \mathcal{U}} \phi^t Y_p^t = \sum_{t \in \mathcal{U} \setminus \{t+1\}} \phi^t Y_p^t$ and $\sum_{t \in \mathcal{U}} X_p^t = \sum_{t \in \mathcal{U} \setminus \{t+1\}} X_p^t$. Moroever, Inequality (4.37) exploits the fact that $\hat{\delta}_p^{\ell,t+1} \geq \hat{\delta}_p^{\ell,t}$. Finally, the induction hypothesis gives Inequality (4.38). This shows that Inequality (4.34) is valid in this case.

  - if $Y_p^{t+1} = 1$

  For each period $\tau \in [\ell, t+1]$, we compute the linear combination $\varpi_p \hat{\delta}_p^{\tau,t+1} (4.3)_\tau + (4.4)_\tau$ of the corresponding inventory balance equations. This gives:

$$\varpi_p \hat{\delta}_p^{\tau,t+1} [S_0^{\tau-1} + r^\tau - X_0^\tau - Q_0^\tau] + S_p^{\tau-1} + \varpi_p \delta_p^\tau X_0^\tau - X_p^\tau - Q_p^\tau = \varpi_p \hat{\delta}_p^{\tau,t+1} S_0^\tau + S_p^\tau$$

Thanks to the non-negativity of variables X_0^τ, Q_0^τ and Q_p^τ and the fact that $\delta_p^\tau \leq \hat{\delta}_p^{\tau,t+1}$, we obtain:

$$\varpi_p \hat{\delta}_p^{\tau,t+1} S_0^{\tau-1} + S_p^{\tau-1} + \varpi_p \hat{\delta}_p^{\tau,t+1} r^\tau - X_p^\tau \geq \varpi_p \hat{\delta}_p^{\tau,t+1} S_0^\tau + S_p^\tau$$

Summing up this inequality over periods ℓ to $t+1$ gives:

$$\varpi_p \Big(\sum_{\tau=\ell}^{t+1} \hat{\delta}_p^{\tau,t+1} S_0^{\tau-1} - \sum_{\tau=\ell}^{t+1} \hat{\delta}_p^{\tau,t+1} S_0^\tau \Big) + S_p^{t+1} + S_p^{\ell-1} + \varpi_p \sum_{\tau=\ell}^{t+1} \hat{\delta}_p^{\tau,t+1} r^\tau - \sum_{\tau=\ell}^{t+1} X_p^\tau \geq 0 \tag{4.39}$$

Moreover, we have:

$$\sum_{\tau=\ell}^{t+1} \hat{\delta}_p^{\tau,t+1} S_0^{\tau-1} - \sum_{\tau=\ell}^{t+1} \hat{\delta}_p^{\tau,t+1} S_0^{\tau} = \hat{\delta}_p^{\ell,t+1} S_0^{\ell-1} + \sum_{\tau=\ell+1}^{t} \left(\hat{\delta}_p^{\tau+1,t+1} - \hat{\delta}_p^{\tau,t+1}\right) S_0^{\tau} - (\hat{\delta}_p^{t+1,t+1}) S_0^{t+1}$$
$$\leq \hat{\delta}_p^{\ell,t+1} S_0^{\ell-1} \tag{4.40}$$

Replacing the corresponding terms in Inequalities (4.40) with Inequalities (4.39) we obtain:

$$\varpi_p \hat{\delta}_p^{\ell,t+1} S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{\tau=\ell}^{t+1} \hat{\delta}_p^{\tau,t+1} r^{\tau} - \sum_{\tau=\ell}^{t+1} X_p^{\tau} \geq 0$$

We then note that, under the current assumptions, $\phi^{t+1} Y_p^{t+1} = \sum_{\tau=\ell}^{t+1} \hat{\delta}_p^{\tau,t+1} r^{\tau}$. Moreover, by non-negativity of variables X_p^{τ}, we have $\sum_{\tau=\ell}^{t+1} X_p^{\tau} \geq \sum_{\tau \in \mathcal{U}} X_p^{\tau}$. This leads to:

$$\varpi_p \hat{\delta}_p^{\ell,t+1} S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \phi^{t+1} Y_p^{t+1} - \sum_{\tau \in \mathcal{U}} X_p^{\tau} \geq 0$$

Finally, adding the non-negative term $\sum_{\tau \in \mathcal{U} \setminus \{t+1\}} Y_p^{\tau}$ to the left hand side of this inequality gives:

$$\varpi_p \hat{\delta}_p^{\ell,t+1} S_0^{\ell-1} + S_p^{\ell-1} + \varpi_p \sum_{\tau \in \mathcal{U}} \phi^{\tau} Y_p^{\tau} - \sum_{\tau \in \mathcal{U}} X_p^{\tau} \geq 0$$

Inequality (4.34) is thus valid in this case.

This shows that if Inequality (4.34) is valid for any $\mathcal{U} \subset \{1, ..., t\}$, it is also valid for any $\mathcal{U} \subset \{1, ..., t+1\}$, which concludes the proof. □

The intuition behind Inequalities (4.34) can be explained as follows. Given a time period ℓ and a period $t \in \mathcal{U}$, the maximum amount than can be processed at time period t on resource p is limited by the availability of the input product of resource p, i.e. by the availability of recoverable component p. This availability is computed in two steps. (1) We first look at the total quantity of item p available in the system at the end of period $\ell - 1$. This quantity takes into account both the amount of product p kept in inventory as such (and recorded by $S_p^{\ell-1}$) and the amount of item p available in the system under the form of an ancestor of p in the bill-of-material, i.e. under the form of a component embedded in a used product. This second quantity, computed as $\varpi_p \hat{\delta}_p^{\ell,t^*} S_0^{\ell-1}$, corresponds to the maximum amount of item p that might be recovered by disassembling the used products held in inventory at the end of period $\ell - 1$. It is obtained by considering that all the used products available in stock at the end of period $\ell - 1$, $S_0^{\ell-1}$, will be disassembled during the period $\nu \in [\ell; t]$ in which the disassembly yield for item p, δ_p^{ν}, is the largest. (2) We then look at the used products returned between period ℓ and period t. We compute an upper bound of the amount of product p that might be recovered by disassembling these used products. This is done by assuming that each returned quantity $r^{\tau}, \tau = \ell, ..., t$, will be disassembled during the period $\nu \in [\tau; t]$ in which the disassembly yield for product p, δ_p^{ν}, is the largest. This corresponds to the term $\phi^t Y_p^t$ in the expression of Inequalities (4.34). Finally, we introduce $(\ell, \mathcal{U})$-returns inequalities for the reassembly processes.

Theorem 4

Let $1 \leq \ell \leq T$ be a period in the planning horizon. Let $\mathcal{U} \subseteq \{\ell, ..., T\}$ be a subset of periods and

$t^* = \max\{\tau : \tau \in \mathcal{U}\}$ *the last period belonging to* $\mathcal{U}$.
The following inequality is valid for Problem (4.1)-(4.11)*:*

$$S_0^{\ell-1}\hat{\delta}_i^{\ell,t^*} + \varpi_i^{-1}S_i^{\ell-1} + \varpi_i^{-1}S_{i+I}^{\ell-1} + \sum_{t\in\mathcal{U}} \phi^t Y_{I+1}^t \geq \sum_{t\in\mathcal{U}} X_{I+1}^t \tag{4.41}$$

with $\phi^t = \sum_{\ell\leq\nu\leq t}(r^\nu \hat{\delta}_i^{\nu,t})$

Proof: The proof is straightforward following the proof of (4.34) and using linear combinations of the inventory balance equations (4.3) -(4.5). □

Separation algorithm

We note that inequalities (4.33), (4.34) and (4.41) form an exponential class of valid inequalities, i.e. their number grows exponentially fast with the number of time periods T. However, the corresponding separation problem is polynomially solvable in $\mathcal{O}(T^2)$. Namely, to find e.g. the violated valid inequalities (4.34) most violated by the current linear relaxation $(\tilde{X}, \tilde{Y}, \tilde{S}, \tilde{L}, \tilde{Q})$ of Problem (4.1)-(4.11), we need to find, for each period $\ell \in \mathcal{T}$, the subset $\mathcal{U}$ minimizing the difference between the left and the right-hand side of inequality (4.34). This amounts to finding the subset $\mathcal{U}$ minimizing $\sum_{t\in\mathcal{U}} \phi^t \tilde{Y}_p^t - \sum_{t\in\mathcal{U}} \tilde{X}_p^t$.

We thus consider a exact separation algorithm in our computational experiments, which can be summarized as follows. For a given process p:

Algorithm 5: Separation algorithm for $(\ell, \mathcal{U})$

```
for ℓ ∈ 𝒯 do
    for t = ℓ + 1, ..., T do
        if φ^t Ỹ_p^t − X̃_p^t ≤ 0 then
            t ∈ 𝒰
        end
        else
            t ∉ 𝒰
        end
    end
end
```

Remark 3

Note that this idea differs from classical approaches, where coefficients ϕ_t *are strengthened by consider the sum of the returns from the beginning of the planning horizon to the time period* t *and do not take into account actual available quantity of returned products at time period* t *(see for example inequalities (32) in [30]). Therefore, these new inequalities attempt to better approximate this quantity at each time period. On the other hand, we note that Inequalities* (4.33) *significantly differ from the ones introduced in [86] to strengthen the linear relaxation of a single-echelon hybrid manufacturing/remanufacturing production system. These inequalities namely consider a time interval* $[k, \ell]$ *and compute a lower bound on the inventory of returned product* S_0^ℓ *at the end of the considered interval based on the past returns and remanufacturing activity :*

$$S_0^\ell + \sum_{k\leq t\leq\ell} \phi^t Y_0^t \geq \sum_{k\leq t\leq\ell} r^t \quad \forall \quad 1 \leq k \leq \ell \leq T \tag{4.42}$$

with $\phi^t = \sum_{k \leq \nu \leq t} r^\nu$.

In contrast, Inequalities (4.33) *consider a time interval denoted by* $[\ell, t^*]$ *and use the inventory of returned product* S_0^ℓ *at the beginning of this time interval to compute an upper bound on the future disassembly activity. Note that Inequalities* (4.42) *cannot be directly applied to our problem. They namely consider that all the returned products must be processed on the remanufacturing resource whereas our model allows to discard some returned products without processing them. Moreover, it is worth mentioning that we not only introduce these new valid inequalities for the first echelon of the production system but also extend them to take into account a multi-echelon remanufacturing system, introducing valid inequalities for each echelon of the multi-echelon system.*

4.5 Single-echelon $(\ell, k, \mathcal{U})$ inequalities

We now seek to strengthen the single-echelon $(k, \mathcal{U})$ inequalities investigated in Chapter 3 by considering the limited quantity of returned products available at each time period in the system. The resulting $(\ell, k, \mathcal{U})$ inequalities are defined as follows:

Proposition 7

Let $0 \leq \ell \leq k \leq T$ *be two periods of the planning horizon.*
Let $\mathcal{U} \subseteq \{k+1, ..., T\}$ *be a subset of periods and* $t^* = \max\{\tau : \tau \in \mathcal{U}\}$ *be the last period belonging to* $\mathcal{U}$.

The following inequalities are valid for Problem (4.12)-(4.25) *for each item* $i \in \mathcal{I}_r$:

$$S_0^\ell \hat{\delta}_i^{\ell,t^*} + \varpi_i^{-1} E_i^k + \sum_{k < t \leq t^*} \phi_i^t Y_0^t \geq \sum_{t \in \mathcal{U}} (d^t - L^t) \qquad \forall i \in \mathcal{I}_r \qquad (4.43)$$

$$S_0^\ell \hat{\delta}_i^{\ell,t^*} + (\varpi_i^{-1} E_i^\ell - E_{2I+I}^\ell) + E_{2I+1}^k + \sum_{k < t \leq t^*} \phi_i^t Y_{I+1}^t \geq \sum_{t \in \mathcal{U}} (d^t - L^t) \qquad \forall i \in \mathcal{I}_r \qquad (4.44)$$

with

$$\phi_i^t = \min \left\{ \sum_{\ell < \nu \leq t} (r^\nu \hat{\delta}_i^{\nu,t}, \sum_{\nu \in \mathcal{U} : t \leq \nu} d^\nu \right\} \qquad (4.45)$$

Moreover, the following inequalities are valid for Problem (4.12)-(4.25) *for any refurbishing process* $p \in \{1, ..., I\}$:

$$S_0^\ell \hat{\delta}_p^{\ell,t^*} + \varpi_p^{-1} (E_p^\ell - E_{p+I}^\ell) + \varpi_p^{-1} E_{p+I}^k + \sum_{k < t \leq t^*} \phi_p^t Y_p^t \geq \sum_{t \in \mathcal{U}} (d^t - L^t) \quad \forall p \in \{1, ...I\} \qquad (4.46)$$

with

$$\phi_p^t = \min \left\{ \sum_{\ell < \nu \leq t} (r^\nu \hat{\delta}_p^{\nu,t}, \sum_{\nu \in \mathcal{U} : t \leq \nu} d^\nu \right\} \qquad (4.47)$$

Proof: Let (X, Y, S, E, L, Q) be a feasible solution of Problem (4.12)-(4.25). We show that this solution complies with inequalities (4.43) for any pair of periods (ℓ, k), any subset $\mathcal{U} \subset \{k+1, ..., T\}$ and any recoverable item $i \in \mathcal{I}_r$.

Let $\tau \in [k+1, t^*]$ be the first production period in which $\phi_i^\tau = \sum_{\nu \in \mathcal{U}: \tau \leq \nu} d^\nu$. By convention, $\tau = t^* + 1$ if there is no such period.

We have $\phi_i^\tau Y_0^\tau = \sum_{t \in \mathcal{U}: \tau \leq t} d^t \geq \sum_{t \in \mathcal{U}: \tau \leq t} (d^t - L^t)$.

We consider two cases.

- Case 1: there is no production on $p = 0$ over interval $[k+1; \tau - 1]$

  In this case, we have $Y_0^t = 0$ and $X_0^t = 0$ for all periods t in $[k+1, \tau-1]$. As no disassembly occurs over $[k+1, \tau-1]$, all the recoverable items needed to satisfy the demand over this time interval, and in particular needed to satisfy $\sum_{t \in \mathcal{U}; t \leq \tau-1} (d^t - L^t)$, should already have been disassembled previously and be in stock at the end of period k. This gives $\varpi_i^{-1} E_i^k \geq \sum_{t \in \mathcal{U}: t \leq \tau-1} (d^t - L^t)$.

  We thus have:

$$\begin{aligned} S_0^\ell \hat{\delta}_i^{\ell,t^*} + \varpi_i^{-1} E_i^k + \sum_{k < t \leq t^*} \phi_i^t Y_0^t &\geq \varpi_i^{-1} E_i^k + \phi_i^\tau Y_0^\tau \\ &\geq \sum_{t \in \mathcal{U}: t \leq \tau-1} (d^t - L^t) + \sum_{t \in \mathcal{U}: t \geq \tau} (d^t - L^t) \\ &\geq \sum_{t \in \mathcal{U}} (d^t - L^t) \end{aligned}$$

  Inequality (4.43) is thus valid in this case.

- Case 2: there is at least one production period on $p = 0$ over interval $[k+1; \tau - 1]$

  Let θ be the last period of production on $p = 0$ over interval $[k+1; \tau - 1]$. By definition of θ, we have: $\phi_i^\theta = \sum_{\ell < \nu \leq \theta} (r^\nu \hat{\delta}_i^{\nu,\theta})$.

  By summing up the inventory balance constraints (4.15) over periods $k+1, \ldots, \tau-1$ and using the fact that variables E_i^k and $Q_i^t, \forall t = k+1, \ldots, \tau-1$, are non-negative, we have:

$$\varpi_i^{-1} E_i^k + \sum_{t=k+1}^{\tau-1} \delta_i^t X_0^t \geq \sum_{t=k+1}^{\tau-1} (d_t - L_t) \tag{4.48}$$

  By definition of τ, θ and ℓ, we have:

$$\sum_{t=k+1}^{\tau-1} \delta_i^t X_0^t = \sum_{t=k+1}^{\theta} \delta_i^t X_0^t \leq \sum_{t=\ell+1}^{\theta} \delta_i^t X_0^t \tag{4.49}$$

  This gives:

$$\varpi_i^{-1} E_i^k + \sum_{t=\ell+1}^{\theta} \delta_i^t X_0^t \geq \sum_{t=k+1}^{\tau-1} (d_t - L_t) \geq \varpi_i^{-1} E_i^k + \sum_{t=k+1}^{\tau-1} \delta_i^t X_0^t \geq \sum_{t \in \mathcal{U}: t \leq \tau-1} (d_t - L_t) \tag{4.50}$$

  We now compute an upper bound of $\sum_{t=\ell+1}^{\theta} \delta_i^t X_0^t$. This one is obtained by first computing the linear combination $\sum_{t=\ell+1}^{\theta} \left(\hat{\delta}_i^{t,\theta}\right) \times (4.14)_t$. This gives:

$$\sum_{t=\ell+1}^{\theta} \left(\hat{\delta}_i^{t,\theta}\right) S_0^t = \sum_{t=\ell+1}^{\theta} \left(\hat{\delta}_i^{t,\theta}\right) \left[S_0^{t-1} + r^t - X_0^t - Q_0^t\right] \tag{4.51}$$

  By the non-negativity of variables Q_0^t and S_0^t and the fact that $\hat{\delta}_i^{t,\theta} \geq \hat{\delta}_i^{t+1,\theta}$, we have:

$$\sum_{t=\ell+1}^{\theta} \delta_i^t X_0^t \leq \sum_{t=\ell+1}^{\theta} \hat{\delta}_i^{t,\theta} X_0^t \tag{4.52}$$

$$\leq \sum_{t=\ell+1}^{\theta} \left(\hat{\delta}_i^{t,\theta}\right) S_0^{t-1} - \sum_{t=\ell+1}^{\theta} \left(\hat{\delta}_i^{t,\theta}\right) S_0^t + \sum_{t=\ell+1}^{\theta} \left(\hat{\delta}_i^{t,\theta}\right) r^t \tag{4.53}$$

$$\leq \left(\hat{\delta}_i^{\ell,\theta}\right) S_0^\ell + \sum_{t=\ell+1}^{\theta} \left(\hat{\delta}_i^{t,\theta}\right) r^t \tag{4.54}$$

$$\leq \left(\hat{\delta}_i^{\ell,t^*}\right) S_0^\ell + \phi_i^\theta Y_0^\theta \tag{4.55}$$

Replacing $\sum_{t=\ell}^{\theta} \delta_i^t X_0^t$ in Inequalities (4.50) by its upper bound provided by (4.55), we have:

$$\varpi_i^{-1} E_i^k + \left(\hat{\delta}_i^{\ell,t^*}\right) S_0^\ell + \phi_i^\theta Y_0^\theta \geq \sum_{t \in \mathcal{U}: t \leq \tau - 1} (d_t - L_t) \tag{4.56}$$

Finally, we have:

$$\begin{aligned} S_0^\ell \hat{\delta}_i^{\ell,t^*} + \varpi_i^{-1} E_i^k + \sum_{k < t \leq t^*} \phi_i^t Y_0^t &\geq S_0^\ell \hat{\delta}_i^{\ell,t^*} + \varpi_i^{-1} E_i^k + \phi_i^\theta Y_0^\theta + \phi_i^\tau Y_0^\tau \\ &\geq \sum_{t \in \mathcal{U}: t \leq \tau - 1} (d_t - L_t) + \sum_{t \in \mathcal{U}: t \geq \tau} (d_t - L_t) \\ &\geq \sum_{t \in \mathcal{U}} (d_t - L_t) \end{aligned}$$

This concludes the proof of validity for Inequality (4.43). The same arguments can be used to prove the validity of Inequalities (4.46) and (4.44). □

It is worth mentioning that the $(k,\mathcal{U})$ inequalities used in Chapter 3 to strengthen the formulation (4.12)-(4.25) can be seen as a particular case of the more general family of $(\ell,k,\mathcal{U})$ inequalities (4.43), (4.44) and (4.46). Namely, by setting ℓ to 0 and by computing the value of ϕ^t without taking the returns into account (i.e. by setting ϕ^t to $\sum_{\nu \in \mathcal{U}: t \leq \nu} d^\nu$), each $(\ell,k,\mathcal{U})$ inequality (4.43),(4.44) and (4.46) becomes a $(k,\mathcal{U})$ inequality.

Proposition 8

The linear relaxation of formulation (4.12)-(4.25) *strengthened by valid inequalities* (4.43),(4.44) *and* (4.46) *is at least as tight as the linear relaxation strengthened by the* $(k,\mathcal{U})$ *valid inequalities used in Chapter 3.*

Proof: Let χ_{LP} be the linear relaxation of polyhedron given by inequalities (4.12)-(4.25), (4.43),(4.44), (4.46) and $\tilde{\chi}_{LP}$ be the linear relaxation of polyhedron given by inequalities (4.12)-(4.25) and the $(k,\mathcal{U})$ inequalities. As any $(k,\mathcal{U})$ inequality is a valid inequality (4.43),(4.44), (4.46) with $\phi^t = \sum_{\nu \in \mathcal{U}: t \leq \nu} d^\nu$ and $\ell = 0$, we have $\chi_{LP} \subseteq \tilde{\chi}_{LP}$. □

The main implication of Proposition 8 is that the lower bound obtained by strengthening the formulation (4.12)-(4.25) with the $(\ell,k,\mathcal{U})$ inequalities is at least as tight as the lower bound obtained while using the single-echelon $(k,\mathcal{U})$ inequalities.

Separation algorithm

We now briefly discuss the resolution of the separation problem for the $(\ell, k, \mathcal{U})$ valid inequalities. Recall that this problem consists in finding an inequality (4.43),(4.44) and (4.46) violated by a a given solution $(\tilde{X}, \tilde{Y}, \tilde{S}, \tilde{E}, \tilde{L}, \tilde{Q})$ of the linear relaxation of Problem (4.12)-(4.25) or prove that no such inequality exists. In the present case, in order to find the most violated inequality among e.g. inequalities (4.43), we should find, for each period $k = 1, ..., T$, the set $\mathcal{U}$ of time periods and the period ℓ that maximize the difference between the right-hand and the left-hand side of the inequality. This computation does not seem easy, in particular because the value of each coefficient ϕ_i^t simultaneously depends on $\mathcal{U}$ and ℓ.

We thus consider a heuristic separation algorithm in our computational experiments, which can be summarized as follows. For a given process p:

Algorithm 6: Separation algorithm for $(\ell, k, \mathcal{U})$

```
for k ∈ 𝒯 do
    for t = k+1, ..., T do
        if d^t(1 − Σ_{τ=k+1}^{t} Ỹ_p^τ) − L̃^t > 0 then
            t ∈ 𝒰
        end
    end
    for ℓ = 0, ..., k do
        compute the value of each coefficient φ_i^t = min{ Σ_{ℓ<ν≤t} r^ν δ̂_i^{ν,t}, Σ_{ν∈𝒰:t≤ν} d^ν }
        compute the left-hand side of the inequality (4.43) (resp. (4.44) and (4.46))
        -Set ℓ to the period index which minimizes this left-hand side value.
    end
end
```

Finally, we can combine Inequalities (4.43),(4.44) and (4.46) with the two-echelon $(k, \mathcal{U})$ inequalities introduced in Section 4.3 to obtain two-echelon $(\ell, k, \mathcal{U})$ inequalities.

Corollary 2

Let $1 \leq k \leq T$ be a period in the planning horizon and $\mathcal{U} \subseteq \{k+1, ..., T\}$ be a subset of periods. Let $t^* = \max\{\ell \in \mathcal{U}\}$ denote the last period belonging to $\mathcal{U}$. We partition $\mathcal{U}$ into 2 subsets: $\mathcal{U}_1 \subseteq \mathcal{U}$ and $\mathcal{U}_0 = \mathcal{U} \setminus \mathcal{U}_1$.

The following inequalities are valid for Problem (4.12)-(4.25) for any refurbishing process $p \in \{1, ..., I\}$:

$$S_0^\ell \hat{\delta}_p^{\ell,t^*} + \varpi_p^{-1} E_p^k + \sum_{k<t\leq t^*} \phi_p^t(\mathcal{U}_0) Y_0^t + \sum_{k<t\leq t^*} \phi_p^t(\mathcal{U}_1) Y_p^t \geq \sum_{t\in\mathcal{U}} (d^t - L^t) \tag{4.57}$$

$$S_0^\ell \hat{\delta}_p^{\ell,t^*} + \varpi_p^{-1}(E_p^\ell - E_{p+I}^\ell) + \varpi_p^{-1} E_{p+I}^k + \sum_{k<t\leq t^*} \psi_p^t(\mathcal{U}_0) Y_p^t + \sum_{k<t\leq t^*} \phi_p^t(\mathcal{U}_1) Y_{I+1}^t \geq \sum_{t\in\mathcal{U}} (d^t - L^t) \tag{4.58}$$

such that

$$\phi_p^t(\tilde{\mathcal{U}}) = \min\left\{ \sum_{\ell<\nu\leq t} (r^\nu \hat{\delta}_p^{\nu,t}), \sum_{\nu\in(\tilde{\mathcal{U}}):t\leq\nu} d^\nu \right\}$$

Moreover, for each item $i \in \mathcal{I}_r$, the following inequalities are valid for Problem (4.12)-(4.25)*:*

$$S_0^\ell \hat{\delta}_i^{\ell,t^*} + \varpi_i^{-1} E_i^k + \sum_{k<t\leq t^*} \phi_i^t(\mathcal{U}_0) Y_0^t + \sum_{k<t\leq t^*} \phi_i^t(\mathcal{U}_1) Y_{I+1}^t \geq \sum_{t\in\mathcal{U}} (d^t - L^t) \tag{4.59}$$

such that

$$\phi_i^t(\tilde{\mathcal{U}}) = \min\Big\{ \sum_{\ell<\nu\leq t} (r^\nu \hat{\delta}_i^{\nu,t}), \sum_{\nu\in\tilde{\mathcal{U}}: t\leq\nu} d^\nu \Big\}$$

4.6 Summary of valid inequalities: class, separation problem and complexity

Before going on with the discussion on our computational results, we provide a brief summary of the proposed valid inequalities and of their related features in Table 4.1. Column "Class" indicates whether the number of valid inequalities in the family grows linearly or exponentially fast with respect to the number of periods in the planning horizon. Column "Complexity" reports the difficulty of the corresponding separation problem. Then, Column "Separation alg." indicates whether the proposed separation problem is solved by an exact or a heuristic algorithm in our numerical experiments. Finally, Column "Time" reports the asymptotic time complexity of the implemented separation algorithm.

Note that all these features have a significant impact on the numerical tractability and efficiency of a class of valid inequalities. Namely, the simplest way of using valid inequalities to strengthen a MILP formulation consists in directly adding all of them in the formulation. When the number of inequalities stays low (in particular when it grows linearly fast with the problem size), this option may be numerically efficient. But when the number of inequalities grows exponentially fast with the problem size, it is not possible to add all of them a priori to the formulation as it would to lead to a prohibitively large mixed-integer linear program. In this second case, it is possible to add part of the known valid inequalities through a cutting-plane generation procedure, which iteratively removes non-integral points of the incumbent formulation (see [79], pag. 190-191). At each iteration of this procedure, we need to find the inequality that is the most violated by the incumbent solution (if it exists): this consists in solving the separation problem. It is thus important to study its complexity, to determine whether it is polynomially solvable or NP-hard and to devise a separation algorithm which may be exact or heuristic, depending on the situation.

Table 4.1: Class and complexity

Inequalities	Class	Complexity	Separation alg.	Time
Single-echelon $(k,\mathcal{U})$	Exponential	Polynomial	Exact	$\mathcal{O}(T^2)$
Two-echelon $(k,\mathcal{U})$	Exponential	Polynomial	Exact	$\mathcal{O}(T^2)$
Single-echelon $(\ell,\mathcal{U})$	Exponential	Polynomial	Exact	$\mathcal{O}(T^2)$
Single-echelon $(\ell,k,\mathcal{U})$	Exponential	unknown	Heuristic	$\mathcal{O}(T^2)$
Two-echelon $(\ell,k,\mathcal{U})$	Exponential	unknown	Heuristic	$\mathcal{O}(T^2)$

4.7 Computational experiments

In this section, we focus on assessing the performance of the proposed valid inequalities. This assessment is carried out in two steps. In the first step, we focus on measuring the strengthening of the linear relaxation of Problem (4.12)-(4.25) obtained through a cutting-plane generation algorithm based on these valid inequalities. This enables us to identify three promising settings for the cutting plane generation. In a second step, we embed the three selected cutting-plane generation algorithms in customized branch-and-cut algorithms and compare the performance of these customized algorithms with the one of the generic branch-and-cut algorithm embedded in the mathematical programming solver CPLEX 12.8. During these experiments, all the linear programs and mixed-integer linear programs 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 3600 seconds.

We first describe the scheme used to randomly generate the input data of the problem. We then provide computational results showing the effectiveness of the proposed valid inequalities at solving the problem.

4.7.1 Instance Generation

We considered three sets of instances: Set 1 instances involve $T = 20$ and $I = 5$ parts, Set 2 instances involve $T = 25$ periods and $I = 10$ parts and Set 3 instances involve $T = 35$ periods and $I = 10$ parts. Within each set, the instances were randomly generated by adapting the procedure presented in [45]. More precisely, we considered four values of the setup-holding cost ratio $f/h \in \{600, 1200, 1800, 2400\}$, two values for the production-holding cost ratio $g/h \in \{2, 4\}$ and three values of the returns-demand quantity ratio $r/d \in \{1, 2, 3\}$. For each set and each possible combination of f/h, g/h, r/d, ten random instances were generated, resulting in a total of 720 instances.

For each instance, the value of each problem parameter was set as follows.

- Demand d^t was uniformly distributed in the interval $[0, 100]$ and the returns quantity r^t was uniformly distributed in the interval $[0.8(r/d)\bar{d}, 1.2(r/d)\bar{d}]$, where $\bar{d} = \frac{\sum d^t}{T}$ is the average demand per period.

- The proportion of recoverable parts π_i^t was uniformly distributed in the interval $[0.4, 0.6]$.

- The bill-of-materials coefficients $\varpi_i = \varpi_{i+I}, i = 1, \ldots, I$, were randomly generated following a discrete uniform distribution over $[1; 6]$ and we set $\varpi_0 = \varpi_{2I+1} = 1$.

- The holding cost h_0^t for the returned product $i = 0$ was fixed to 1. The holding cost h_i^t for each recoverable item $i \in \mathcal{I}_r$ was randomly generated following a discrete uniform distribution over interval $[2, 7]$. Similarly, the holding cost h_i^t for each serviceable item $i \in \mathcal{I}_s$ was randomly generated following a discrete uniform distribution over interval $[7, 12]$. Finally, in order to ensure non negative echelon costs, we set the value of the inventory holding cost for the remanufactured product, h_{2I+1}^t, to $\sum_{i=1}^{I} \varpi_i h_{I+i}^t + \epsilon$, where ϵ follows a discrete uniform distribution over interval $[80, 100]$.

- The production cost g^t was uniformly distributed in the interval $[0.8(g/h)\bar{h}, 1.2(g/h)\bar{h}]$, where $\bar{h} = \frac{\sum h_{2I+1}^t}{T}$ is the average holding cost.

- The set-up cost f^t was uniformly distributed in the interval $[0.8(f/h)\bar{h}, 1.2(f/h)\bar{h}]$.

- Discarding costs were set to $q_i^t = 0.8\bar{h}_i^t$, where $\bar{h}_i^t = \frac{1}{T}\sum_{\kappa=t}^{T} h_i^\kappa$
- The unit penalty cost for lost sales, l^n, was fixed to 10000 per unit.

We now discuss the results of the extensive computational experiments we conducted in order to assess the performance of the valid inequalities proposed in this chapter.

4.7.2 Cutting-plane generation algorithms: preliminary assessment on small instances

In the first step, we focus on measuring the strengthening of the linear relaxation of Problem (4.12)-(4.25) obtained when adding a subset of the valid inequalities discussed in Sections 4.3 to 4.5 through a cutting-plane generation algorithm. More precisely, we start by solving the linear relaxation of Problem (4.12)-(4.25). We then iteratively add to the formulation violated valid inequalities from one or two of the classes discussed above until no more violated valid inequalities can be found. When the cutting-plane generation stops, we measure the linear gap $\texttt{Gap}_{LP}$, i.e. the relative difference between the lower bound provided by the (strengthened) linear relaxation and the optimal value, and `#Cuts`, the total number of cuts generated by the algorithm.

We consider several settings for the cutting plane algorithm differing from one another by the classes of valid inequalities generated at each iteration. We refer to each of these settings as follows:

- CPX: no cutting plane is generated.
- $(k,\mathcal{U})$: we generate cutting planes based on the single-echelon $(k,\mathcal{U})$ inequalities proposed in [66].
- $2E$-$(k,\mathcal{U})$: we generate cutting planes based on the two-echelon $(k,\mathcal{U})$ inequalities (4.26)-(4.28)
- $(\ell,k,\mathcal{U})$: we generate cutting planes based on the single-echelon $(\ell,k,\mathcal{U})$ inequalities with returns (4.43),(4.44) and (4.46).
- $2E$-$(\ell,k,\mathcal{U})$: we generate cutting planes based on the two-echelon $(\ell,k,\mathcal{U})$ inequalities with returns (4.57)-(4.59).
- $(k,\mathcal{U}) + (\ell,\mathcal{U})$: we generate cutting planes based on the single-echelon $(k,\mathcal{U})$ inequalities proposed in [66] and the single-echelon $(\ell,\mathcal{U})$-returns inequalities.
- $2E$-$(k,\mathcal{U})$+$(\ell,\mathcal{U})$: we generate cutting planes based on the two-echelon $(k,\mathcal{U})$ inequalities (4.26)-(4.28) and the single-echelon $(\ell,\mathcal{U})$-returns inequalities.
- $(\ell,k,\mathcal{U}) + (\ell,\mathcal{U})$: we generate cutting planes based on the single-echelon $(\ell,k,\mathcal{U})$ inequalities with returns (4.43),(4.44), (4.46) and the single-echelon $(\ell,\mathcal{U})$-returns inequalities.
- $2E$-$(\ell,k,\mathcal{U}) + (\ell,\mathcal{U})$: we generate cutting planes based on the two-echelon $(\ell,k,\mathcal{U})$ inequalities with returns (4.57)-(4.57) and the single-echelon $(\ell,\mathcal{U})$-returns inequalities.

Note that we do not report the individual performance of the single-echelon $(\ell,\mathcal{U})$-returns inequalities. Namely, our preliminary results showed that inequalities (4.33), (4.34) and (4.41), when used alone, perform rather poorly at strengthening the linear relaxation of the problem. However, their performance improves when they are used in combination with inequalities. We thus report in Table 4.2 the results obtained with each class of $(k,\mathcal{U})$ inequalities, when used alone, and in Table 4.3 the results obtained with each class of $(k,\mathcal{U})$ inequalities, when used in combination with single-echelon $(\ell,\mathcal{U})$-returns inequalities. These results were obtained for the small-size instances of Set 1.

Results in Table 4.2 first show that the formulation improvement obtained when using the two-echelon $(k,\mathcal{U})$ inequalities instead of the single-echelon $(k,\mathcal{U})$ inequalities is limited. The average integrality gap is namely slightly reduced from 14.22% with setting $(k,\mathcal{U})$ to 14.08% with setting $2E$-$(k,\mathcal{U})$. Moreover, this gap improvement is obtained at the expense of a significant number of cutting planes added to the formulation: `#Cuts` thus increases from 166 with setting $(k,\mathcal{U})$ to 376 with setting $2E$-$(k,\mathcal{U})$. This suggests that the performance of the proposed two-echelon $(k,\mathcal{U})$ inequalities at solving large-size instances may be limited: the increase in the number of constraints coming from the generation of two-echelon $(k,\mathcal{U})$ inequalities will not be offset by the formulation strengthening so that the overall computation time needed to solve the problem with a branch-and-cut algorithm may deteriorate.

In contrast, the results displayed in Table 4.2 show that the single-echelon $(k,\mathcal{U})$ inequalities with returns perform significantly better than the initial single-echelon $(k,\mathcal{U})$ inequalities. The average integrality gap is namely significantly reduced from from 14.22% with setting $(k,\mathcal{U})$ to 9.36% with setting $(k,\mathcal{U}) + (\ell,\mathcal{U})$. Although this formulation strengthening is obtained through an increase in the number of generated cuts (from 166 to 236), the resulting gap improvement may be sufficient to observe a positive impact on the overall computation time needed to solve the problem to optimality. Finally, we note that, similar to the the two-echelon $(k,\mathcal{U})$ inequalities, the extension of the single-echelon $(k,\mathcal{U})$ inequalities with returns to the two-echelon $(k,\mathcal{U})$ inequalities with returns does not seem very promising.

Table 4.2: Performance of each cutting-plane generation setting over Set 1 instances

Instances		CPX		$(k,\mathcal{U})$		$2E$-$(k,\mathcal{U})$		$(\ell,k,\mathcal{U})$		$2E$-$(\ell,k,\mathcal{U})$	
r/d	g/h	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$
1	2	7,95	0	5,96	185	5,92	395	3,66	240	3,58	337
	4	5,44	0	3,91	205	3,89	396	2,53	250	2,47	326
Average		6,70	0	4,94	195	4,91	396	3,09	245	3,03	332
2	2	38,02	0	18,02	189	17,88	439	10,72	277	10,46	384
	4	35,89	0	17,64	206	17,53	450	11,12	286	10,92	367
Average		36,96	0	17,83	197	17,70	445	10,92	281	10,69	375
3	2	40,35	0	20,99	106	20,73	280	14,31	183	14,08	261
	4	34,41	0	18,77	108	18,53	293	13,86	179	13,64	270
Average		37,38	0	19,88	107	19,63	287	14,08	181	13,86	265
Total Average		27,01	0	14,22	166	14,08	376	9,36	236	9,19	324

Table 4.3: Performance of each cutting-plane generation setting over Set 1 instances

Instances		$(\ell,\mathcal{U})$		$(k,\mathcal{U}) + (\ell,\mathcal{U})$		$2E$-$(k,\mathcal{U}) + (\ell,\mathcal{U})$		$(\ell,k,\mathcal{U}) + (\ell,\mathcal{U})$		$2E$-$(\ell,k,\mathcal{U}) + (\ell,\mathcal{U})$	
r/d	g/h	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$	Gap_{LP}	$\#Cuts$
1	2	5,53	52	2,87	323	2,84	618	2,87	321	2,83	447
	4	3,73	63	1,90	348	1,88	610	1,90	341	1,87	444
Average		4,63	57	2,39	336	2,36	614	2,39	331	2,35	445
2	2	22,95	401	7,76	526	7,65	894	7,74	522	7,61	676
	4	20,80	521	7,91	573	7,82	905	7,89	546	7,76	664
Average		21,87	461	7,83	550	7,74	900	7,82	534	7,69	670
3	2	31.99	357	11,33	576	11,17	912	11,30	518	11,14	699
	4	28.69	376	11,42	575	11,26	942	11,41	504	11,25	706
Average		30.34	366	11,37	576	11,21	927	11,35	511	11,19	703
Total Average		18,95	295	7,20	487	7,10	814	7,19	459	7,08	606

Regarding the single-echelon $(\ell,\mathcal{U})$-return inequalities, we note that a significant reduction of the integrality gap from 14.22% (setting $(k,\mathcal{U})$) to 7.20% (setting $(k,\mathcal{U}) + (\ell,\mathcal{U})$) can be obtained

when using them in combination with the single-echelon $(k,\mathcal{U})$ inequalities. This shows the potential usefulness and complementarity of these two classes of inequalities. A reduction of the integrality gap can also be observed when using the single-echelon $(\ell,\mathcal{U})$-return inequalities in combination with the single-echelon $(\ell,\mathcal{U})$ inequalities with returns. However, this gap reduction (from 9.36% with setting $(\ell,k,\mathcal{U})$ to 7.19% with setting $(\ell,k,\mathcal{U})+(\ell,\mathcal{U})$) may not be large enough to compensate for the strong increase in the number of generated cutting-planes (from 236 to 459).

In what follows, we will thus focus on further evaluating the performance of the studied valid inequalities using the cutting-plane generation settings which appeared to be the most promising during our preliminary experiments, namely CPX, $(k,\mathcal{U})$, $(k,\mathcal{U})+(\ell,\mathcal{U})$ and $(\ell,k,\mathcal{U})$.

4.7.3 Branch-and-cut algorithms: assessment on medium-size instances

We now assess the performance of the proposed valid inequalities by using them in customized branch-and-cut algorithms aiming at solving the mixed-integer linear program (4.12)-(4.25) to optimality. As mentioned above, we considered the four settings for the cutting-plane generation algorithm which seemed to be the most computationally promising in our preliminary tests: CPX, $(k,\mathcal{U})$, $(k,\mathcal{U})+(\ell,\mathcal{U})$ and $(\ell,k,\mathcal{U})$. Note that these cutting-plane generation algorithms are run within the UserConstraints callback routine of CPLEX 12.8.

Table 4.4: Performance of CPLEX and branch-and-cut methods over Set 2 instances.

r/d	f/h	Method	Gap_{LP}	Gap^{root}_{MIP}	Gap_{MIP}	$\#Cuts$	$\#Nodes$	Total Time
1	2	CPX	8.26	4.10	0.06	422	415,616	898.00
		$(k,\mathcal{U})$	6.08	3.66	0.05	855	180,687	1,011.79
		$(k,\mathcal{U})+(\ell,\mathcal{U})$	3.59	3.37	0.04	1,059	142,688	878.31
		$(\ell,k,\mathcal{U})$	4.17	3.55	0.03	891	238,910	792.64
	4	CPX	5.38	2.49	0.06	436	768,171	1,360.48
		$(k,\mathcal{U})$	3.95	2.33	0.05	816	437,563	1,088.78
		$(k,\mathcal{U})+(\ell,\mathcal{U})$	2.38	2.18	0.04	1,065	464,285	1,008.08
		$(\ell,k,\mathcal{U})$	2.84	2.22	0.04	871	468,827	1,056.29
2	2	CPX	40.86	11.28	0.97	778	537,370	2,570.62
		$(\ell,k,\mathcal{U})$	19.36	9.17	0.46	949	354,801	1,883.71
		$(k,\mathcal{U})+(\ell,\mathcal{U})$	8.67	8.38	0.33	1,517	234,589	2,028.58
		$(\ell,k,\mathcal{U})$	11.88	8.68	0.22	1,106	345,226	1,558.09
	4	CPX	37.79	10.71	2.29	843	629,586	3,280.23
		$(k,\mathcal{U})$	18.39	9.25	1.06	1,030	953,407	2,856.09
		$(k,\mathcal{U})+(\ell,\mathcal{U})$	8.74	8.19	0.85	1,652	449,304	2,885.18
		$(\ell,k,\mathcal{U})$	11.89	8.48	0.94	1,205	637,218	2,883.50
3	2	CPX	41.86	13.47	0.04	502	210,487	545.15
		$(k,\mathcal{U})$	20.60	9.68	0.02	705	140,289	296.65
		$(k,\mathcal{U})+(\ell,\mathcal{U})$	11.07	8.80	0.02	1,658	113,575	541.58
		$(\ell,k,\mathcal{U})$	14.16	9.11	0.02	795	119,074	250.52
	4	CPX	38.29	12.44	0.09	539	655,390	817.93
		$(k,\mathcal{U})$	20.81	9.71	0.02	708	368,794	458.41
		$(k,\mathcal{U})+(\ell,\mathcal{U})$	12.65	8.93	0.05	1,713	152,699	736.40
		$(\ell,k,\mathcal{U})$	15.36	8.99	0.06	801	338,830	522.70

Tables 4.4 and 4.5 report the results obtained on the medium-size instances of Sets 2 and 3. Column `Method` indicates the setting of the cutting-plane algorithm used to solve the instances. Similar to what was displayed in Tables 4.2-4.3, Column $\texttt{Gap}_{LP}$ reports the gap between the objective

Table 4.5: Performance of CPLEX and branch-and-cut methods over Set 3 instances.

r/d	g/h	Method	Gap_{LP}	Gap^{root}_{MIP}	Gap_{MIP}	$\#Cuts$	$\#Nodes$	Total Time
1	2	CPX	7.48	3.65	0.25	579	393,547	2,544.42
		$(k,\mathcal{U})$	5.09	3.08	0.25	1,351	114,803	2,300.22
		$(k,\mathcal{U}) + (\ell,\mathcal{U})$	2.96	2.81	0.17	1,771	102,314	2,128.85
		$(\ell,k,\mathcal{U})$	3.49	2.95	0.11	1,369	144,516	1,764.24
	4	CPX	4.96	2.33	0.19	646	545,143	2,930.72
		$(k,\mathcal{U})$	3.31	2.00	0.11	1,398	250,394	2,529.28
		$(k,\mathcal{U}) + (\ell,\mathcal{U})$	1.97	1.83	0.09	1,902	185,079	2,401.90
		$(\ell,k,\mathcal{U})$	2.40	1.91	0.10	1384	233,612	2,319.77
2	2	CPX	44.73	12.89	6.76	888	170,434	3,599.07
		$(k,\mathcal{U})$	20.10	10.00	4.34	1,387	130,088	3,508.81
		$(k,\mathcal{U}) + (\ell,\mathcal{U})$	9.61	9.35	4.68	2,415	40,552	3,599.46
		$(\ell,k,\mathcal{U})$	12.85	9.54	3.77	1,590	125,340	3,362.24
	4	CPX	43.15	12.54	7.29	960	209,853	3,599.07
		$(k,\mathcal{U})$	20.02	10.40	5.03	1,442	259,433	3,599.54
		$(k,\mathcal{U}) + (\ell,\mathcal{U})$	9.56	9.11	4.39	2,600	88,832	3,599.53
		$(\ell,k,\mathcal{U})$	13.14	9.49	4.61	1,709	149,221	3,599.44
3	2	CPX	43.25	14.11	2.67	799	274,369	3,044.18
		$(k,\mathcal{U})$	18.97	9.48	0.55	1,102	177,695	1,855.28
		$(k,\mathcal{U}) + (\ell,\mathcal{U})$	10.52	8.75	1.32	2,950	40,543	2,423.61
		$(\ell,k,\mathcal{U})$	13.38	9.11	0.64	1,184	178,227	1,614.35
	4	CPX	37.56	11.83	1.60	773	331,016	2,802.69
		$(k,\mathcal{U})$	16.82	8.30	0.19	1,062	154,567	1,035.84
		$(k,\mathcal{U}) + (\ell,\mathcal{U})$	10.26	7.62	0.54	3,019	52,068	1,844.06
		$(\ell,k,\mathcal{U})$	12.67	7.86	0.20	1,149	103,955	1,029.84

value of the linear relaxation strengthened by the corresponding cutting-plane generation algorithm and the best feasible solution found through the branch-and-bound search. For the CPX setting, it reports the initial integrality gap of Formulation (4.12)-(4.25). Column $\texttt{Gap}^{root}_{MIP}$ reports the gap between the lower bound obtained at the root node of the search tree (after the generation by CPLEX 12.8 solver of its generic cuts) and the best feasible solution found through the branch-and-bound search. Column $\texttt{Gap}_{MIP}$ provides the gap between the best lower bound and the best feasible solution found throughout the branch-and-bound search. The average number of total cuts generated during the branch-and-bound tree is reported in Column `#Cuts` and the average number of total nodes explored is reported in Column `#Nodes`. Column `Total Time` reports the average of total CPU time.

We first observe from these results that the customized branch-and-cut algorithms based on the $(k,\mathcal{U})$, $(k,\mathcal{U}) + (\ell,\mathcal{U})$ or $(\ell,k,\mathcal{U})$ settings of the cutting-plane generation algorithm significantly outperform the generic branch-and-cut algorithm embedded in CPLEX 12.8, providing solutions of better quality within shorter computation times. Specifically, the total computation time `Total Time` is reduced on average by 20% when using the $(k,\mathcal{U})$ setting, by 14% when using the $(k,\mathcal{U}) + (\ell,\mathcal{U})$ setting and by 26% when using the $(\ell,k,\mathcal{U})$ setting.

Regarding the relative performance of the $(k,\mathcal{U})$, $(k,\mathcal{U}) + (\ell,\mathcal{U})$ and $(\ell,k,\mathcal{U})$ settings, we note that the $(k,\mathcal{U}) + (\ell,\mathcal{U})$ and $(\ell,k,\mathcal{U})$ settings outperform the $(k,\mathcal{U})$ setting when the value of the demand-returns ratio is small, ie. when $r/d \in \{1,2\}$. Thus, over the 240 instances corresponding to a value of r/d equal to 1, the total average computation time is reduced from 1732s when using the $(k,\mathcal{U})$ setting to 1604s when using the $(k,\mathcal{U}) + (\ell,\mathcal{U})$ setting and 1482s when using the $(\ell,k,\mathcal{U})$ setting. Similarly, over the 240 instances corresponding to a value of r/d equal to 2, the average MIP gap is reduced from 2.72% when using the $(k,\mathcal{U})$ setting to 2.56% when using the $(k,\mathcal{U}) + (\ell,\mathcal{U})$

setting and 2.38% when using the $(\ell, k, \mathcal{U})$ setting.

We note however that the relative performance of the $(k, \mathcal{U}) + (\ell, \mathcal{U})$ and $(\ell, k, \mathcal{U})$ settings deteriorates for the instances corresponding to the largest considered value of the demand-returns ratio. Namely, when r/d is set to 3, the branch-and-cut algorithm based the $(k, \mathcal{U})$ setting provides smaller MIP gap and/or smaller computation times. This might be explained by the fact that the corresponding instances involve a large amount of returned products so that the quantity processed on a resource at a given period is not (or at least to a lesser extent) limited by the availability of the returned products. This means that the proposed refinements in the expression of the valid inequalities are less relevant in this case.

4.8 Conclusion and perspectives

In this chapter, we proposed several new classes of valid inequalities in order to strengthen the formulation of the multi-echelon lot-sizing problem with lost sales and returns introduced in Chapter 3. Our main contribution is the development of a new class of valid inequalities which take into account, at each production echelon, the limitations on the produced quantities coming from the limited availability of the returned products. The results of our computational experiments show that a branch-and-cut algorithm based on these new valid inequalities performs well as compared to the generic branch-and-cut algorithm of CPLEX 12.8 solver and to the branch-and-cut algorithm based on previously published valid inequalities and investigated in Chapter 3.

The work presented here opens several research directions. First, it would be interesting to develop an exact separation algorithm to generate the single-echelon (ℓ, k, U) inequalities with returns in the presented branch-and-cut algorithm. It may namely further improve its relative performance. Second, these inequalities have shown their usefulness to solve the problem in a deterministic setting. It might also be worth investigating whether they could also be helpful to solve its stochastic variant. This point will be partly tackled in Chapter 6 of this manuscript. Finally, on a longer term perspective, we could seek to exploit the idea underlying the expression of these inequalities to develop valid inequalities for a larger class of lot-sizing problems with returns.

Part III

Decomposition based approaches

Chapter 5

Combining polyhedral approaches and SDDiP

Part III of this thesis is devoted to the study of decomposition-based approaches to solve multi-stage stochastic lot-sizing problems. More precisely, we investigate how the recently introduced stochastic dual dynamic integer programming (SDDiP) algorithm may be useful to solve our stochastic remanufacturing planning problem. However, in view of the complexity and novelty of this algorithm, we first focus in this chapter on its application on the much simpler stochastic uncapacitated lot-sizing problem and propose a new extension of the SDDiP algorithm in order to improve its computational performance. The SDDiP algorithm relies on a full decomposition of the problem into a set of small sub-problems linked together by expected cost-to-go functions. The proposed extension aims at being more computationally efficient in the management of these expected cost-to-go functions, in particular by reducing their number and by exploiting the current knowledge on the polyhedral structure of the stochastic uncapacitated lot-sizing problem. The algorithm is based on a partial decomposition of the problem into a set of stochastic sub-problems, each one involving a subset of nodes forming a sub-tree of the initial scenario tree. We then introduce a cutting-plane generation procedure that iteratively strengthens the linear relaxation of these sub-problems and enables the generation of additional strengthened Benders' cut, which improves the convergence of the method. We carry out extensive computational experiments on randomly generated large-size instances. Our numerical results show that the proposed algorithm significantly outperforms the SDDiP algorithm at providing good-quality solutions within the computation time limit.

5.1 Introduction

The single-item deterministic uncapacitated lot-sizing problem (ULS) is a production planning problem first introduced in [100]. It considers a single type of item and aims at determining the quantity to be produced in each time period in order to meet demand over a finite discrete-time planning horizon. Producing a positive amount in a period incurs a fixed cost, called set-up cost, together with a production cost per unit produced and an inventory holding cost per unit held in stock between two consecutive periods. The objective is to build a production plan such that the customers' demand is met in each time period and the total costs, i.e. the sum of set-up, production, and inventory holding costs over the whole planning horizon, are minimized. This fundamental problem naturally appears as an embedded sub-problem in many practical production planning problems. Solving it efficiently is thus essential to develop algorithms capable of dealing with real-world problems.

As such, the deterministic ULS is known to be solvable in strongly polynomial time. A simple dynamic programming algorithm is proposed in [100]. It is based on the zero-inventory-ordering property, i.e. production is undertaken in a period only if the entering inventory level drops to zero, and runs in $\mathcal{O}(T^2)$ time, where T is the number of time periods. This time complexity was later

improved to $\mathcal{O}(T \log T)$ in [3] and [99]. We refer the reader to [21] for an updated and comprehensive survey on the single-item dynamic lot-sizing problem.

However, in many applications, assuming deterministically known input data (demand and costs) is not realistic. Examples of real-world lot-sizing problems with uncertain input parameters can be found among others in [25] for the spinning industry, [55] for a manufacturing company producing braking equipment, [41] for a chemical-petrochemical case study, [62] for a remanufacturing system, [68] for a hybrid manufacturing/remanufacturing system and [73] for humanitarian logistics.

In the present paper, we thus investigate a stochastic extension of the ULS, denoted SULS in what follows, in which the problem parameters 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 can be made progressively as more and more information on the demand and cost realizations are collected. In order to address this problem, we rely on a multi-stage stochastic integer programming approach and assume that the underlying stochastic input process has a finite probability space. The information on the evolution of the uncertain parameters can thus be represented by a discrete scenario tree.

5.1.1 Related literature

Scarf [91] found that, when all cost parameters are deterministically known and only the demand is subject to uncertainty, the optimal solution of SULS can be obtained by following a (s, S) inventory management policy. However, authors of [51] showed that a special case of SULS in which the production and inventory holding costs are stochastic, the set-up costs are set to zero and the uncertain demand can take only two possible values in each period is NP-Hard. It is thus unlikely to find polynomial algorithms in the number of time periods for the problem. Guan and Miller [47] developed a dynamic programming algorithm for solving the SULS, which is polynomial in the number of nodes of the scenario tree. However, this number increases exponentially fast with the number of time periods so that using their algorithm to solve problems with a medium to large size planning horizon might lead to numerical difficulties. Moreover, this algorithm is based on some specific properties of the optimal solutions of the SULS. As a consequence, its direct extension to more general lot-sizing problems whose optimal solutions do not display these properties is not straightforward. This is why other solution approaches based on mixed-integer linear programming or nested Benders' decomposition have also been explored. In what follows, we provide an overview of the related literature.

Using a scenario tree to represent the evolution of the uncertain parameters namely leads to the formulation of a Mixed-Integer Linear Program (MILP) which can be solved using mathematical programming solvers. Consequently, several works focused on the polyhedral study of this MILP in order to strengthen its linear relaxation and improve the computational efficiency of the branch-and-cut algorithms embedded in MILP solvers. Valid inequalities can be found in [46], [34] and [45]. In particular, authors of [45] proposed a general method for generating cutting planes for multi-stage stochastic integer programs based on combining valid inequalities previously known for the deterministic variant of the corresponding problem and applied it on the SULS. Their numerical results showed that a branch-and-cut algorithm based on these new inequalities is more effective at solving instances on medium-size scenarios than a stand-alone mathematical programming solver. Some extended formulations have also been studied. An extended formulation using variable disaggregation was proposed in [4]. More recently, Zhao and Guan [104] developed an extended formulation that provides integral solutions for the general SULS. However, the size of this formulation grows exponentially fast with the number of time periods, making its use computationally unpractical for solving instances defined on large-size scenario trees.

In general, solution approaches based on strengthening MILP formulations do not scale up well with the size of the scenario tree. They namely entail solving very large-scale (mixed-integer) linear

programs, with millions of variables and constraints, which leads to memory issues and/or prohibitive computation times in practice. Decomposition methods, such as the nested Benders' decomposition algorithm, are thus an attractive alternative to tackle instances with large-size scenario trees. In particular, the Stochastic Dual Dynamic Programming (SDDP) approach proposed in [75] has been widely used to solve large-size multi-stage stochastic linear programs. This approach relies on a dynamic programming formulation of the stochastic problem and leads to a decomposition of the overall problem into a series of small deterministic sub-problems. Each of these problems focuses on making decisions for a small subset of nodes belonging to the same scenario and the same decision stage, taking into account not only the current cost of these decisions but also their future cost which is represented by an expected cost-to-go function. In a linear setting, the expected cost-to-go functions are convex and piecewise linear and can thus be under-approximated through a set of supporting hyperplanes. The SDDP algorithm builds such an approximation by iteratively adding Benders' cuts to each sub-problem and converges to an optimal solution in a finite number of iterations.

Recently, Zou et al. [105] proposed a new extension of the SDDP algorithm, called the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm, capable of solving multi-stage stochastic integer programs in which the state variables, i.e. the variables linking the nodes to one another, are restricted to be binary. One of their main contributions was to introduce a new class of cutting planes, called Lagrangian cuts, which satisfies the validity, tightness and finiteness conditions ensuring the convergence of the algorithm to optimality. For problems in which the state variables are not binary but continuous, the authors propose to introduce auxiliary binary variables in order to make a binary approximation of the state variables. However, for large size scenario trees, this approximation might be computationally inefficient and leads to large optimality gaps as shown e.g. by the numerical results presented in [82].

5.1.2 Contributions

In the present work, we propose a new extension of the SDDiP algorithm. This extension aims at being more computationally efficient in the management of the expected cost-to-go functions involved in the problem, in particular by reducing their number and by exploiting the current knowledge on the polyhedral structure of the SULS. It relies on the following three main ideas:

- Similar to the SDDiP algorithm, we exploit a dynamic programming formulation and decompose the problem into smaller sub-problems. However, whereas the SDDiP algorithm fully decomposes the original problem into small deterministic sub-problems, we partially decompose the problem into a set of somewhat larger stochastic sub-problems, each one involving a subset of nodes forming a sub-tree of the initial scenario tree. These sub-problems are more computationally demanding to solve than the deterministic sub-problems involved in the original SDDiP algorithm. However, this computational effort might be counterbalanced by an improvement in the quality of the feasible solution found at each iteration of the algorithm. Namely, when each sub-problem covers a larger portion of the planning horizon and the number of expected cost-to-go functions for which an approximation has to be iteratively built is reduced, the feasible solution obtained at a given iteration of the algorithm will be based on a better representation of the future costs, i.e. will be less myopic, and will tend to be of better quality. This might have a positive impact on the global convergence speed of the algorithm.

- In the SDDiP algorithm proposed in [105], three classes of cuts are used to under-approximate the expected cost-to-go functions, namely the *Integer optimality cuts*, the *Lagrangian cuts* and the *strengthened Benders' cuts*. In particular, the strengthened Benders' cuts generated at a given node of the scenario tree are linear inequalities for which part of the coefficients are obtained by solving the linear relaxation of the sub-problems corresponding to its children nodes and by recording the dual values of the constraints linking the sub-problems to one another. In

addition to the strengthened Benders' cuts generated using the initial linear relaxation of the children sub-problems, we propose to generate additional strengthened Benders' cuts using an improved linear relaxation of these sub-problems. We thus introduce a cutting-plane generation procedure that takes advantage of previously published results on the polyhedral structure of the SULS to iteratively strengthen the relaxation of these sub-problems. Our computational results show that the joint use of the strengthened Benders' cuts obtained using the initial relaxation and the ones obtained using an improved relaxation of the sub-problems leads to a significant improvement of the computational efficiency of the SDDiP algorithm.

- Finally, as proposed e.g. in [53] and [82], before actually running the SDDiP algorithm as defined in [105], we introduce an initial phase in which only strengthened Benders' cuts are generated on a dynamic programming formulation using the initial continuous state variables. This strategy relies on the idea that, even if the obtained cuts do not satisfy the tightness condition necessary to theoretically ensure the global convergence of the SDDiP algorithm, they enable to build a first under-approximation of the expected cost-to-go functions with a reduced computational effort as each sub-problem involves a limited number of binary variables. This initial under-approximation is then further refined in a second phase based on a dynamic programming formulation using a binary approximation of the state variables.

Note that the use of a sub-tree decomposition (or node aggregation) has been explored by several authors in the context of stochastic dynamic programming. Cerisola and Ramos [26] studied a multi-stage stochastic linear problem for hydro-power generation scheduling. They proposed to decompose the scenario tree into connected sub-trees and presented several node aggregation protocols to generate these sub-trees. Later, authors of [28] considered multi-stage stochastic mixed-integer programs and developed a stochastic dynamic programming approach in which the original scenario tree is decomposed into sub-trees. However, their algorithm significantly differs from the SDDiP algorithm as it builds an upper envelope of the expected cost-to-go functions and only provides a heuristic solution whereas the SDDiP algorithm under-approximates the expected cost-to-go functions and is capable of providing both a lower and an upper bound of the optimal value. Note that authors of [7] and [37] recently presented an extension on the algorithm proposed in [28] aiming at exploiting parallel computing to further reduce the computation time.

The contributions of this work are threefold following these three main ideas. First, we propose a new extension of the SDDiP algorithm in which a partial decomposition of the scenario tree is used to generate sub-problems. To the best of our knowledge, this is the first time such an extension is studied in the context of the SDDiP algorithm. Second, we propose to take advantage of the tree structure of the sub-problems to exploit results on the polyhedral structure of the SULS and generate additional strengthened Benders' cuts to approximate the expected cost-to-go functions. Third, we carry out extensive computational experiments to assess the performance of the proposed algorithm at solving the SULS. We thus compare its performance with the one of a stand-alone mathematical solver and the one of the SDDiP algorithm proposed in [105]. The results show that this new algorithm outperforms ILOG-CPLEX and the SDDiP algorithm at solving large-size instances of the SULS.

The remaining part of this paper is organized as follows. Section 5.2 introduces a deterministic equivalent mixed-integer linear programming formulation and a stochastic dynamic programming formulation of the SULS. Section 5.3 presents the extension of the SDDiP algorithm in which a partial decomposition of the scenario tree is used. Section 5.4 then describes two further enhancements of the algorithm. Finally, the results of our computational experiments are reported in Section 5.5. Conclusions and directions for further works are discussed in Section 6.6.

5.2 Mathematical formulations

We assume a stochastic input process with finite probability space. The resulting information structure can be represented by a scenario tree. With a slight abuse of notation, we will refer to this scenario tree (and all other scenario sub-trees involved in the present work) by mentioning only its set of nodes $\mathcal{V}$. Each node $n \in \mathcal{V}$ corresponds to a single time period t^n and a single-stage σ^n. Let $\mathcal{V}^t$ be the set of nodes belonging to time period t. Each node n represents the state of the system that can be distinguished by the information unfolded up to time period t^n. Each node n has a unique predecessor node denoted a^n belonging to time period $t^n - 1$. By convention, the root node of the scenario tree is indexed by 1 and a^1 is set to 0. At any non-leaf node of the tree, one or several branches indicate future possible outcomes of the random variables from the current node. Let $\mathcal{C}(n)$ be the set of immediate children of node n, $\mathcal{V}(n)$ the sub-tree of $\mathcal{V}$ rooted in n and $\mathcal{L}(n)$ the set of leaf nodes belonging to $\mathcal{V}(n)$. The probability associated with the state represented by node n is denoted by ρ^n. A scenario is defined as a path from the root node to a leaf node in the scenario tree and represents a possible outcome of the stochastic input parameters over the whole planning horizon. The set of nodes on the path from node n to node m is denoted by $\mathcal{P}(n, m)$. The reader can refer to Figure 3.2 in Subsection 2.2.2 for an illustration of these notations on a small scenario tree.

The stochastic input parameters are defined as follows:

- d^n: demand at node $n \in \mathcal{V}$,
- f^n: set-up cost at node $n \in \mathcal{V}$,
- h^n: unit inventory holding cost at node $n \in \mathcal{V}$,
- g^n: unit production cost at node $n \in \mathcal{V}$.

Moreover, we assume that at each stage, the realization of the random parameters happens before we have to make a decision for this stage. This means that the values of d^n, f^n, h^n and g^n, for all $n \in \mathcal{T}^\sigma$, are assumed to be known at the beginning of the first period belonging to stage σ.

5.2.1 Extensive MILP formulation

Based on the uncertainty representation described above, the SULS can be reformulated as an equivalent deterministic model in the form of an MILP. We introduce the following decision variables:

- X^n: quantity produced at node $n \in \mathcal{V}$,
- $Y^n = 1$ if a set-up for production is carried out at node $n \in \mathcal{V}$, $Y^n = 0$ otherwise,
- S^n: inventory level at node $n \in \mathcal{V}$,

This leads to the following MILP formulation:

$$\min \sum_{n \in \mathcal{V}} \rho^n (f^n Y^n + h^n S^n + g^n X^n) \tag{5.1}$$

$$X^n \leq M^n Y^n \qquad \forall n \in \mathcal{V} \tag{5.2}$$

$$S^n + d^n = X^n + S^{a^n} \qquad \forall n \in \mathcal{V} \tag{5.3}$$

$$X^n, S^n \geq 0, Y^n \in \{0, 1\} \qquad \forall n \in \mathcal{V} \tag{5.4}$$

The objective function (5.1) aims at minimizing the expected total set-up, inventory holding and production costs over all nodes of the scenario tree. Constraints (5.2) link the production quantity

variables to the set-up variables. Note that the value of constant M^n can be set by using an upper bound on the quantity to be processed at node n, usually defined as the maximum future demand as seen from node n, i.e. $M^n = \max_{\ell \in \mathcal{L}(n)} d^{n\ell}$, where $d^{n\ell} = \sum_{m \in \mathcal{P}(n,\ell)} d^m$. Constraints (5.3) are the inventory balance constraints. Constraints (5.4) provide the decision variables domain.

Several MILP formulation strengthening techniques have been investigated for this problem: see Section 2.3.2 for a detailed presentation of the valid inequalities proposed in [16] and [45].

Problem (5.1)-(5.4) can thus be solved using MILP solvers. Nonetheless, the size of the formulation grows exponentially fast with the number of nodes $|\mathcal{V}|$ in the scenario tree, leading to prohibitive computation times in practice. We thus investigate in what follows a dynamic programming formulation which serves as a basis to develop a decomposition algorithm to solve the problem.

5.2.2 Dynamic programming formulation

An alternative to the extensive formulation of the SULS discussed above is a dynamic programming formulation involving nested expected cost-to-go functions. This approach decomposes the original problem into a series of smaller sub-problems linked together by dynamic programming equations. When applying the SDDiP algorithm proposed in [105] on the SULS, a full decomposition of the problem is carried out, resulting in a large number of small sub-problems. Each of these sub-problems is a small deterministic lot-sizing problem aiming at planning production on a subset of nodes corresponding to a single scenario and a single decision stage. In what follows, we propose to consider a partial decomposition of the problem resulting in a smaller number of larger sub-problems, each one being a stochastic lot-sizing problem aiming at planning production on a sub-tree of the original scenario tree.

We introduce some additional notation in order to explain how this partial decomposition is carried out. We first partition the set of decision stages $\mathcal{S} = \{1, \ldots, \Sigma\}$ into a series of macro-stages $\mathcal{G} = \{1, \ldots, \Gamma\}$, where each macro-stage $\gamma \in \mathcal{G}$ contains a number of consecutive stages denoted $\mathcal{S}(\gamma)$. We let $t(\gamma)$ (resp. $t'(\gamma)$) represent the first (resp. the last) time period belonging to macro-stage γ.

Using the set of macro-stages $\mathcal{G}$ defined above, we can decompose the scenario tree $\mathcal{V}$ into a series of smaller sub-trees as follows. For a given macro-stage γ, each node η belonging to the first time period in γ, i.e. each node $\eta \in \mathcal{V}^{t(\gamma)}$, is the root node of a sub-tree defined by the set of nodes $\mathcal{W}^\eta = \cup_{t=t(\gamma),\ldots,t'(\gamma)} \mathcal{V}^t \cap \mathcal{V}(\eta)$. We recall that $\mathcal{V}(\eta)$ is the sub-tree of $\mathcal{V}$ rooted in η, $\mathcal{W}^\eta$ is thus the restriction of $\mathcal{V}(\eta)$ to the nodes belonging to macro-stage γ. Let $\mathfrak{L}(\eta) = \mathcal{W}^\eta \cap \mathcal{V}^{t'(\gamma)}$ be the set of leaf nodes of sub-tree $\mathcal{W}^\eta$. Finally, we denote as $\mho = \cup_{\gamma \in \mathcal{G}} \mathcal{V}^{t(\gamma)}$ the set of sub-tree root nodes induced by $\mathcal{G}$.

To illustrate the notation related to the macro-stages, we use the scenario tree depicted in Figure 5.1. The set of stages $\mathcal{S}$ is partitioned into $\Gamma = 2$ macro-stages with $\mathcal{S}(1) = \{1, 2\}$ and $\mathcal{S}(2) = \{3, 4\}$. The first time period of macro-stage $\gamma = 1$ is $t(1) = 1$, its last time period is $t'(1) = 6$. Similarly, we have $t(2) = 7$ and $t'(2) = 12$. In this case, the set of sub-tree root nodes is $\mho = \{1, 10, 13, 16, 19\}$. With this partition, node $\eta = 1$ is the root node of the sub-tree $\mathcal{W}^1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ involving the set of leaf nodes $\mathfrak{L}(1) = \{6, 9\}$. Node $\eta = 10$ is the root node of sub-tree $\mathcal{W}^{10} = \{10, 11, 12, 22, 23, 24, 25, 26, 27\}$ involving the set of leaf nodes $\mathfrak{L}(10) = \{24, 27\}$. Sub-trees $\mathcal{W}^{13}$, $\mathcal{W}^{16}$ and $\mathcal{W}^{19}$ are defined in the same way as sub-tree $\mathcal{W}^{10}$.

For each node $\eta \in \mho$, sub-problem P^η is formulated as:

$$\mathscr{Q}^\eta(S^{a^\eta}) = \min \sum_{n \in \mathcal{W}^\eta} \rho^n (f^n Y^n + h^n S^n + g^n X^n) + \sum_{\ell \in \mathfrak{L}(\eta)} \sum_{m \in \mathcal{C}(\ell)} \mathscr{Q}^m(S^\ell) \tag{5.5}$$

$$X^n \leq M^n Y^n \qquad \forall n \in \mathcal{W}^\eta \tag{5.6}$$

$$S^n + d^n = S^{a^n} + X^n \qquad \forall n \in \mathcal{W}^\eta \tag{5.7}$$

$$X^n, S^n \geq 0, Y^n \in \{0, 1\} \qquad \forall n \in \mathcal{W}^\eta \tag{5.8}$$

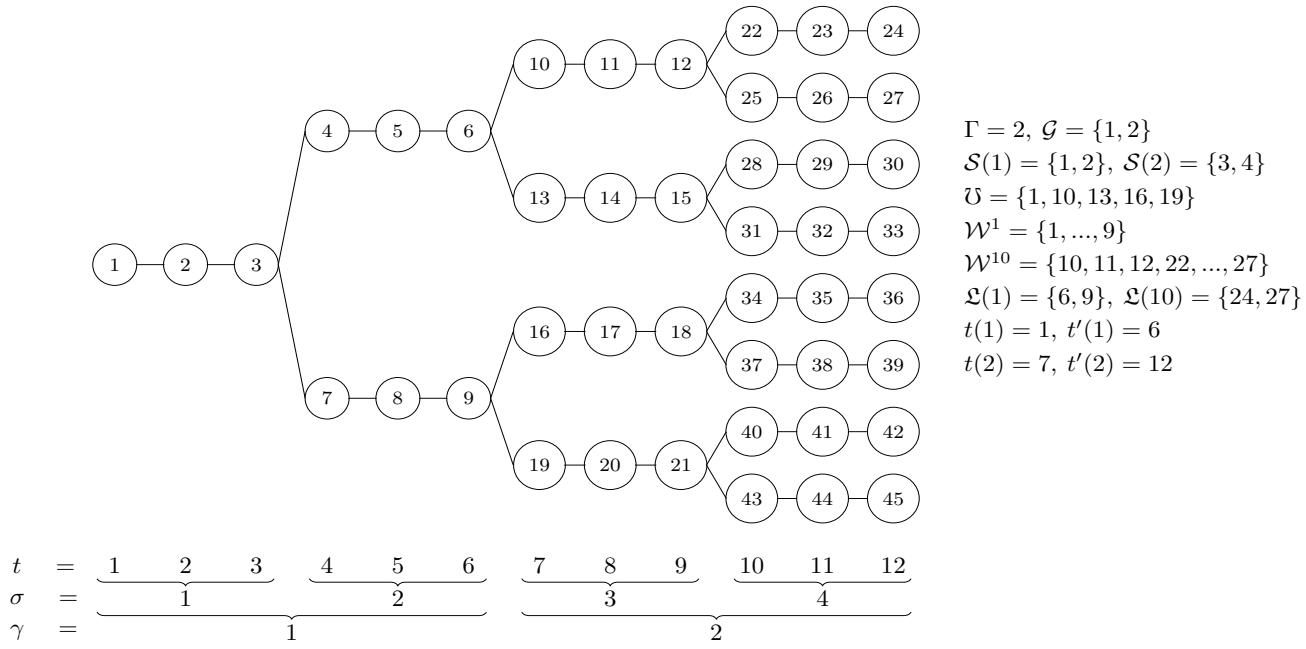


Figure 5.1: Partial decomposition of the scenario tree

Sub-problem P^η thus focuses on defining the production plan on sub-tree $\mathcal{W}^\eta$ based on the entering stock level S^{a^η} imposed by the parent node of η in the scenario tree. The objective function comprises two terms: a term related to the expected set-up, production and inventory holding costs over sub-tree $\mathcal{W}^\eta$ and a term which represents the expected future costs incurred by the production decisions made in sub-tree $\mathcal{W}^\eta$.

In (5.5), $\mathscr{Q}^\eta(S^{a^\eta})$ denotes the optimal value of sub-problem P^η as a function of the entering stock level S^{a^η} and $\mathscr{Q}^m(S^\ell)$ the optimal value of sub-problem P^m as a function of the entering stock level S^ℓ. The expected cost-to-go function at node $\ell \in \mathfrak{L}(\eta)$ is defined as the expected value of $\mathscr{Q}^m(\cdot)$ over all the children of ℓ in the initial scenario tree $\mathcal{V}$, i.e. over all $m \in \mathcal{C}(\ell)$, which gives $\mathcal{Q}^\ell(\cdot) = \sum_{m \in \mathcal{C}(\ell)} Q^m(\cdot)$. The expected future costs of the decisions made in $\mathcal{W}^\eta$ are thus computed as the sum, over all nodes $\ell \in \mathfrak{L}(\eta)$, of $\mathcal{Q}^\ell(S^\ell)$.

We note that in case of $\mathcal{G} \equiv \mathcal{S}$, i.e. in case each macro-stage corresponds to a single initial decision stage, each sub-tree $\mathcal{W}^\eta$ reduces to a set of nodes belonging to a single deterministic scenario involving T^{σ^η} periods and we obtain a decomposition similar to the one used in [105].

5.3 Sub-tree-based SDDiP algorithm

We now present the proposed extension of the SDDiP algorithm applied to the SULS. This extension relies on the dynamic programming formulation (5.5)-(5.8) and corresponds to a partial decomposition of the original problem into a set of smaller problems, each one expressed on a sub-tree of the scenario tree. As described in the SDDiP proposed in [105], the main idea is to solve a sequence of sub-problems in which the expected cost-to-go functions $\mathcal{Q}^\ell(\cdot), \ell \in \mathfrak{L}(\eta)$, of each sub-problem $P^\eta, \eta \in \mathfrak{V}$, are iteratively approximated by a piece-wise linear function. However, whereas the original SDDiP considers a large number of small deterministic sub-problems, we use a smaller number of medium-size stochastic sub-problems.

Note that a key assumption for developing the proposed algorithm is that the scenario tree displays the stage-wise independence property. When there are several time periods per decision stage, this property can be defined as follows. For any two nodes m and m' belonging to stage $\sigma - 1$ and such that $t^m = t^{m'} = \max\{t, t \in \mathcal{T}^{\sigma-1}\}$, the set of nodes $\cup_{t \in \mathcal{T}^\sigma} \mathcal{V}^t \cap \mathcal{V}(m)$ and $\cup_{t \in \mathcal{T}^\sigma} \mathcal{V}^t \cap \mathcal{V}(m')$

are defined by identical data and conditional probabilities.

Straightforwardly, when the stage-wise independence property holds, for any two nodes m and m' belonging to the last period $t'(\gamma-1)$ of macro-stage $\gamma-1$, the two sets $\cup_{\eta\in\mathcal{C}(m)}\mathcal{W}^\eta$, and $\cup_{\eta\in\mathcal{C}(m')}\mathcal{W}^\eta$ contain $R^\gamma = |\mathcal{C}(m)| = |\mathcal{C}(m')|$ sub-trees defined by identical data and conditional probabilities. The stochastic process can thus be represented at macro-stage γ by a set $\mathcal{R}^\gamma = \{1,\dots,R^\gamma\}$ of independent realizations. Each realization $\mathcal{X}^{\gamma,\zeta}$ corresponds to a subtree describing one of the possible evolutions of the uncertain parameters over periods $t(\gamma),\dots,t'(\gamma)$. Let $\xi^{\gamma,\zeta}$ denote the root node of $\mathcal{X}^{\gamma,\zeta}$ and $\mathfrak{L}(\gamma,\zeta)$ denote the set of its leaf nodes.

The expected cost-to-go functions thus depend only on the macro-stage rather than on the node, i.e. we have $\mathcal{Q}^m(\cdot)\equiv\mathcal{Q}^\gamma(\cdot)$, for all $m\in\mathcal{V}^{t'(\gamma)}$. Hence, only one expected cost-to-go function has to be approximated per macro-stage and the cuts generated at different nodes $m\in\mathcal{V}^{t'(\gamma)}$ are added to a single set of cuts defining the piece-wise linear approximation of function $\mathcal{Q}^\gamma(\cdot)$. As a consequence, we can define a single sub-problem P^γ per macro-stage and each sub-problem $P^\eta, \eta\in\mho$, will be described as $P^{\gamma^\eta}(S^{a^\eta},\mathcal{X}^{\gamma^\eta,\zeta})$ where $\mathcal{X}^{\gamma^\eta,\zeta}$ is the realization corresponding to $\mathcal{W}^\eta$.

5.3.1 Sub-problem reformulation

We first describe how each sub-problem $P^\gamma(S^m,\mathcal{X}^{\gamma,\zeta})$, for $m\in\mathcal{V}^{t'(\gamma-1)}$ and $\zeta\in\mathcal{R}^\gamma$, can be reformulated to introduce binary state variables.

Namely, in the SULS, the state variables are the continuous inventory variables S^n. As the SDDiP developed in [105] requires the state variables to be binary, we first carry out a binary approximation of the state variables before applying the algorithm to our problem. This binary approximation is obtained by replacing the continuous variable S^n by a set of binary variables $U^{n,\beta}$ such that $S^n=\sum_{\beta\in\mathcal{B}}2^\beta U^{n,\beta}$, where $\mathcal{B}=\{1,\dots,B\}$. We have $U^{n,\beta}=1$ if coefficient 2^β is used to compute the value of S^n and $U^{n,\beta}=0$ otherwise. We note however that this binary approximation is not needed for all inventory variables, but only for those coupling the sub-problems $P^\gamma(\cdot,\cdot)$, to one another. Thus, in sub-problem $P^\gamma(S^m,\mathcal{X}^{\gamma,\zeta})$, we use a binary approximation for the entering stock S^m at root node $\xi^{\gamma,\zeta}$ and for the leaving stock S^ℓ at each leaf node $\ell\in\mathfrak{L}(\gamma,\zeta)$.

Then, as indicated in [105], we introduce local copies of the binary state variables relative to root node $\xi^{\gamma,\zeta}$. More precisely, $\hat{U}^{\xi^{\gamma,\zeta},\beta}$ is an auxiliary continuous decision variable representing the value of the state variable $U^{m,\beta}$ at the parent node m. It is thus a local copy in problem $P^\gamma(S^m,\mathcal{X}^{\gamma,\zeta})$ of the state variable $U^{m,\beta}$, the value of which is considered as a given input parameter for this problem.

This leads to the following reformulation of sub-problem $P^\gamma(U^m,\mathcal{X}^{\gamma,\zeta})$:

$$\mathscr{Q}^{\gamma,\zeta}(U^m)=\min\sum_{n\in\mathcal{X}^{\gamma,\zeta}}\rho^n(f^nY^n+h^nS^n+g^nX^n)+\sum_{\ell\in\mathfrak{L}(\gamma,\zeta)}\mathcal{Q}^\gamma(U^\ell) \tag{5.9}$$

$$X^n\le M^nY^n \qquad \forall n\in\mathcal{X}^{\gamma,\zeta} \tag{5.10}$$

$$S^{\xi^{\gamma,\zeta}}+d^{\xi^{\gamma,\zeta}}=\sum_{\beta\in\mathcal{B}}2^\beta\hat{U}^{\xi^{\gamma,\zeta},\beta}+X^{\xi^{\gamma,\zeta}} \tag{5.11}$$

$$\hat{U}^{\xi^{\gamma,\zeta},\beta}=U^{m,\beta} \qquad \forall\beta\in\mathcal{B} \tag{5.12}$$

$$S^n+d^n=S^{a^n}+X^n \qquad \forall n\in\mathcal{X}^{\gamma,\zeta}\setminus\{\xi^{\gamma,\zeta}\} \tag{5.13}$$

$$S^\ell=\textstyle\sum_{\beta\in\mathcal{B}}2^\beta U^{\ell,\beta} \qquad \forall\ell\in\mathfrak{L}(\gamma,\zeta) \tag{5.14}$$

$$\hat{U}^{\xi^{\gamma,\zeta},\beta}\in[0,1] \qquad \forall\beta\in\mathcal{B} \tag{5.15}$$

$$U^{\ell,\beta}\in\{0,1\} \qquad \forall\ell\in\mathfrak{L}(\gamma,\zeta),\forall\beta\in\mathcal{B} \tag{5.16}$$

$$X^n,S^n\ge0, Y^n\in\{0,1\} \qquad \forall n\in\mathcal{X}^{\gamma,\zeta} \tag{5.17}$$

where U^n denotes the vector of binary variables $U^n=(U^{n0},\dots,U^{n\beta},\dots,U^{nB})$.

In this reformulation, Constraint (5.11) corresponds to the inventory balance at node $\xi^{\gamma,\zeta}$ in which the entering stock level S^m is computed using the auxiliary variables $\hat{U}^{\xi^{\gamma,\zeta}\beta}$. Equalities (5.12) are copy constraints ensuring that the value of each auxiliary variable $\hat{U}^{\xi^{\gamma,\zeta},\beta}$ is equal to the value of the corresponding state variable $U^{m,\beta}$ imposed by the parent node m. Constraints (5.13) ensure the inventory balance at each node of sub-tree $\mathcal{X}^{\gamma,\zeta}$ except the root node $\xi^{\gamma,\zeta}$. Constraints (5.14) define, for each leaf node $\ell \in \mathfrak{L}(\gamma,\zeta)$, the value of the binary variables $U^{\ell,\beta}$, which will be used to compute the future expected costs as $\mathcal{Q}^{\gamma}(U^{\ell}) = \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} \mathscr{Q}^{\gamma+1,r'}(U^{\ell})$. Note that, although variables $\hat{U}^{\xi^{\gamma,\zeta},\beta}$ and constraints (5.12) are redundant for sub-problem $P^{\gamma}(U^m, \mathcal{X}^{\gamma,\zeta})$, they will play a key role in the generation of the Lagrangian and strengthened Benders' cuts used in the SDDiP algorithm to approximate the expected cost-to-functions.

The main components of the proposed sub-tree-based SDDiP algorithm applied to the SULS are described in the following.

5.3.2 Sampling step

In the sampling step, a subset of W scenarios, i.e. a set of paths going from the root node to a leaf node, are randomly selected. Let $\Omega_{\upsilon} = \{\omega_{\upsilon}^1, \ldots, \omega_{\upsilon}^w, \ldots, \omega_{\upsilon}^W\}$ be the set of sampled scenarios, ω_{υ}^w be the set of nodes belonging to scenario w at iteration υ and $\zeta_{\upsilon}^{w,\gamma}$ be the index of the realization in $\mathcal{R}^{\gamma}$ containing the values of the uncertain parameters in scenario ω_{υ}^w at macro-stage γ.

5.3.3 Forward step

At iteration υ, the forward step proceeds stage-wise from $\gamma = 1$ to Γ. For each sampled scenario ω_{υ}^w and each macro-stage γ, we solve problem $P_{\upsilon}^{\gamma}(U_{\upsilon}^m, \mathcal{X}^{\gamma,\zeta_{\upsilon}^{w,\gamma}})$ where $m = \omega_{\upsilon}^w \cap \mathcal{V}^{t'(\gamma-1)}$ is the node in the sampled scenario ω_{υ}^w belonging to the last period of γ. To solve this problem, the expected future costs are computed using an approximate representation of the expected cost-to-go functions $\mathcal{Q}^{\gamma}(\cdot)$.

Let $\psi_{\upsilon}^{\gamma}(\cdot)$ be the approximation of the expected cost-to-go function $\mathcal{Q}^{\gamma}(\cdot)$ available at iteration υ for macro-stage γ. It is defined by the set of supporting hyperplanes generated until iteration υ. We thus have:

$$\psi_{\upsilon}^{\gamma}(U^{\ell}) = \min\{\theta^{\gamma\ell} : \theta^{\gamma\ell} \geq \sum_{\zeta \in \mathcal{R}^{\gamma+1}} \nu_u^{\gamma+1,\zeta} + \pi_u^{\gamma+1,\zeta} U^{\ell} \quad \forall \quad u = \{1, \ldots, \upsilon - 1\}\} \tag{5.18}$$

where $\nu_u^{\gamma+1,\zeta}$ and $\pi_u^{\gamma+1,\zeta}$ are the coefficients of the cut generated at iteration $u < \upsilon$ by considering realization $\zeta \in \mathcal{R}^{\gamma+1}$. This leads to the following sub-problem $\hat{P}_{\upsilon}^{\gamma}(U_{\upsilon}^m, \psi_{\upsilon}^{\gamma}, \mathcal{X}^{\gamma,\zeta_{\upsilon}^{w,\gamma}})$:

$$\hat{\mathscr{Q}}_{\upsilon}^{\gamma,\zeta_{\upsilon}^{w,\gamma}}(U_{\upsilon}^m) = \min \sum_{n \in \mathcal{X}^{\gamma,\zeta_{\upsilon}^{w,\gamma}}} \rho^n (f^n Y^n + h^n S^n + g^n X^n) + \sum_{\ell \in \mathfrak{L}(\gamma,\zeta_{\upsilon}^{w,\gamma})} \theta^{\gamma\ell} \tag{5.19}$$

$$\theta^{\gamma\ell} \geq \sum_{\zeta \in \mathcal{R}^{\gamma+1}} \nu_u^{\gamma+1,\zeta} + \pi_u^{\gamma+1,\zeta} U^{\ell} \qquad \forall \quad u = 1, \ldots, \upsilon - 1, \quad \forall \ell \in \mathfrak{L}(\gamma, \zeta_{\upsilon}^{w,\gamma}) \tag{5.20}$$

Constraints $(5.10) - (5.17)$ for $\zeta = \zeta_{\upsilon}^{w,\gamma}$

The forward step at iteration υ ends when sub-problem $\hat{P}_{\upsilon}^{\gamma}(U_{\upsilon}^m, \psi_{\upsilon}^{\gamma}, \mathcal{X}^{\gamma,\zeta_{\upsilon}^{w,\gamma}})$ has been solved for all sampled scenarios and all macro-stages. Its output is a feasible production plan for all nodes belonging to a sampled scenario. In particular, it provides a value U_{υ}^m for all state variables U^m such that $m \in \omega_{\upsilon}^w \cap \mathcal{V}^{t'(\gamma)}, \gamma \in \mathcal{G}, w = 1, \ldots, W$. These values will be used in the backward step to generate additional cuts and improve the approximation of the expected cost-to-go functions $\mathcal{Q}^{\gamma}(\cdot), \gamma \in \mathcal{G}$.

5.3.4 Backward step

The aim of the backward step is to update the current approximation $\psi_\upsilon^\gamma(\cdot)$ of the expected cost-to-go function $\mathcal{Q}^\gamma(\cdot)$ for each macro-stage γ by generating new supporting hyperplanes and obtain a better approximation which is denoted $\psi_{\upsilon+1}^\gamma(\cdot)$.

This step starts from macro-stage Γ and goes back to macro-stage 1. Note that the sub-problems relative to macro-stage Γ do not have any expected future costs, therefore $\psi_\upsilon^\Gamma \equiv 0$, for all υ. At each macro-stage $\gamma = \Gamma - 1, \ldots, 1$, the updating of the approximation of $\mathcal{Q}^\gamma(\cdot)$ is carried out as follows. For each scenario $w = 1, \ldots, W$, each node $m \in \omega_\upsilon^w \cap \mathcal{V}^{t'(\gamma)}$ and each realization $\zeta \in \mathcal{R}^{\gamma+1}$, we solve a suitable relaxation of $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$ and collect the cut coefficients $\{\nu_\upsilon^{\gamma+1,\zeta}, \pi_\upsilon^{\gamma+1,\zeta}\}$. These coefficients are then used to generate a new linear inequality of type (5.18) to be added to the current approximation of $\mathcal{Q}^\gamma(\cdot)$. The backward step continues iteratively until the approximation of the expected cost-to-go function at macro-stage $\gamma = 1$ is updated. Since $\psi_{\upsilon+1}^1$ is an under-approximation of the expected cost-to-go function $\mathcal{Q}^1(\cdot)$, the optimal value $\hat{\mathscr{Q}}_{\upsilon+1}^{1,1}(0)$ of problem $\hat{P}_\upsilon^1(0, \psi_{\upsilon+1}^1, \mathcal{X}^{1,1})$, provides a lower bound of the optimal value of the stochastic problem.

Cut families

We now briefly recall the three types of cutting planes used in [105] to improve the approximation of the expected cost-to-go functions during the backward step. Let us consider a macro-stage γ, a scenario index w and the node $m = \omega_\upsilon^w \cap \mathcal{V}^{t'(\gamma)}$. Let U_υ^m be the value of the state variables U^m in the solution of problem $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_\upsilon^\gamma, \mathcal{X}^{\gamma,\zeta_\upsilon^{w,\gamma}})$ solved in the forward step of iteration υ. The three following cuts can be added to compute the approximation ψ_υ^γ of $\mathcal{Q}^\gamma(\cdot)$.

Integer optimality cut: The algorithm solves problem $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$, for each $\zeta \in \mathcal{R}^{\gamma+1}$, with an updated approximation $\psi_{\upsilon+1}^{\gamma+1}$ of $\mathcal{Q}^{\gamma+1}(\cdot)$. Let $\nu_{\upsilon+1}^{\gamma+1,\zeta}$ be its optimal objective value and $\bar{\nu}_{\upsilon+1}^{\gamma+1} = \sum_{\zeta \in \mathcal{R}^{\gamma+1}} \nu_{\upsilon+1}^{\gamma+1,\zeta}$. The Integer optimality cut takes the following form:

$$\theta^{\gamma,m} \geq \bar{\nu}_{\upsilon+1}^{\gamma+1} \Big(\sum_{\beta=0}^{B} (U_\upsilon^{m,\beta} - 1) U^{m,\beta} + \sum_{\beta=0}^{B} (U^{m,\beta} - 1) U_\upsilon^{m,\beta} \Big) + \bar{\nu}_{\upsilon+1}^{\gamma+1}$$

Lagrangian cut: We consider, for each $\zeta \in \mathcal{R}^{\gamma+1}$, the Lagrangian relaxation of problem $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$ in which the copy constraints (5.12) are dualized. Each corresponding Lagrangian dual problem is solved to optimality. The generated Lagrangian cut takes the form of inequality (5.18), where $\nu_\upsilon^{\gamma+1,\zeta}$ corresponds to the optimal value of the Lagrangian dual problem and coefficient $\pi_\upsilon^{\gamma+1,r,\beta}$ of variable $U^{m,\beta}$ to the optimal value of the Lagrangian multiplier relative to copy constraint $\hat{U}^{\xi^{\gamma,\zeta},\beta} = U_\upsilon^{m,\beta}$.

Strengthened Benders' cut: We solve, for each $\zeta \in \mathcal{R}^{\gamma+1}$, the linear relaxation of problem $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$. The value of each coefficient $\pi_\upsilon^{\gamma+1,r,\beta}$ is set to the dual value of the copy constraint $\hat{U}^{\xi^{\gamma,\zeta},\beta} = U_\upsilon^{m,\beta}$ in this linear relaxation. The value of $\nu_\upsilon^{\gamma+1,\zeta}$ is obtained by solving the Lagrangian relaxation of problem $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$ in which each copy constraint $\hat{U}^{\xi^{\gamma,\zeta},\beta} = U_\upsilon^{m,\beta}$ is dualized and its Lagrangian multiplier set to $\pi_\upsilon^{\gamma+1,r,\beta}$.

5.3.5 Stopping criteria

Two stopping criteria are commonly used for the SDDiP in the literature. The first one is based on a maximum number of consecutive iterations without any improvement of the lower bound, the second one on a maximum total number of iterations.

As a synthesis, the main steps of the proposed sub-tree-based SDDiP algorithm applied to the stochastic ULS are summarized in Algorithm 7.

Note that depending on the partition of $\mathcal{S}$, the number of macro-stages Γ can take any value between 1 and Σ. For $\Gamma = 1$, the forward step corresponds to solving the original problem (5.1)-(5.4) defined on the whole scenario tree $\mathcal{V}$ and no backward step is needed: Algorithm 7 thus directly solves the stochastic problem as a MILP, without any decomposition. For $\Gamma = \Sigma$, Algorithm 7 corresponds to the SDDiP algorithm of [105]. In general, we will have $\Gamma < \Sigma$, which means that the number of expected cost-to-go functions $\mathcal{Q}^\gamma(\cdot)$ to be approximated will be smaller in Algorithm 7 than the one to be handled in the SDDiP algorithm. This may have a positive impact on the global convergence of the algorithm. Namely, with $\Gamma < \Sigma$, each sub-problem $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ covers a larger portion of the planning horizon and uses an approximation of its expected future costs which will be globally better as it will rely on a smaller number of approximate expected cost-to-go functions. As a consequence, the feasible solution obtained by solving $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ at a given iteration of the algorithm will tend to be less myopic and thus to provide lower and upper bounds LB and UB of better quality. However, each sub-problem $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ is now an MILP expressed on a small sub-tree. In particular, the large number of binary variables $U^{\ell,\beta}$ needed to carry out the binary approximation of the leaving inventory at each leaf node $\ell \in \mathfrak{L}(\gamma, \zeta)$ makes its resolution computationally more expensive than the one of a sub-problem expressed on a deterministic scenario involving a single leaf node. In what follows, we thus discuss two algorithmic enhancements aiming at further improving the numerical efficiency of Algorithm 7.

5.4 Algorithmic enhancements

In this section, we aim at enhancing the numerical efficiency of Algorithm 7, mostly through a more efficient building of the approximation of the expected cost-to-go functions. In what follows, we detail the two proposed algorithmic enhancements.

5.4.1 Approximate sub-tree-based SDDiP

[53] and [82] both proposed to use an approximate variant of the SDDiP algorithm in which the state variables may be continuous. In this case, the finite convergence of the algorithm is not theoretically guaranteed but, as this approximation leads to a significant reduction of the computational effort required at each iteration of the algorithm, it may positively impact the solution quality in practice. We thus explain in what follows how this approximate SDDiP algorithm can be adapted to the case where a partial sub-tree-based decomposition of the scenario tree is used.

The algorithm is based on a reformulation of problem $P^\gamma(S^m, \mathcal{X}^{\gamma,\zeta})$ in which a single auxiliary variable $\tilde{S}^{\xi^{\gamma,\zeta}}$ is introduced. Variable $\tilde{S}^{\xi^{\gamma,\zeta}}$ can be seen as a local copy of the inventory variable at the parent node S^m in $P^\gamma(S^m, \mathcal{X}^{\gamma,\zeta})$. This results in the following reformulation of $P^\gamma(S^m, \mathcal{X}^{\gamma,\zeta})$:

$$\mathscr{Q}^{\gamma,\zeta}(S^m) = \min \sum_{n \in \mathcal{X}^{\gamma,\zeta}} \rho^n (f^n Y^n + h^n S^n + g^n X^n) + \sum_{\ell \in \mathfrak{L}(\gamma,\zeta)} \mathcal{Q}^\gamma(S^\ell) \tag{5.21}$$

$$S^{\xi^{\gamma,\zeta}} + d^{\xi^{\gamma,\zeta}} = \tilde{S}^{\xi^{\gamma,\zeta}} + X^{\tilde{S}^{\xi^{\gamma,\zeta}}} \tag{5.22}$$

$$\tilde{S}^{\xi^{\gamma,\zeta}} = S^m \tag{5.23}$$

$$S^n + d^n = S^{a^n} + X^n \qquad \forall n \in \mathcal{X}^{\gamma,\zeta} \setminus \{\xi^{\gamma,\zeta}\} \tag{5.24}$$

$$\text{Constraints } (5.6), (5.8) \tag{5.25}$$

In this reformulation, the expected cost-to-go function $\mathcal{Q}^\gamma(S^\ell) = \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} \mathscr{Q}^{\gamma+1,\zeta'}(S^\ell)$ is a function of the continuous state variable S^ℓ. We thus build an under-approximation of $\mathcal{Q}^\gamma(\cdot)$ through a set of linear cuts involving continuous variables S^ℓ instead of binary variables $U^{\ell,\beta}$. Let $\tilde{\psi}_v^\gamma(\cdot)$ be the

Algorithm 7: SDDiP algorithm

Initialize $LB \leftarrow -\infty, UB \leftarrow +\infty, \upsilon \leftarrow 1$
while *no stopping criterion is satisfied* **do**
  Sampling step
  Randomly select W scenarios $\Omega_\upsilon = \{\omega_\upsilon^1, ..., \omega_\upsilon^W\}$
  Forward step
  for $w = 1, ..., W$ **do**
    for $\gamma = 1, ..., \Gamma$ **do**
      Solve $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_\upsilon^\gamma, \mathcal{X}^{\gamma,\zeta_\upsilon^{w,\gamma}})$ for $m = \omega_\upsilon^w \cap \mathcal{V}^{t'(\gamma-1)}$
      Record u_υ^ℓ for $\ell = \omega_\upsilon^w \cap \mathfrak{L}(\gamma, \zeta_\upsilon^{w,\gamma})$
    end
    $\upsilon^w \leftarrow \sum_{n \in \omega_\upsilon^w} (f^n Y_\upsilon^n + h^n S_\upsilon^n + g^n X_\upsilon^n)$
  end
  $\hat{\mu} \leftarrow \sum_{w=1}^W \upsilon^w$ and $\hat{\sigma}^2 \leftarrow \frac{1}{W-1} \sum_{w=1}^W (\upsilon^w - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2} \frac{\hat{\sigma}}{\sqrt{W}}$
  Backward step
  for $\gamma = \Gamma - 1, ..., 1$ **do**
    for $w = 1, ..., W$ **do**
      Let $m = \omega_\upsilon^w \cap \mathcal{V}^{t'(\gamma)}$
      for $\zeta \in \mathcal{R}^{\gamma+1}$ **do**
        Solve the linear relaxation of $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$ and collect the coefficients of the strengthened Benders' cut
        Solve the Lagrangian relaxation of $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$ and collect the constant value of the strengthened Benders' cut
        Solve $\hat{P}_\upsilon^{\gamma+1}(U_\upsilon^m, \psi_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$ and collect the coefficients of the Integer optimality cut
        Solve the Lagrangian dual problem and collect the coefficients of the Lagrangian cut
      end
    end
    Add the three generated cuts to ψ_υ^γ to get $\psi_{\upsilon+1}^\gamma$
  end
  $LB \leftarrow \hat{\mathscr{Q}}_{\upsilon+1}^{1,1}(0)$
  $\upsilon \leftarrow \upsilon + 1$
end

approximation of the expected cost-to-go function $\mathcal{Q}^\gamma(\cdot)$ available at iteration υ for macro-stage γ in the approximate SDDiP algorithm. We have:

$$\tilde{\psi}_\upsilon^\gamma(S^\ell) = \min\{\theta^{\gamma\ell} : \theta^{\gamma\ell} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\tilde{\nu}_j^{\gamma+1,\zeta'} + \tilde{\pi}_j^{\gamma+1,\zeta'} S^\ell) \quad \forall j = \{1, ..., i-1\}\} \tag{5.26}$$

where $\tilde{\nu}_j^{\gamma+1,\zeta'}$ and $\tilde{\pi}_j^{\gamma+1,\zeta'}$ are the coefficients of the cut generated at iteration $j < \upsilon$ by considering realization $\zeta' \in \mathcal{R}^{\gamma+1}$. This leads to the following approximate sub-problem $\tilde{P}_\upsilon^\gamma(S_\upsilon^m, \tilde{\psi}_\upsilon^\gamma, \mathcal{X}^{\gamma,\zeta})$:

$$\tilde{\mathscr{Q}}_\upsilon^{\gamma,\zeta}(S^m) = \min \sum_{n \in \mathcal{X}^{\gamma,\zeta}} \rho^n (f^n Y^n + h^n S^n + g^n X^n) + \sum_{\ell \in \mathfrak{L}(\gamma,\zeta)} \theta^{\gamma\ell} \tag{5.27}$$

$$\theta^{\gamma\ell} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\tilde{\nu}_j^{\gamma+1,\zeta'} + \tilde{\pi}_j^{\gamma+1,\zeta'} S^\ell) \qquad \forall j = 1, ..., \upsilon - 1, \quad \forall \ell \in \mathfrak{L}(\gamma, \zeta) \tag{5.28}$$

$$\text{Constraints } (5.22) - (5.25) \tag{5.29}$$

In the backward step of the approximate SDDiP algorithm, only strengthened Benders' cuts are generated. More precisely, for each macro-stage $\gamma = \Gamma - 1, \ldots, 1$, the updating of the approximation of $\mathcal{Q}^\gamma(\cdot)$ is carried out as follows. For each node $m \in \Omega_\upsilon \cap \mathcal{V}^{t'(\gamma)}$ and each realization $\zeta' \in \mathcal{R}^{\gamma+1}$, we first solve the linear relaxation of $\tilde{P}_\upsilon^{\gamma+1}(S_\upsilon^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$, get the dual value of constraint (5.23) in its optimal solution and set $\tilde{\pi}_j^{\gamma+1,\zeta'}$ to this value. We then solve a Lagrangian relaxation of $\tilde{P}_\upsilon^{\gamma+1}(S_\upsilon^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$ in which constraint $\tilde{S}^{\xi^{\gamma,\zeta}} = S^m$ has been dualized with a Lagrangian multiplier set to $\tilde{\pi}_j^{\gamma+1,\zeta'}$. We record the optimal value of this Lagrangian relaxation and set $\tilde{\nu}_j^{\gamma+1,\zeta'}$ to this value. Once this procedure is carried out for all realization of $\zeta' \in \mathcal{R}^{\gamma+1}$, we get the cut $\theta^{\gamma,m} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\tilde{\nu}_j^{\gamma+1,\zeta'} + \tilde{\pi}_j^{\gamma+1,\zeta'} S^m)$ to be added to $\tilde{\psi}_\upsilon^\gamma$ to get $\tilde{\psi}_{\upsilon+1}^\gamma$.

In our numerical experiments, this approximate sub-tree-based SDDiP algorithm is used in an initial phase which is carried out before actually running Algorithm 7. The objective of this initial phase is to build a first under-approximation of each expected cost-to-go functions $\mathcal{Q}^\gamma(\cdot)$ and obtain a good initial lower bound for the problem with a reduced computational effort. This initial lower bound will then be further improved during a second phase in which Algorithm 7 is run and all Integer Optimality, Lagrangian and strengthened Benders' cuts are generated. Note that any cut of type $\theta^{\gamma\ell} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\tilde{\nu}_j^{\gamma+1,\zeta'} + \tilde{\pi}_j^{\gamma+1,\zeta'} S^\ell)$ generated during the first phase provides a cut of type $\theta^{\gamma\ell} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\nu_j^{\gamma+1,\zeta'} + \pi_j^{\gamma+1,\zeta'} U^\ell)$ which can be used in the second phase by setting $\nu_j^{\gamma,\zeta'} = \tilde{\nu}_\upsilon^{\gamma,\zeta'}$ and $\pi_j^{\gamma,r',\beta} = \tilde{\pi}_\upsilon^{\gamma,\zeta'}$, for $\beta \in \mathcal{B}$.

5.4.2 Leveraging the current knowledge on the polyhedral structure of the SULS

The second enhancement of the algorithm seeks to exploit the alternative MILP formulations currently known for SULS to generate additional strengthened Benders' cuts (and improve the approximation of the expected cost-to-go functions at a relatively limited computational effort) and speed up the resolution of the numerous MILPs to be solved over the course of Algorithm 7.

Generation of additional strengthened Benders' cuts

Recall that to, generate a strengthened Benders' cut to be added at a given iteration υ to the approximation of $\mathcal{Q}^{\gamma-1}(\cdot)$, the algorithm first solves, for each $r \in \mathcal{R}^\gamma$, the linear relaxation of problem $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_{\upsilon+1}^\gamma, \mathcal{X}^{\gamma,\zeta})$, where m is a node belonging to $\mathcal{V}^{t'(\gamma-1)}$. This linear relaxation can be computed using different formulations of $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_{\upsilon+1}^\gamma, \mathcal{X}^{\gamma,\zeta})$. A first option is to use the initial MILP formulation defined in Subsection 5.3.3. Other options consist in using the initial MILP formulation strengthened by path inequalities (2.12) or the initial MILP formulation strengthened by tree inequalities (2.13). Then, the algorithm collects the dual value of the copy constraint $\hat{U}^{\xi^{\gamma,\zeta},\beta} = U_\upsilon^{m,\beta}$ in the linear relaxation and set coefficient $\pi_\upsilon^{\gamma,\zeta,\beta}$ to it. The Lagrangian relaxation of problem $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_{\upsilon+1}^\gamma, \mathcal{X}^{\gamma,\zeta})$ in which each copy constraint $\hat{U}^{\xi^{\gamma,\zeta},\beta} = U_\upsilon^{m,\beta}$ is dualized with a Lagrangian multiplier set to $\pi_\upsilon^{\gamma,\zeta,\beta}$ is then solved. Its optimal value provides coefficient $\nu_\upsilon^{\gamma,\zeta}$.

A key observation here is that the dual value of constraint $\hat{U}^{\xi^{\gamma,\zeta},\beta} = U_\upsilon^{m,\beta}$ in the linear relaxation will vary according to the MILP formulation used for $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_{\upsilon+1}^\gamma, \mathcal{X}^{\gamma,\zeta})$. Hence, for a given value of the entering stock described by the binary vector U_υ^m, by considering the three alternative MILP formulations available for $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_{\upsilon+1}^\gamma, \mathcal{X}^{\gamma,\zeta})$, it is possible to generate three different strengthened Benders' cuts, i.e. three cuts corresponding to different values of coefficients $\{\nu_\upsilon^{\gamma,\zeta}, \pi_\upsilon^{\gamma,\zeta}\}$,

We point out here that, in general, there does not seem to be a dominance relationship between these three cuts. In other words, a cut generated using a stronger formulation of $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_{\upsilon+1}^\gamma, \mathcal{X}^{\gamma,\zeta})$ does not necessarily lead to a better approximation of $\mathcal{Q}^{\gamma-1}(\cdot)$. The reader is referred to Appendix 5.7 for a small example illustrating this point. We thus propose an extension of Algorithm 7 in which strengthened Benders' cuts based on the three MILP formulations available for $\hat{P}_\upsilon^\gamma(U_\upsilon^m, \psi_{\upsilon+1}^\gamma, \mathcal{X}^{\gamma,\zeta})$ are sequentially generated.

Before presenting this extension, we first note that valid inequalities generated for sub-problem $\hat{P}_v^\gamma(U_v^m, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ at a given iteration v are valid for any sub-problem $\hat{P}_j^\gamma(\cdot, \psi_j^\gamma, \mathcal{X}^{\gamma,\zeta})$ to be solved at iteration $j \leq v$. Namely, let us consider reformulation (5.9)-(5.17) of sub-problem $P^\gamma(U^m, \mathcal{X}^{\gamma,\zeta})$. Valid inequalities of type (2.12) and (2.13) can be expressed as follows:

$$\sum_{\beta \in \mathcal{B}} 2^\beta \hat{U}^{\xi^{\gamma,\zeta}\beta} + \sum_{n \in \mathscr{S}_{\mathscr{O}}} X^n + \sum_{n \in \bar{\mathscr{S}}_{\mathscr{O}}} \Delta_n(\mathscr{O}) Y^n \geq d^{1 n_O} \tag{5.30}$$

for any subset $\mathscr{O}$ of $\mathcal{X}^{\gamma,\zeta}$.

Note that inequalities (5.30) only use local copy variables $\hat{u}$ and thus remain valid for any value of the entering stock level described by the state variable U^m. Moreover, sub-problem $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$ and $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ only differ with respect to the objective function evaluation and have the same feasible space. Thus, any inequality valid for $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$ is also valid for $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$.

Let $\chi_v^{\gamma,\zeta} = \{\sum_{\beta \in \mathcal{B}} 2^\beta \hat{U}^{\xi^{\gamma,\zeta}\beta} + \sum_{n \in \mathscr{S}_{\mathscr{O}_c}} X^n + \sum_{n \in \bar{\mathscr{S}}_{\mathscr{O}_c}} \Delta_n(\mathscr{O}_c) Y^n \geq d^{1 n_{O_c}}, \mathscr{O}_c \subset \mathcal{X}^{\gamma,\zeta}, c = 1, \ldots, C_v^{\gamma,\zeta}\}$ denote the set of $C_v^{\gamma,\zeta}$ inequalities (2.12) and/or (2.13) related to sub-problem $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$ generated until the beginning of iteration v. Let $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta}, F)$ denote problem $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ expressed using formulation F where $F = IF(\emptyset)$ denote the initial formulation without any strengthening and $F = IF(\chi_v^{\gamma,\zeta})$ the initial formulation strengthened by a set of inequalities $\chi_v^{\gamma,\zeta}$.

We propose the following strategy to sequentially add strengthened Benders' cuts based on the three available MILP formulations to the approximations of the expected cost-to-go functions. This strategy is based on three increasing levels of formulation strengthening :

- Level $\lambda = 0$: Algorithm 7 is run as described in Section 5.3, i.e. we solve the linear relaxation of sub-problems $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta}, IF(\emptyset))$ to obtain the cut coefficients. The algorithm moves to the next level after a predefined number of consecutive iterations.

- Level $\lambda = 1$: Algorithm 7 is run using the linear relaxation of sub-problems $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta}, IF(\chi_v^{\gamma,\zeta}))$ to obtain the cut coefficients. In this level, at each iteration, only path inequalities (2.12) are added to $\chi_v^{\gamma,\zeta}$ using a single run of a cutting plane generation procedure based on the separation algorithm presented in [17]. The algorithm moves to the next level after no violated path inequalities have been found during a predefined number of consecutive iterations.

- Level $\lambda = 2$: Algorithm 7 is run using the linear relaxation of sub-problems $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta}, IF(\chi_v^{\gamma,\zeta}))$ to obtain the cut coefficients. In this level, at each iteration, tree inequalities (2.13) are added to $\chi_v^{\gamma,\zeta}$ using a single run of the cutting plane generation procedure presented in [45]. The algorithm moves to the next level after no violated tree inequalities have been found during a predefined number of consecutive iterations.

As will be shown by the numerical results to be presented in Section 5.5, the joint use of strengthened Benders' cuts generated using the three alternative MILP formulations available for the sub-problems allows to significantly improve the quality of the solution provided by the algorithm. Note that an alternative strategy could be to simultaneously add strengthened Benders' cuts based on these three MILPs formulations at each iteration of the algorithm. However, this would significantly increase the computational effort carried out at each iteration as it would require the resolution of many additional linear and Lagrangian relaxations of problem $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta}, \cdot)$, namely one for each formulation $F = \{IF(\emptyset), IF(\chi_v^{\gamma,\zeta})\}$, each set of inequalities $\chi_v^{\gamma,\zeta}$, each realization $\zeta \in \mathcal{R}^\gamma$ and each macro-stage $\gamma \in \mathcal{G}$. This would lead to a strong decrease in the number of iterations carried out by the algorithm within the allotted computation time, which could negatively impact its performance. We thus chose to implement a strategy in which the strengthened Benders' cuts based on the alternative MILPs formulations are added sequentially to the expected cost-to-go functions approximations.

Sub-problem resolution with strengthened MILP formulations

Over the course of Algorithm 7, a rather large number of MILPs have to be solved. More precisely, at each iteration of the algorithm, there are one MILP to be solved for each scenario and each macro-stage in the forward step and two MILPS (the sub-problem itself and its Lagrangian relaxation) to be solved for each scenario, each macro-stage and each realization per macro-stage in the backward step. Thus, even if each of these MILPs is of moderate size, their resolution might be computationally expensive.

Valid inequalities (2.12) and (2.13) can be used to speed up the resolution of the various MILPs involved in Algorithm 7 through a strengthening of their linear relaxation and an improvement of the lower bounds used at each node of the Branch & Bound search tree. In order to save the computational effort needed to run the related cutting plane generation procedures, we propose to strengthen the MILP formulation of each sub-problem $\hat{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ using only the valid inequalities already added to the current set of inequalities $\chi_v^{\gamma,\zeta}$ and to solve each sub-problem using formulation $IF(\chi_v^{\gamma,\zeta})$.

For the sake of clarity, in Subsection 5.4.2, we focused on explaining how this second algorithmic enhancement is carried out in Algorithm 7. It can, however, be straightforwardly adapted for the approximate version of Algorithm 7 described in Subsection 5.4.1.

In what follows, we will refer to the version of Algorithm 7 in which the two proposed enhancements discussed in this section have been implemented as extSDDiP. A detailed description of this algorithm is provided in Appendix 5.8.

5.5 Computational experiments

In this section, we focus on assessing the performance of the extSDDiP algorithm proposed in Sections 5.3 and 5.4. This is done by comparing it with the performance of a stand-alone mathematical programming solver ILOG-CPLEX using the extensive MILP formulation (5.1)-(5.4) and the one of the SDDiP algorithm using the dynamic programming reformulation (5.5)-(5.8) with $\mathcal{G} \equiv \mathcal{S}$.

In what follows, we first describe the scheme used to randomly generate instances of the SULS and the experimental set-up. We then discuss the results of our computational experiments.

5.5.1 Instance Generation

We randomly generated instances following the same procedure as the one used in [45]. This procedure considers various scenario tree structures, several ratios of the production cost to inventory holding cost and several ratios of the set-up cost to the inventory holding cost.

Regarding the scenario tree structure, we used only balanced trees with Σ stages, a constant number $b = |\mathcal{T}^\sigma|$, for all $\sigma \in \mathcal{S}$, of time periods per stage and a constant number $R = R^\sigma$, for all $\sigma \in \mathcal{S}$, of equiprobable realizations per stage. We generated four sets of instances corresponding to four types of scenario tree structures:

- Instances of Set 1 involve a short planning horizon and a small number of stages, but a relatively large number of realizations per stage: $\Sigma = 4$, $b = 1$ and $R \in \{10, 20\}$.
- Instances of Set 2 involve a short planning horizon and a medium number of stages, but a relatively large number of realizations per stage: $\Sigma = 6$, $b = 1$ and $R \in \{10, 20\}$.
- Instances of Set 3 involve a long planning horizon with a medium number of stages and a medium number of realization per stage: $\Sigma = 8$, $b \in \{2, 5\}$ and $R = 5$.
- Instances of Set 4 involve a medium-size planning horizon with a large number of decision stages and a small number of realizations per stage: $\Sigma = 12$, $b = 1$ and $R = 3$.

As for the costs, we used the same numerical values as [45], i.e. a production to holding cost ratio $g/h \in \{2, 4\}$ and a set-up to holding cost ratio $f/h \in \{200, 400\}$. More precisely, the holding cost h^n at node n of the tree was generated from the uniform distribution $U[0, 10]$. The production cost g^n was randomly generated from $U[0.8(g/h)\bar{h}, 1.2(g/h)\bar{h}]$, where $\bar{h} = \sum_{n\in\mathcal{V}} h^n/|\mathcal{V}|$ is the average holding cost. The set-up cost f^n was randomly generated from $U[0.8(f/h)\bar{h}, 1.2(f/h)\bar{h}]$. The demand d^n was generated from the discrete uniform distribution $DU[0, 100]$. The probability ρ^n of node n is given by $\rho^n = (\frac{1}{R})^{\sigma^n - 1}$. Finally, for each set and each considered combination of Σ, b, R, g/h and f/h, five random instances were generated, resulting in a total of 140 instances.

5.5.2 Experimental setup

Each instance is first solved with the mathematical programming solver CPLEX 12.8 using the extensive MILP formulation (5.1)-(5.4). For the Set 1 instances which involve less than 8000 scenarios, we use the cutting-plane generation strategy proposed in [45] to strengthen this formulation by adding violated path inequalities (2.12) and tree inequalities (2.13) at each node of the branch-and-cut search tree. For the other instances, we use the standard branch-and-cut algorithm of solver CPLEX 12.8 using the initial formulation (5.1)-(5.4). Our preliminary experiments namely showed that this was more efficient than using the customized branch-and-cut algorithm based on valid inequalities (2.12) and (2.13) for the large instances. This solution method is denoted by CPX in what follows.

Each instance is then solved by the SDDiP algorithm proposed in [105] and by the extSDDiP algorithm. For both algorithms, the number of scenarios sampled at each iteration is set to $W = 1$. Moreover, the binary approximation of the continuous state variables s^n is carried out as follows. For each instance, we compute an upper bound of the inventory level at node n as $s_{max} = max_{\ell\in\mathcal{L}(0)} d^{0,\ell}$. The number B of binary variables $u^{n,\beta}$ is set to $B = \lceil log_2(s_{max})\rceil$. Note that for the instances considered in our numerical experiments, introducing this binary approximation of the inventory variables will not lead to a sub-optimal solution. Namely, the randomly generated demand vectors only comprise integer components and the optimal leaving inventory at each node is known to take an integer value in this case: see [47].

Regarding the partition of the set of decision stages $\mathcal{S}$ into macro-stages $\mathcal{G}$, we consider only decompositions in which the number of stages per macro-stage, denoted by G, is constant, i.e. $G = |\mathcal{S}(\gamma)|$, for all $\gamma \in \mathcal{G}$. For each instance, depending on the value of Σ, we consider values of $G = \Sigma/\Gamma$ in the set $\{2, 3, 4, 6\}$.

Furthermore, in order to better assess the impact of the two enhancements presented in Section 5.4, several variants of the extSDDiP algorithm are implemented.

First, to evaluate the usefulness of the approximate subtree-based SDDiP algorithm discussed in Subsection 5.4.1, we consider the following three implementations of the extSDDiP algorithm:

- extSDDiP-I corresponds to the case where only the approximate subtree-based SDDiP algorithm based on formulation (5.5)-(5.8) with continuous state variables s^n is run.
- extSDDiP-II corresponds to the case where only Algorithm 7 is run.
- extSDDiP-I/II corresponds to a 2-phase algorithm in which phase I first runs the approximate subtree-based SDDiP algorithm to build an initial approximation of the expected cost-to-go functions and phase II runs Algorithm 7 to further improve these approximations.

Second, we also seek to assess the impact on the algorithmic performance of exploiting alternative MILP formulations of the SULS, as presented in Subsection 5.4.2. Each considered setting is described by the maximum level of formulation strengthening λ_{max} used to strengthen the sub-problem formulation. Thus, extSDDiP-I/II-λ_{max} denotes for instance a 2-phase implementation of the extSDDiP in which the sub-problem formulation strengthening levels $0, \ldots, \lambda_{max}$ are sequentially used following the strategy described at the end of Subsection 5.4.2.

Regarding the stopping criteria, the maximum number of consecutive iterations without any improvement of the lower bound LB is set to 30 and the maximum total number of iterations to 1000. The algorithm stops as soon as one of these two conditions is reached. Note that at this point, the upper bound UB is computed considering only $K = 1$ scenario and thus might not be statistically representative. Thus, after the algorithm has stopped, we compute an updated upper bound based on a larger number of scenarios. For Set 1 instances, a true upper bound is obtained by computing a feasible production plan for the $|\mathcal{L}(0)|$ scenarios. For the other instances, a statistical upper bound is computed as follows. We randomly sample 1000 scenarios and compute a feasible solution for each of them using the final approximation of the expected cost-to-go functions to evaluate the objective function at each (macro)-stage. We then construct a 95% confidence interval and report the right endpoint of this interval as the statistical upper bound of the optimal value.

Each algorithm was implemented in C++ using the Concert Technology environment. All (mixed-integer) linear programs were solved using CPLEX 12.8 with the default settings and the Lagrangian dual problems were solved by a sub-gradient algorithm. All tests were run on the computing infrastructure of the Laboratoire d'Informatique de Paris VI (LIP6), which consists of 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 impose a time limit of 1800 seconds to method CPX to solve each instance. For the SDDiP and extSDDiP algorithms, we impose a time limit of 900 seconds to compute a lower bound and 900 seconds to compute the true or statistical upper bound.

5.5.3 Results

Tables 5.1-5.4 display the numerical results. Columns R and b describe the structure of the scenario tree when needed. The corresponding number of nodes in the scenario tree, $|\mathcal{V}|$, and the number of scenarios, $|\mathcal{L}(0)|$, are then provided. Column G indicates the number of stages per macro-stage in the partial decomposition of the scenario tree and Column `Method` indicates the algorithm used to solve each instance. Each line in the table thus provides the average results of the indicated resolution method over the 20 instances corresponding to the given scenario tree structure with various values of the ratios f/h and g/h. Column `Gap` displays the gap between the lower bound (LB) and the upper bound (UB) found by each method, i.e. $Gap = |UB - LB|/UB$. The average total computation time in seconds is reported in Column `Time (s)`, the average number of iterations in Column `#ite` and the total number of valid inequalities of type (2.12) and (2.13) generated are provided in Column `#VI`.

Results from Table 5.1 first show that, when using the extensive formulation (5.1)-(5.4) strengthened by valid inequalities (2.12) and (2.13), method CPX outperforms the other methods for the smallest considered instances, i.e. the instances corresponding to $\Sigma = 4$, $R = 10$ and $b = 1$, providing an average gap of 1.36% within the allotted time limit. When the number of realizations per stage increases, i.e. for the instances corresponding to $\Sigma = 4$, $R = 20$ and $b = 1$, the relative performance of method CPX deteriorates but the average gap remains below 5%. However, when the number of stages, and consequently the size of the scenario tree, increases, the performance of method CPX strongly deteriorates. This can be seen from the results displayed in Tables 5.2-5.4: method CPX namely provides gaps between 19% for the instances with $\Sigma = 6$, $R = 10$ and $b = 1$ and 94% for the instances with $\Sigma = 6$ and $R = 20$ and $b = 1$.

We also observe from the results displayed in Tables 5.1-5.4 that method SDDiP compares well with method CPX. It namely consistently provides average gaps between 9% and 30% for all instances and significantly outperforms method CPX in terms of solution quality for the largest instances.

Furthermore, these results show that, on average, algorithm extSDDiP significantly outperforms methods CPX and SDDiP. We namely first note that, whatever the partial decomposition and the formulation strengthening setting used, i.e. whatever the value of $G \in \{2, 3, 4, 6\}$ and $\lambda_{max} \in \{0, 1, 2\}$, the gap provided by algorithm extSDDiP-I/II is significantly smaller than the one provided by

Table 5.1: Performance of each method at solving instances from Set 1 ($\Sigma = 4, b = 1$) of the SULS problem

R	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
10	1111	1000	1	SDDiP	9.59	1,100.96	135	0
			2	extSDDiP-I/II-0	4.18	875.47	90	0
				extSDDiP-I/II-1	3.21	869.66	105	823
				extSDDiP-I/II-2	2.81	852.25	140	867
			4	CPX	1.36	1590.96	-	2642
20	8420	8000	1	SDDiP	15.62	1,074.37	77	0
			2	extSDDiP-I/II-0	5.10	967.10	60	0
				extSDDiP-I/II-1	3.26	964.41	79	5,106
				extSDDiP-I/II-2	2.98	961.24	133	7,283
			4	CPX	4.67	1,801.45	-	3224

Table 5.2: Performance of each method at solving instances from Set 3 ($\Sigma = 6, b = 1$) of the SULS problem

R	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
10	111111	100000	1	SDDiP	21.49	1,241.18	80	0
			2	extSDDiP-I/II-0	12.35	1,161.47	46	0
				extSDDiP-I/II-1	7.93	1,155.39	76	1,331
				extSDDiP-I/II-2	6.56	1,047.61	124	1,709
			3	extSDDiP-I/II-0	7.19	1,065.16	36	0
				extSDDiP-I/II-1	4.01	1,084.60	50	9,994
				extSDDiP-I/II-2	4.18	1,100.41	72	22,039
			6	CPX	19.28	1801.78	-	0
20	3.36×10^6	3.2×10^6	1	SDDiP	28.77	1,214.26	46	0
			2	extSDDiP-I/II-0	13.88	1,150.22	38	0
				extSDDiP-I/II-1	8.51	1,127.27	73	3,551
				extSDDiP-I/II-2	7.88	1,050.58	138	11,805
			3	extSDDiP-I/II-0	12.14	1,473.89	9	0
				extSDDiP-I/II-1	12.01	1,526.49	8	996
				extSDDiP-I/II-2	11.96	1,555.29	8	1,093
			6	CPX	94.24	1,801.52	-	0

algorithm SDDiP. In particular, if we consider the results obtained with algorithm extSDDiP-I/II-2 with a partial decomposition using $G = 2$ stages per macro-stage, we obtain an average gap over the 140 instances of 5.07% as compared to an average gap of 19.69% obtained with algorithm SDDiP and an average gap of 41.16% obtained with CPX. Moreover, we would like to point out that, by considering, for each set of instances, the values of G and λ_{max} providing the lowest gap, algorithm extSDDiP-I/II is able to provide a solution within an average gap of 3.76%. This clearly shows that jointly using the partial decomposition of the scenario tree into sub-trees discussed in Section 5.3 and the two algorithmic enhancements presented in Section 5.4 leads to a significant improvement of the performance of the SDDiP algorithm proposed in [105].

We now discuss the individual impact of each of these three elements on the performance of algorithm extSDDiP. The separate impact of the partial decomposition of the scenario tree into sub-

Table 5.3: Performance of each method at solving instances from Set 3 ($\Sigma = 8, R = 5$) of the SULS problem

b	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
2	195312	78125	1	SDDiP	20.08	1662.00	91	0
			2	extSDDiP-I/II-0	13.08	1,669.77	52	0
				extSDDiP-I/II-1	6.83	1,656.55	81	1,024
				extSDDiP-I/II-2	6.21	1,604.36	115	1,121
			4	extSDDiP-I/II-0	5.38	1,380.34	25	0
				extSDDiP-I/II-1	2.89	1,438.71	31	7,762
				extSDDiP-I/II-2	3.22	1,358.63	42	10,118
			8	CPX	43.81	1,802.40	-	0
5	488280	78125	1	SDDiP	13.14	2075.09	70	0
			2	extSDDiP-I/II-0	7.42	1,787.85	42	0
				extSDDiP-I/II-1	2.90	1,591.52	82	4,124
				extSDDiP-I/II-2	2.91	1,588.13	94	4,248
			4	extSDDiP-I/II-0	4.32	1,726.38	16	0
				extSDDiP-I/II-1	2.32	1,863.85	20	23,051
				extSDDiP-I/II-2	1.82	1,841.01	21	26,621
			8	CPX	71.04	1,803.99	-	0

Table 5.4: Performance of each method at solving instances from Set 4 ($\Sigma = 12, b = 1, R = 3$) of the SULS problem

$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
265720	177147	1	SDDiP	29.12	1,716.06	100	0
		2	extSDDiP-I/II-0	16.61	1,742.85	72	0
			extSDDiP-I/II-1	8.65	1,650.81	110	142
			extSDDiP-I/II-2	6.12	1,526.16	175	144
		3	extSDDiP-I/II-0	13.62	1,868.93	41	0
			extSDDiP-I/II-1	6.67	1,746.58	67	687
			extSDDiP-I/II-2	7.01	1,705.17	105	871
		4	extSDDiP-I/II-0	10.46	1,853.51	28	0
			extSDDiP-I/II-1	5.25	1,853.44	47	1,543
			extSDDiP-I/II-2	6.20	1,821.16	72	3,017
		6	extSDDiP-I/II-0	5.28	1,625.98	17	0
			extSDDiP-I/II-1	4.11	1,677.38	21	4,260
			extSDDiP-I/II-2	3.90	1,557.63	29	6,601
		12	CPX	53.78	1,803.29	-	0

trees on the algorithmic performance can be evaluated by looking at the results obtained by algorithm extSDDiP-II-0: see Tables 5.10-5.13 in Appendix 5.9. We thus observe that the average gap can be reduced from 19.68% with algorithm SDDiP to 11.16% with algorithm extSDDiP-II-0 using a partial decomposition involving $G = 2$ stages per macro-stage. Moreover, in general, increasing the value of G further improves the performance of algorithm extSDDiP-II-0: an average gap of 7.50% can thus be observed when using algorithm extSDDiP-II-0 with $G \in \{3, 4, 6\}$. A noticeable exception can be found for the instances corresponding to $\Sigma = 6$, $R = 20$ and $b = 1$ for which the increase of G from 2 to 3 leads to a strong deterioration of the algorithmic performance (see Table 5.11). This might

be explained by the fact that, for these instances, when $G = 3$, each sub-tree involves 421 nodes, among which 400 are leaf nodes. A binary approximation of the leaving inventory level has to be used for each of these 400 nodes so that each sub-problem comprises a rather large number of binary variables and requires a large computational effort to be solved. It is thus not possible to carry out one full iteration of algorithm extSDDiP-II-0 within the allotted computation time.

We now focus on the impact of the first considered enhancement: the introduction of an initial phase based on an approximate sub-tree-based SDDiP algorithm. This impact can be measured by comparing the results obtained with algorithm extSDDiP-II-0 (see Appendix 5.9) with the ones obtained with algorithm extSDDiP-I/II-0. When using a partial decomposition involving $G = 2$ stages per macro-stage, the average gap over the 140 instances is thus reduced from 11.16% with algorithm extSDDiP-II-0 to 10.37% with algorithm extSDDiP-I/II-0. In case larger values of $G \in \{3, 4, 6\}$ are used, a similar reduction of the gap from 7.50% with algorithm extSDDiP-II-0 to 6.60% with algorithm extSDDiP-I/II-0 is observed. Even if the gap reduction seems to be rather limited on average, we note that the use of this initial phase seems to be particularly interesting when the partial decomposition of the scenario tree leads to sub-trees involving a large number of leaf nodes, as it is the case for the instances corresponding to $\Sigma = 6$, $R = 20$ and $b = 1$. Namely, for these instances, when $G = 2$, the average gap is reduced from 17.35% with algorithm extSDDiP-II-0 to 13.88% with algorithm extSDDiP-I/II-0. Furthermore, when $G = 3$, algorithm extSDDiP-I/II-0 provides solutions within an average gap of 12.14% whereas algorithm extSDDiP-II-0 does not provide any feasible solution. The main reason for this is that the approximate sub-tree-based SDDiP algorithm does not require the introduction of additional binary variables for the binary approximation of the state variables. As a consequence, each sub-problem is a small-size MILP which can be solved in a limited computation time. This enables the algorithm to carry out more iterations, to generate more cutting-planes to approximate the expected cost-to-go functions and thus to provide better lower bounds.

To get an assessment of the usefulness of exploiting the alternative MILP formulations available for SULS, we first look at the results provided in Tables 5.10-5.13 in Section 5.9. We thus note that when $G = 2$, the average gap is reduced from 11.16% with algorithm extSDDiP-II-0 to 7.43% with algorithm extSDDiP-II-1. This clearly shows the positive impact of generating strengthened Benders' cut using a linear relaxation of the sub-problems strengthened by path inequalities (2.12). However, for larger values of $G \in \{3, 4, 6\}$, we observe that the average gap is increased from 7.50% with algorithm extSDDiP-II-0 to 8.88% with algorithm extSDDiP-II-1. This deterioration might be explained by the large number of valid inequalities added to the sub-problem formulations: see e.g. the instances corresponding to $\Sigma = 6$, $R = 20$ and $b = 1$ for which more than 20000 path inequalities are added to the various sub-problems. This deterioration might be explained by the fact that adding too many valid inequalities leads to an increase in the size of the MILPs to be solved at each iteration of the extSDDiP algorithm so that fewer iterations can be carried out and weaker lower bounds are provided. Moreover, results from Tables 5.10-5.13 also seem to indicate that the use of additional sub-problem formulation strengthening techniques does not enable algorithm extSDDiP-II-2 to provide better quality solutions.

Thus, each of the two proposed algorithmic enhancements, when used separately, has a moderate positive impact on the solution quality. However, their combined use, in particular the use of alternative MILP formulations of the SULS to generate additional strengthened Benders' cuts in Phase I of the extSDDiP algorithm, seems to significantly improve the algorithmic performance. Namely, when $G = 2$, the average gap is reduced from 10.37% with algorithm extSDDiP-I/II-0 to 5.89% with algorithm extSDDiP-I/II-1 and 5.06% with algorithm extSDDiP-I/II-2. Similarly, when $G \in \{3, 4, 6\}$, the average gap is reduced from 8.34% with algorithm extSDDiP-I/II-0 to 5.32% with algorithm extSDDiP-I/II-1 and 5.47% with algorithm extSDDiP-I/II-2. This improvement might be explained by the fact that, when running algorithm extSDDiP-I/II with the formulation strengthening settings 1 or 2, more iterations of phase I of the algorithm are carried out than when running algorithm

extSDDiP-I/II-0. This enables algorithms extSDDiP-I/II-1 and extSDDiP-I/II-2 to generate more strengthened Benders' cut in Phase I than algorithm extSDDiP-I/II-0 and consequently to obtain a better initial lower bound at the end of phase I. All three algorithms will improve this lower bound during phase II. However, a phase II iteration is more computationally demanding than a phase I iteration so that only a limited number of phase II iterations might be carried out within the allotted time. Thus, using algorithm extSDDiP-I/II-0, does not enable to generate enough cutting planes during phase II to make up for the difference with algorithms extSDDiP-I/II-1 and extSDDiP-I/II-2 in the lower bound value obtained at the end of phase I.

Finally, we would like to point out that the approximate sub-tree based algorithm, i.e. extSDDiP-I, when combined with the use of sub-problem strengthening techniques to generate additional strengthened Benders' cuts, shows a remarkable computational performance. The results provided in Tables 5.6-5.9 (see Appendix 5.9) namely show that using algorithm extSDDiP-I-2 with $G = 2$, enables to provide solutions displaying an average gap of 6.27% within an average computation time of only 166.71s. This suggests that algorithm extSDDiP-I, even if it does not have any theoretical guarantee of convergence, could be used as a heuristic solution approach capable of providing in practice near-optimal solutions in reduced computation times.

5.6 Conclusion and perspectives

We investigated a multi-stage stochastic integer programming approach for the SULS problem and focused on the resolution of instances involving large-size scenario trees. We presented a new extension of the SDDiP algorithm proposed in [105]. This new extension is based on three main features: the partial decomposition of the stochastic problem into smaller stochastic sub-problems (rather than into deterministic sub-problems), the introduction of an initial phase in which the state variables are kept continuous and the exploitation of alternative MILP formulations of the stochastic sub-problems to generate additional strengthened Benders' cuts. Computational experiments carried out on randomly generated instances show that the proposed extended algorithm significantly outperforms the original SDDiP algorithm.

An interesting direction for further research could be to extend this work to single-item single-echelon stochastic lot-sizing problems involving complicating features such as the possibility of backlogging the demand, a limited production capacity or upper bounds on the inventory level. These extensions of the SULS problem would comprise a limited number of continuous state variables at each node and valid inequalities that could be used to obtain alternative MILP formulations. These inequalities are known for most of them (see e.g. [80]). It should thus be possible to adapt the two-phase algorithm extSDDiP-I/II for these problems. It might also be worth investigating whether the proposed extended algorithm could be used to solve multi-item and/or multi-echelon stochastic lot-sizing problems. For such problems, the number of continuous state variables for which a binary approximation would have to be built will be much larger so that the use of the one-phase algorithm extSDDiP-I might be more appropriate.

5.7 Appendix A: Generation of strengthened Benders' cuts using alternative MILP formulations of the sub-problems

This section provides additional insights about the generation of additional strengthened Benders' cuts using alternative MILP formulations of the sub-problems. We present a numerical example illustrating the fact that there is no dominance between the strengthened Benders' cuts generated while using different MILP formulations of the SULS. To do this, we use formulation (5.21)-(5.25) rather than formulation (5.9)-(5.17). It namely has a single continuous state variable at each node, which facilitates the graphic representation of the approximation of the expected cost-to-go function.

Example 1. Consider the following scenario tree involving $\mathcal{S} = 4$ stages and $R^\sigma = 3$ realizations at stages 2 to 4. Each realization is described by its stage σ and its realization index $r \in \{1, 2, 3\}$. The structure of the scenario tree is provided in Figure 5.2 and the corresponding values of the uncertain parameters are provided in Table 5.5.

(σ, ζ)	d	f	g	h
(1,1)	87	934	10	0
(2,1)	69	1182	20	10
(2,2)	38	585	13	2
(2,3)	73	1259	11	1
(3,1)	7	1743	18	5
(3,2)	86	956	20	6
(3,3)	23	108	12	10
(4,1)	14	643	3	6
(4,2)	11	1583	9	0
(4,3)	91	1074	13	10

Table 5.5: Numerical values for Example 1

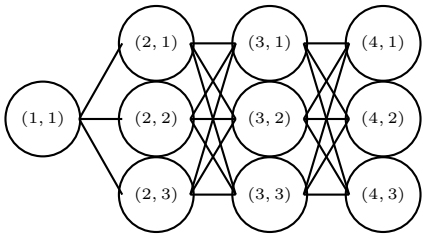


Figure 5.2: Scenario tree structure for Example 1

Let us consider a partial decomposition involving $\Gamma = 2$ macro-stages with $\mathcal{S}(1) = \{1, 2\}$ and $\mathcal{S}(2) = \{3, 4\}$. At macro-stage $\gamma = 1$, we have a single realization $\mathcal{X}^{1,1} = \{(1, 1), (2, 1), (2, 2), (2, 3)\}$. At macro-stage $\gamma = 2$, we have 3 realizations: $\mathcal{X}^{2,1} = \{(3, 1), (4, 1), (4, 2), (4, 3)\}$, $\mathcal{X}^{2,2} = \{(3, 2), (4, 1), (4, 2), (4, 3)\}$, $\mathcal{X}^{2,3} = \{(3, 3), (4, 1), (4, 2), (4, 3)\}$.

At the first iteration of Algorithm 1, as $\psi_1^1 \equiv 0$, the feasible solution obtained at the end of the forward step is such that the leaving inventory at nodes $(2, 1)$, $(2, 2)$ and $(2, 3)$ is equal to 0. The strengthened Benders' cuts generated during the backward step of the first iteration of Algorithm 1 to approximate $\mathcal{Q}^1(s^\ell)$ are provided below:

- by solving the LP relaxation of sub-problems $\tilde{P}_1^2(0, 0, \mathcal{X}^{2,\zeta}, IF(\emptyset))$, $r \in \{1, 2, 3\}$:

$$\theta^1 \geq 682.18 - 8.24 s^\ell \tag{5.31}$$

- by solving the LP relaxation of sub-problems $\tilde{P}_1^2(0, 0, \mathcal{X}^{2,\zeta}, IF(\phi_1^{2,\zeta}))$, $r \in \{1, 2, 3\}$, in which $\phi_1^{2,\zeta}$ contains only path inequalities (2.12):

$$\theta^1 \geq 882.55 - 25.40 s^\ell \tag{5.32}$$

- by solving the LP relaxation of sub-problems $\tilde{P}_1^2(0, 0, \mathcal{X}^{2,\zeta}, IF(\phi_1^{2,\zeta}))$, $r \in \{1, 2, 3\}$, in which $\phi_1^{2,\zeta}$ contains both path inequalities (2.12) and tree inequalities (2.13):

$$\theta^1 \geq 887.88 - 26.18 s^\ell \tag{5.33}$$

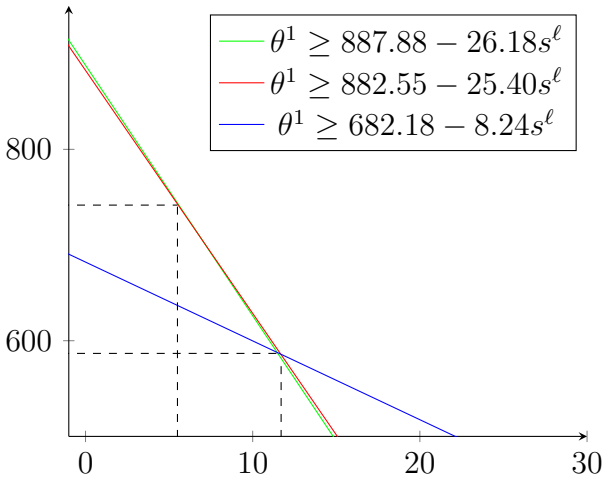


Figure 5.3: Illustration for Example 1.

Figure 1 provides a graphic representation of the three generated cuts. We observe that there is no dominance amongst them, i.e. none of the cuts seems to provide a better approximation of the expected cost-to-go function for all possible value of the leaving inventory level s^ℓ. More specifically, when s^ℓ lies in the interval $[1, 7]$, the best approximation is obtained by the cut generated using formulation $IF(\phi_i^{\gamma,\zeta})$ strengthened by inequalities (2.12) and (2.13). When s^ℓ lies in $[7, 12]$, the best approximation is obtained by the cut generated using formulation $IF(\phi_i^{\gamma,\zeta})$ strengthened only by inequalities (2.12). Finally, when s^ℓ is greater than 12, the best approximation is obtained by the cut generated using formulation $IF(\emptyset)$. This shows the practical interest of generated strengthened Benders'cuts based on alternative MILP formulation of the sub-problems.

5.8 Appendix B: Detailed description of the proposed enhanced sub-tree-based SDDiP algorithm

Algorithm 8: Strengthening sub-problems algorithm

if $\lambda = 0$ **then**
  $F = IF(\emptyset))$
else if $\lambda = 1$ **then**
  Run the cutting plane procedure to generate path inequalities (2.12)
  Add the generated inequalities to $\phi_v^{\gamma,\zeta}$ to get $\phi_{v+1}^{\gamma,\zeta}$
  $F = IF(\phi_v^{\gamma,\zeta}))$
else if $\lambda = 2$ **then**
  Run the cutting plane procedure to generate tree inequalities (2.13)
  Add the generated inequalities to $\phi_v^{\gamma,\zeta}$ to get $\phi_{v+1}^{\gamma,\zeta}$
  $F = IF(\phi_v^{\gamma,\zeta}))$

Algorithm 9: Approximate extSDDiP algorithm

while *no stopping criterion is satisfied* **do**
  Randomly select W scenarios $\Omega_v = \{\omega_v^1, ..., \omega_v^W\}$
  for $w = 1, ..., W$ **do**
    for $\gamma = 1, ..., \Gamma$ **do**
      Solve $\tilde{P}_v^{\gamma}(s_v^m, \tilde{\psi}_v^{\gamma}, \mathcal{X}^{\gamma,\zeta_v^{k,\gamma}}, IF(\phi_v^{\gamma,\zeta}))$ for $m = \omega_v^w \cap \mathcal{V}^{t'(\gamma-1)}$
      Record s_v^{ℓ} for $\ell = \omega_v^w \cap \mathfrak{L}(\gamma, \zeta_v^{w,\gamma})$
    end
  end
  for $\gamma = \Gamma - 1, ..., 1$ **do**
    for $w = 1, ..., W$ **do**
      for $\zeta \in \mathcal{R}^{\gamma+1}$ **do**
        Let $m = \omega_v^w \cap \mathcal{V}^{t'(\gamma)}$
        Run Algorithm 8
        Solve the Linear relaxation of $\tilde{P}_v^{\gamma+1}(s_v^m, \tilde{\psi}_{v+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta}, F(\phi_{v+1}^{\gamma,\zeta}))$ and collect the coefficients of the strengthened Benders' cut
        Solve the Lagrangian relaxation of $\tilde{P}_v^{\gamma+1}(s_v^m, \tilde{\psi}_{v+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta}, F(\phi_{v+1}^{\gamma,\zeta}))$ and collect the constant value of the strengthened Benders' cut
      end
      Add the generated cut to $\tilde{\psi}_v^{\gamma}$ to get $\tilde{\psi}_{v+1}^{\gamma}$
    end
    if *a criterion for Algorithm 8 is satisfied* **then**
      $\lambda \leftarrow \lambda + 1$
    end
    $LB \leftarrow \tilde{\mathscr{Q}}_{v+1}^{1,1}(0)$
    $v \leftarrow v + 1$
  end
end

Algorithm 10: extSDDiP algorithm

Initialize $LB \leftarrow -\infty, UB \leftarrow +\infty, v \leftarrow 1, \lambda \leftarrow 0$
for $\gamma = \Gamma - 1, ..., 1$ **do**
  for $\zeta \in \mathcal{R}^{\gamma+1}$ **do**
    Initialize $\phi^{\gamma,\zeta} \leftarrow \emptyset$
  end
end
Run Algorithm 9
while *no stopping criterion is satisfied* **do**
  Randomly select W scenarios $\Omega_v = \{\omega_v^1, ..., \omega_v^W\}$
  for $w = 1, ..., W$ **do**
    for $\gamma = 1, ..., \Gamma$ **do**
      Solve $\hat{P}_v^{\gamma}(u_v^m, \psi_v^{\gamma}, \mathcal{X}^{\gamma,\zeta_v^{w,\gamma}}, IF(\phi_v^{\gamma,\zeta}))$ for $m = \omega_v^k \cap \mathcal{V}^{t'(\gamma-1)}$.
      Record u_v^{ℓ} for $\ell = \omega_v^w \cap \mathfrak{L}(\gamma, \zeta_v^{w,\gamma})$
    end
    $\upsilon^w \leftarrow \sum_{n \in \omega_v^w} (f^n y_v^n + h^n s_v^n + g^n x_v^n)$
  end
  $\hat{\mu} \leftarrow \sum_{w=1}^{W} \upsilon^w$ and $\hat{\sigma}^2 \leftarrow \frac{1}{W-1} \sum_{w=1}^{W} (\upsilon^w - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2} \frac{\hat{\sigma}}{\sqrt{W}}$
  for $\gamma = \Gamma - 1, ..., 1$ **do**
    for $w = 1, ..., W$ **do**
      for $\zeta \in \mathcal{R}^{\gamma+1}$ **do**
        -Let $m = \omega_v^w \cap \mathcal{V}^{t'(\gamma)}$
        -Run Algorithm 8
        -Solve the Linear relaxation of $\hat{P}_v^{\gamma+1}(u_v^m, \psi_{v+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta}, F(\phi_{v+1}^{\gamma,\zeta}))$ and collect the coefficients of the strengthened Benders' cut
        -Solve the Lagrangian relaxation of $\hat{P}_v^{\gamma+1}(u_v^m, \psi_{v+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta}, F(\phi_{v+1}^{\gamma,\zeta}))$ and collect the constant value of the strengthened Benders' cut
        -Solve $\hat{P}_v^{\gamma+1}(u_v^m, \psi_{v+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta}, F(\phi_{v+1}^{\gamma,\zeta}))$ and collect the coefficients of the Integer optimality cut
        -Solve the Lagrangian dual problem and collect the coefficients of the Lagrangian cut
      end
      Add the 3 generated cuts to ψ_v^{γ} to get ψ_{v+1}^{γ}
    end
    if *a criterion for Algorithm 8 is satisfied* **then**
      $\lambda \leftarrow \lambda + 1$
    end
    $LB \leftarrow \hat{Q}_{v+1}^{1,1}(0)$
    $v \leftarrow v + 1$
  end
end

5.9 Appendix C: Additional computational experiments

This section provides the results of additional computational experiments carried out in order to assess the individual impact of each of the three main elements of algorithm extSDDiP, namely the partial decomposition of the stochastic problem into smaller stochastic sub-problems, the introduction of an initial phase in which the state variables are kept continuous and the exploitation of alternative MILP formulations of the stochastic sub-problems to generate additional strengthened Benders's cuts, on its overall performance.

Tables 5.6-5.9 provide the results obtained while running extSDDiP-I, i.e. running only the approximate sub-tree based algorithm. Tables 5.10-5.13 provide the results obtained while running extSDDiP-II, i.e. running only Algorithm 1 without an initial phase based on the approximate sub-tree based algorithm.

Table 5.6: Performance of the approximate algorithm at solving instances from Set 1 ($\Sigma = 4, b = 1$) of the SULS problem

R	$\lvert\mathcal{V}\rvert$	# Scen	G	Method	Gap	Time (s)	# ite	# VI
10	1111	1000	1	SDDiP	9.59	1,100.96	135	0
			2	extSDDiP-I-0	9.44	7.42	16	0
				extSDDiP-I-1	6.55	12.56	33	92
				extSDDiP-I-2	5.61	21.26	64	678
			4	CPX	1.36	1590	-	2642
20	8420	8000	1	SDDiP	15.62	1,074.37	77	0
			2	extSDDiP-I-0	8.28	25.40	18	0
				extSDDiP-I-1	5.25	36.50	35	377
				extSDDiP-I-2	4.43	75.93	86	5,545
			4	CPX	4.67	1,801	-	3224

Table 5.7: Performance of the approximate algorithm at solving instances from Set 2 ($\Sigma = 6, b = 1$) of the SULS problem

R	$\lvert\mathcal{V}\rvert$	# Scen	G	Method	Gap	Time (s)	# ite	# VI
10	111111	100000	1	SDDiP	21.49	1,241.18	80	0
			2	extSDDiP-I-0	13.91	23.22	23	0
				extSDDiP-I-1	8.93	44.99	54	286
				extSDDiP-I-2	6.73	104.13	101	1,563
			3	extSDDiP-I-0	7.66	146.98	18	0
				extSDDiP-I-1	4.85	288.06	38	1,886
				extSDDiP-I-2	4.56	787.07	71	20,829
			6	CPX	19.28	1801	-	0
20	3.36×10^6	3.2×10^6	1	SDDiP	28.77	1,214.26	46	0
			2	extSDDiP-I-0	15.47	71.33	30	0
				extSDDiP-I-1	8.88	198.51	67	1,132
				extSDDiP-I-2	7.95	621.07	137	11,762
			3	extSDDiP-I-0	10.67	1,349.91	8	0
				extSDDiP-I-1	10.75	1,369.64	7	0
				extSDDiP-I-2	11.81	1,386.40	7	0
			6	CPX	94.24	1,801	-	0

Table 5.8: Performance of the approximate algorithm at solving instances from Set 3 ($\Sigma = 8, R = 5$) of the SULS

b	$\lvert\mathcal{V}\rvert$	# Scen	G	Method	Gap	Time (s)	# ite	# VI
2	195312	78125	1	SDDiP	20.08	1662.00	91	0
			2	extSDDiP-I-0	15.16	40.83	21	0
				extSDDiP-I-1	6.61	55.23	52	432
				extSDDiP-I-2	6.30	76.85	84	1,002
			4	extSDDiP-I-0	7.04	676.63	13	0
				extSDDiP-I-1	3.49	1,011.61	28	6,008
				extSDDiP-I-2	3.12	1,285.02	32	7,975
			8	CPX	43.81	1,802.40	-	0
5	488280	78125	1	SDDiP	13.14	2075.09	70	0
			2	extSDDiP-I-0	13.73	113.42	21	0
				extSDDiP-I-1	2.56	130.48	49	2,196
				extSDDiP-I-2	2.58	163.25	65	3,746
			4	extSDDiP-I-0	6.39	1,179.24	12	0
				extSDDiP-I-1	3.12	1,739.40	18	19,857
				extSDDiP-I-2	2.89	1,765.80	19	21,455
			8	CPX	71.04	1,803.99	-	0

Table 5.9: Performance of the approximate algorithm at solving instances from Set 4 ($\Sigma = 12, R = 3, b = 1$) of the SULS

$\lvert\mathcal{V}\rvert$	# Scen	G	Method	Gap	Time (s)	# ite	# VI
265720	177147	1	SDDiP	29.12	1,716.06	100	0
		2	extSDDiP-I-0	24.01	73.32	20	0
			extSDDiP-I-1	11.15	88.51	53	86
			extSDDiP-I-2	10.32	104.50	81	137
		3	extSDDiP-I-0	15.78	23.95	19	0
			extSDDiP-I-1	7.87	36.14	47	235
			extSDDiP-I-2	6.43	59.33	84	775
		4	extSDDiP-I-0	12.82	74.63	17	0
			extSDDiP-I-1	4.84	99.06	38	627
			extSDDiP-I-2	4.80	196.46	64	2,535
		6	extSDDiP-I-0	7.72	1,009.62	10	0
			extSDDiP-I-1	4.25	1,275.45	18	3,323
			extSDDiP-I-2	4.11	1,361.15	23	4,747
		12	CPX	53.78	1,803.29	-	0

Table 5.10: Performance of Algorithm 7 at solving instances from Set 1 ($\Sigma = 4, b = 1$) of the SULS problem

R	$\lvert\mathcal{V}\rvert$	$\#Scen$	G	Method	Gap	Time (s)	# ite	# VI
10	1111	1000	1	SDDiP	9.59	1,100.96	135	0
			2	extSDDiP-II-0	4.86	951.89	74	0
				extSDDiP-II-1	3.00	908.93	77	879
				extSDDiP-II-2	3.16	928.67	72	875
			4	CPX	1.36	1590	-	2642
20	8420	8000	1	SDDiP	15.62	1,074.37	77	0
				extSDDiP-II-0	5.54	971.92	42	0
				extSDDiP-II-1	3.58	961.21	41	5,075
				extSDDiP-II-2	3.70	958.20	41	5,068
			4	CPX	4.67	1,801	-	3224

Table 5.11: Performance of Algorithm 7 at solving instances from Set 2 ($\Sigma = 6, b = 1$) of the SULS problem

R	$\lvert\mathcal{V}\rvert$	# Scen	G	Method	Gap	Time (s)	# ite	# VI
10	111111	100000	1	SDDiP	21.49	1,241.18	80	0
			2	extSDDiP-II-0	12.42	1,182.40	25	0
				extSDDiP-II-1	9.21	1,187.84	25	1,400
				extSDDiP-II-2	8.84	1,176.92	25	1,389
			3	extSDDiP-II-0	7.68	1,086.78	21	0
				extSDDiP-II-1	6.11	1,179.83	15	10,557
				extSDDiP-II-2	5.27	1,167.99	15	10,489
			6	CPX	19.28	1801	-	0
20	3.36×10^6	3.2×10^6	1	SDDiP	28.77	1,214.26	46	0
			2	extSDDiP-II-0	17.35	1,118.36	11	0
				extSDDiP-II-1	17.50	1,274.20	10	4,081
				extSDDiP-II-2	16.85	1,268.43	9	3,957
			3	extSDDiP-II-0	-	-	-	-
				extSDDiP-II-1	-	-	-	-
				extSDDiP-II-2	-	-	-	-
			6	CPX	94.24	1,801	-	0

Table 5.12: Performance of Algorithm 7 at solving instances from Set 3 ($\Sigma = 8, R = 5$) of the SULS problem

b	$\lvert\mathcal{V}\rvert$	# Scen	G	Method	Gap	Time (s)	# ite	# VI
2	195312	78125	1	SDDiP	20.08	1662.00	91	0
			2	extSDDiP-II-0	13.42	1,653.96	33	0
				extSDDiP-II-1	6.93	1,665.12	37	1,106
				extSDDiP-II-2	7.21	1,667.20	35	1,101
			4	extSDDiP-II-0	6.91	1,475.52	13	0
				extSDDiP-II-1	9.29	1,645.23	9	10,933
				extSDDiP-II-2	8.80	1,632.74	9	11,075
			8	CPX	43.81	1,802.40	-	0
5	488280	78125	1	SDDiP	13.14	2075.09	70	0
			2	extSDDiP-II-0	7.76	1,780.95	29	0
				extSDDiP-II-1	3.66	1,730.56	35	4,815
				extSDDiP-II-2	3.53	1,703.66	37	4,831
			4	extSDDiP-II-0	6.41	1,629.26	10	0
				extSDDiP-II-1	24.18	1,864.58	4	20,673
				extSDDiP-II-2	26.63	1,862.54	4	18,764
			8	CPX	71.04	1,803.99	-	0

Table 5.13: Performance of Algorithm 7 at solving instances from Set 4 ($\Sigma = 12, R = 3, b = 2$) of the SULS problem

$\lvert\mathcal{V}\rvert$	# Scen	G	Method	Gap	Time (s)	# ite	# VI
265720	177147	1	SDDiP	29.12	1,716.06	100	0
		2	extSDDiP-II-0	16.32	1,693.26	60	0
			extSDDiP-II-1	8.11	1,618.20	73	138
			extSDDiP-II-2	8.32	1,651.21	72	138
		3	extSDDiP-II-0	13.67	1,725.26	25	0
			extSDDiP-II-1	7.76	1,726.47	27	741
			extSDDiP-II-2	7.79	1,712.67	27	740
		4	extSDDiP-II-0	11.67	1,853.11	13	0
			extSDDiP-II-1	8.88	1,852.27	12	1,662
			extSDDiP-II-2	8.78	1,865.50	12	1,657
		6	extSDDiP-II-0	6.18	1,601.00	12	0
			extSDDiP-II-1	6.28	1,760.09	10	7,264
			extSDDiP-II-2	5.51	1,728.80	9	7,134
		12	CPX	53.78	1,803.29	-	0

Chapter 6

Multi-echelon stochastic lot-sizing with remanufacturing and lost sales: a dual dynamic decomposition approach

This chapter comes back to the initial stochastic remanufacturing planning problem and investigates the development of a decomposition-based approach for this problem. More precisely, we consider the multi-echelon stochastic lot-sizing problem with remanufacturing and lost sales introduced in Chapter 3 and adapt the extended SDDiP algorithm proposed in Chapter 5 to this problem. This adaptation mainly aims at reducing the computational burden linked to the resolution of the sub-problems needed to build the approximation of the expected cost-to-go functions in the backward step of the algorithm. When considering the stochastic remanufacturing planning problem, these sub-problems are namely medium-size mixed-integer programs so that their resolution by a mathematical solver requires a non negligible time. Some algorithmic enhancements are explored to handle this issue. Our computational experiments carried out on large-size randomly generated instances suggest that the proposed approximate extSDDiP algorithm is capable of obtaining near-optimal solutions in practicable computation times and significantly outperforms both the mathematical programming solver CPLEX 12.8 using an extensive MILP formulation and the initial SDDiP algorithm proposed in [105].

6.1 Introduction

Chapter 5 presented a new decomposition algorithm combining the SDDiP algorithm with a polyhedral approach to solve a multi-stage stochastic programming extension of the simplest available lot-sizing model, i.e. the single-item uncapacitated lot-sizing model or ULS. As mentioned in Section 5.6, an interesting research direction would be to study how the proposed algorithm might be used to solve more general variants of lot-sizing problems involving e.g. multiple items and multiple production echelons.

As a first step in this direction, we propose in the present chapter to apply the extSDDiP algorithm investigated in Chapter 5 to solve the stochastic multi-item multi-echelon lot-sizing problem studied in Chapter 3 to build production planning for a remanufacturing system. We refer the reader to Chapter 3 for a detailed description of the problem and its mixed-integer linear programming formulation.

We focus here on adapting the extSDDiP algorithm to solve this multi-echelon lot-sizing problem. This adaptation mainly aims at reducing the computational burden linked to the resolution of the sub-problems needed to build the approximation of the expected cost-to-go functions in the backward step of the algorithm. Many of these sub-problems are namely medium-size mixed-integer programs so that their resolution by a mathematical solver requires a non negligible time. The proposed

adaptation thus relies on three key elements.

First, we only use the approximate variant of the sub-tree based SDDiP algorithm which relies on continuous state variables, i.e. the extSDDiP-I variant of the algorithm discussed in Chapter 5. Namely, the combination of a partial decomposition of the scenario tree into sub-trees and of the presence of multiple items leads to a significant increase in the number of continuous state variables (corresponding to inventory level variables) for which a binary approximation would have to be built. This would thus lead to prohibitive computation times to carry out each iteration of the extSDDiP algorithm.

Second, we use the concept of ε-optimal cuts recently introduced in [84] to approximate recourse functions in the context of a Benders decomposition algorithm. Basically, an ε-optimal cut is a cut obtained by solving each sub-problem to near optimality, i.e. a cut built by using a feasible solution of each sub-problem whose objective value is at most at ε units of the optimal value. We thus investigate the use of ε-optimal strengthened Benders' cuts, i.e., the generation of strengthened Benders' cuts obtained by using a solution of each Lagrangian sub-problem which is at most ε units of the optimal value. Note that the use of ε-optimal strengthened Benders' cuts is motivated from an algorithmic standpoint by the fact that their generation might require a significantly reduced computational effort as compared to the one needed to generate 'classical' strengthened Benders' cuts. Hence, these cuts may be useful as they may enable the extSDDiP algorithm to build a good approximation of the expected cost-to-go functions with a reasonable computational effort.

Third, similarly to Chapter 5, we consider three alternative MILP formulations of the problem in order to generate a wider set of ϵ-optimal strengthened Benders' cuts in the backward step of the extSDDiP algorithm, namely the initial MILP formulation (3.12)-(3.25), the MILP formulation (3.12)-(3.25) strengthened by the (k, U) path inequalities (3.38)-(3.39) discussed in Chapter 3 and the MILP formulation (3.12)-(3.25) strengthened by the (ℓ, k, U) valid inequalities (4.43),(4.44) and (4.46) introduced in Chapter 4.

Note that, to the best of our knowledge, this is the first attempt at developing a dynamic programming decomposition approach for a multi-item multi-echelon stochastic lot-sizing problem.

The remaining part of this chapter is organized as follows. Section 6.2 introduces a new dynamic programming formulation for the problem under study. Section 6.3 presents the sub-tree-based SDDiP algorithm proposed in Chapter 5 as applied to the problem under study. The proposed adaptation of this algorithm is then presented in Section 6.4. It combines an approximate version of the extSDDiP algorithm based on continuous state variables with the generation of ε-optimal strengthened Benders' cuts relying on three alternative MILP formulations of the sub-problems. Computational experiments are reported in Section 6.5. Finally, conclusions and directions for further works are discussed in Section 6.6.

6.2 Dynamic programming formulation

In Chapter 3, we introduced a multi-echelon remanufacturing system and investigated a stochastic lot-sizing problem aiming at planning production on this system under uncertain input data. We proposed to use a multi-stage stochastic programming approach in which the evolution of the uncertain parameters is represented by a scenario tree $\mathcal{V}$. Based on this scenario tree representation, we formulated the problem as a deterministic equivalent MILP: see formulation (3.12)-(3.25) presented in Section 3.3, page 34. We then proposed a branch-and-cut algorithm based on new path and tree valid inequalities (see Section 3.4) to solve this problem. Although the computational results provided in Section 3.6 showed that the proposed algorithms were efficient at solving medium-size instances of the problem, numerical difficulties were encountered to solve instances involving large-size scenario trees.

In order to be able to provide good-quality solutions for larger instances, we investigate in what follows an alternative solution approach relying on a partial decomposition of the problem into a

series of smaller stochastic sub-problems linked together by dynamic programming equations.

The starting point of this solution approach is a partial decomposition of the scenario tree into a set of smaller sub-trees. We recall here the notation introduced in Section 5.2.2 to describe this partial decomposition. We first partition the set of decision stages $\mathcal{S} = \{1, \ldots, \Sigma\}$ into a series of macro-stages $\mathcal{G} = \{1, \ldots, \Gamma\}$, where each macro-stage $\gamma \in \mathcal{G}$ contains a number of consecutive stages denoted $\mathcal{S}(\gamma)$. We let $t(\gamma)$ (resp. $t'(\gamma)$) represent the first (resp. the last) time period belonging to macro-stage γ. Using the set of macro-stages $\mathcal{G}$ defined above, we can decompose the scenario tree $\mathcal{V}$ into a series of smaller sub-trees as follows. For a given macro-stage γ, each node η belonging to the first time period in γ, i.e. each node $\eta \in \mathcal{V}^{t(\gamma)}$, is the root node of a sub-tree defined by the set of nodes $\mathcal{W}^\eta = \cup_{t=t(\gamma),\ldots,t'(\gamma)} \mathcal{V}^t \cap \mathcal{V}(\eta)$. We recall that $\mathcal{V}(\eta)$ is the sub-tree of $\mathcal{V}$ rooted in η, $\mathcal{W}^\eta$ is thus the restriction of $\mathcal{V}(\eta)$ to the nodes belonging to macro-stage γ. Let $\mathfrak{L}(\eta) = \mathcal{W}^\eta \cap \mathcal{V}^{t'(\gamma)}$ be the set of leaf nodes of sub-tree $\mathcal{W}^\eta$. Finally, we denote as $\mho = \cup_{\gamma \in \mathcal{G}} \mathcal{V}^{t(\gamma)}$ the set of sub-tree root nodes induced by $\mathcal{G}$. We refer the reader to Figure 5.1 in page 87 for an illustration of these notations on a small scenario tree.

The sub-problem P^η related to node $\eta \in \mho$ thus focuses on defining the production plan for the nodes belonging to sub-tree $\mathcal{W}^\eta$ based on the entering stock level of each item i imposed by its parent node a^η in the scenario tree. This entering stock is described by $(S_0^{a^\eta}, E^{a^\eta})$ where $S_0^{a^\eta}$ denotes the natural inventory level of the used product indexed by $i = 0$ and $E^{a^\eta} = (E_1^{a^\eta}, ..., E_I^{a^\eta}, ..., E_{2I+1}^{a^\eta})$ is a vector describing the echelon inventory level for each item $i \in \mathcal{I} \setminus \{0\}$. P^η can be formulated as follows.

$$\mathscr{Q}^\eta(S_0^{a^\eta}, E^{a^\eta}) = \min \sum_{n \in \mathcal{W}^\eta} \rho^n \Big(\sum_{p \in \mathcal{J}} f_p^n Y_p^n + h_0^n S_0^n + \sum_{i \in \mathcal{I} \setminus \{0\}} eh_i^n E_i^n + l^n L^n + \sum_{i \in \mathcal{I}_r \cup \{0\}} q_i^n Q_i^n + g^n X_0^n \Big) + \sum_{\ell \in \mathfrak{L}(\eta)} \sum_{m \in \mathcal{C}(\ell)} \mathscr{Q}^m(S_0^\ell, E^\ell) \quad (6.1)$$

$$X_p^n \leq M_p^n Y_p^n \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{W}^\eta \quad (6.2)$$

$$S_0^n = S_0^{a^n} + r^n - X_0^n - Q_0^n \qquad \forall n \in \mathcal{W}^\eta \quad (6.3)$$

$$E_i^n = E_i^{a^n} + \delta_i^n \varpi_i X_0^n - \varpi_i d^n + \varpi_i L^n - Q_i^n \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{W}^\eta \quad (6.4)$$

$$E_i^n = E_i^{a^n} + X_{i-I}^n - \varpi_i d^n + \varpi_i L^n \qquad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{W}^\eta \quad (6.5)$$

$$E_{2I+1}^n = E_{2I+1}^{a^n} + X_{I+1}^n - d^n + L^n \qquad \forall n \in \mathcal{W}^\eta \quad (6.6)$$

$$E_i^n - E_{I+i}^n \geq 0 \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{W}^\eta \quad (6.7)$$

$$E_i^n - \varpi_i E_{2I+1}^n \geq 0 \qquad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{W}^\eta \quad (6.8)$$

$$E_{2I+1}^n \geq 0 \qquad \forall n \in \mathcal{W}^\eta \quad (6.9)$$

$$S_0^n \geq 0, L^n \geq 0 \qquad \forall n \in \mathcal{W}^\eta \quad (6.10)$$

$$X_p^n \geq 0, Y_p^n \in \{0, 1\} \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{W}^\eta \quad (6.11)$$

The objective function (6.1) comprises two terms: a term related to the expected production costs over sub-tree $\mathcal{W}^\eta$ and a term representing the future expected production costs incurred by the decisions made over sub-tree $\mathcal{W}^\eta$. More precisely, in (6.1), $\mathscr{Q}^\eta(S_0^{a^\eta}, E^{a^\eta})$ denotes the optimal value of sub-problem P^η as a function of the entering stock level $(S_0^{a^\eta}, E^{a^\eta})$ and $\mathscr{Q}^m(S_0^\ell, E^\ell)$ the optimal value of sub-problem P^m as a function of the entering stock level (S_0^ℓ, E^ℓ). The expected cost-to-go function at node $\ell \in \mathfrak{L}(\eta)$ is defined as the expected value of $\mathscr{Q}^m(\cdot)$ over all the children of ℓ in the initial scenario tree $\mathcal{V}$, i.e. over all $m \in \mathcal{C}(\ell)$, which gives $\mathcal{Q}^\ell(\cdot) = \sum_{m \in \mathcal{C}(\ell)} \mathscr{Q}^m(\cdot)$. The expected future costs of the decisions made in $\mathcal{W}^\eta$ are thus computed as the sum, over all nodes $\ell \in \mathfrak{L}(\eta)$, of $\mathcal{Q}^\ell(S_0^\ell, E^\ell)$. Note that for all leaf nodes, i.e. for all $\ell \in \mathcal{V}^T$, $\mathcal{Q}^\ell(\cdot) \equiv 0$.

We note that in case of $\mathcal{G} \equiv \mathcal{S}$, i.e. in case each macro-stage corresponds to a single initial decision stage, each sub-tree $\mathcal{W}^\eta$ reduces to a set of nodes belonging to a single deterministic scenario involving T^{σ^η} periods and we obtain a decomposition similar to the one used by [105].

6.3 Partial decomposition approach

We now investigate how the extSDDiP algorithm presented in Chapter 5 can be applied to solve the stochastic remanufacturing planning problem.

Recall that the SDDiP and extSDDiP algorithms rely on the key assumption that the scenario tree displays the stage-wise independence property. When there are several time periods per decision stage, this property can be defined as follows. For any two nodes m and m' belonging to stage $\sigma - 1$ and such that $t^m = t^{m'} = \max\{t, t \in \mathcal{T}^{\sigma-1}\}$, the set of nodes $\cup_{t\in\mathcal{T}^\sigma}\mathcal{V}^t \cap \mathcal{V}(m)$ and $\cup_{t\in\mathcal{T}^\sigma}\mathcal{V}^t \cap \mathcal{V}(m')$ are defined by identical data and conditional probabilities.

This stage-wise independence property enables us to significantly reduce the number of expected cost-to-go functions for which a piece-wise linear approximation must be build. Namely, as explained in Section 5.3, in this case, the stochastic process can be represented at macro-stage γ by a set $\mathcal{R}^\gamma = \{1, \dots, R^\gamma\}$ of independent realizations. Each realization $\mathcal{X}^{\gamma,\zeta}$ corresponds to a subtree describing one of the possible evolutions of the uncertain parameters over periods $t(\gamma), \dots, t'(\gamma)$. Let $\xi^{\gamma,\zeta}$ denote the root node of $\mathcal{X}^{\gamma,\zeta}$ and $\mathfrak{L}(\gamma,\zeta)$ denote the set of its leaf nodes. The expected cost-to-go functions thus depend on the macro-stage rather than on the node, i.e. we have $\mathcal{Q}^m(\cdot) \equiv \mathcal{Q}^\gamma(\cdot)$, for all $m \in \mathcal{V}^{t'(\gamma)}$, so that only one expected cost-to-go function has to be approximated per macro-stage. Moreover, we can define a single sub-problem P^γ per macro-stage and each sub-problem $P^\eta, \eta \in \mho$, will be described as $P^{\gamma^\eta}(S_0^{a^\eta}, E^{a^\eta}, \mathcal{X}^{\gamma^\eta,\zeta})$ where $\mathcal{X}^{\gamma^\eta,\zeta}$ is the realization corresponding to $\mathcal{W}^\eta$.

6.3.1 Sub-problem reformulation

Similar to what is done in Section 5.3.1 for the SULS, we modify the initial dynamic programming formulation (6.1)-(6.11) in order to be able to apply the extSDDiP algorithm.

As this algorithm requires the state variables to be binary, we first carry out a binary approximation of the state variables (S_0^n, E^n) . This binary approximation is obtained by replacing the continuous variable E_i^n by a set of binary variables $U_i^{n,\beta}$ such that $E_i^n = \sum_{\beta\in\mathcal{B}} 2^\beta U_i^{n,\beta}$ for each $i \in \mathcal{I} \setminus \{0\}$, where $\mathcal{B} = \{1, \dots, B\}$. We have $U_i^{n,\beta} = 1$ if coefficient 2^β is used to compute the value of E_i^n and $U_i^{n,\beta} = 0$ otherwise. Similarly, binary variables $U_0^{n,\beta}$ are introduced in order to build a binary approximation of variables S_0^i. We note however that this binary approximation is not needed for all inventory variables, but only for those coupling the sub-problems $P^\gamma(\cdot,\cdot)$, to one another. Thus, in sub-problem $P^\gamma(S_0^m, E^m, \mathcal{X}^{\gamma,\zeta})$, we use a binary approximation for the entering stock (S_0^m, E^m) at root node $\xi^{\gamma,\zeta}$ and for the leaving stock (S_0^ℓ, E^ℓ) at each leaf node $\ell \in \mathfrak{L}(\gamma,\zeta)$.

Then, as indicated by [105], we introduce local copies of the binary state variables relative to root node $\xi^{\gamma,\zeta}$. More precisely, $\hat{U}_i^{\xi^{\gamma,\zeta},\beta}$ is an auxiliary continuous decision variable representing the value of the state variable $U_i^{m,\beta}$ at the parent node m. It is thus a local copy in problem $P^\gamma(S_0^m, E^m, \mathcal{X}^{\gamma,\zeta})$ of the state variable $U_i^{m,\beta}$, the value of which is considered as a given input parameter for this problem. Note that, in the backward step of the SDDiP algorithm, the corresponding constraints of these local copies variables will be dualized in order to generate cuts to approximate the expected cost-to-go functions.

This leads to the following reformulation of sub-problem $P^\gamma(U^m, \mathcal{X}^{\gamma,\zeta})$:

$$\mathscr{Q}^{\gamma,\zeta}(U^m) = \min \sum_{n\in\mathcal{X}^{\gamma,\zeta}} \rho^n \Big(\sum_{p\in\mathcal{J}} f_p^n Y_p^n + h_0^n S_0^n + \sum_{i\in\mathcal{I}\setminus\{0\}} e h_i^n E_i^n + l^n L^n + \sum_{i\in\mathcal{I}_r\cup\{0\}} q_i^n Q_i^n + g^n X_0^n \Big) + \sum_{\ell\in\mathfrak{L}(\gamma,\zeta)} \mathcal{Q}^\gamma(U^\ell) \quad (6.12)$$

$$X_p^n \leq M_p^n Y_p^n \qquad \forall p \in \mathcal{J}, n \in \mathcal{X}^{\gamma,\zeta} \quad (6.13)$$

$$S_0^{\xi^{\gamma,\zeta}} = \sum_{\beta\in\mathcal{B}} 2^\beta \hat{U}_0^{\xi^{\gamma,\zeta},\beta} + r^{\xi^{\gamma,\zeta}} - X_0^{\xi^{\gamma,\zeta}} - Q_0^{\xi^{\gamma,\zeta}} \quad (6.14)$$

$$E_i^{\xi^{\gamma,\zeta}} = \sum_{\beta\in\mathcal{B}} 2^\beta \hat{U}_i^{\xi^{\gamma,\zeta},\beta} + \delta_i^{\xi^{\gamma,\zeta}} \varpi_i X_0^{\xi^{\gamma,\zeta}} - \varpi_i d^{\xi^{\gamma,\zeta}} + \varpi_i L^{\xi^{\gamma,\zeta}} - Q_i^{\xi^{\gamma,\zeta}} \qquad \forall i \in \mathcal{I}_r \quad (6.15)$$

$$E_i^{\xi^{\gamma,\zeta}} = \sum_{\beta\in\mathcal{B}} 2^\beta \hat{U}_i^{\xi^{\gamma,\zeta},\beta} + X_{i-I}^{\xi^{\gamma,\zeta}} - \varpi_i d^{\xi^{\gamma,\zeta}} + \varpi_i L^{\xi^{\gamma,\zeta}} \qquad \forall i \in \mathcal{I}_s \quad (6.16)$$

$$E_{2I+1}^{\xi^{\gamma,\zeta}} = \sum_{\beta\in\mathcal{B}} 2^\beta \hat{U}_{2I+1}^{\xi^{\gamma,\zeta},\beta} + X_{I+1}^n - d^{\xi^{\gamma,\zeta}} + L^{\xi^{\gamma,\zeta}} \quad (6.17)$$

$$\hat{U}_i^{\xi^{\gamma,\zeta},\beta} = U_i^{m,\beta} \qquad \forall i \in \mathcal{I}, \beta \in \mathcal{B} \quad (6.18)$$

$$S_0^n = S_0^{a^n} + r^n - X_0^n - Q_0^n \qquad \forall n \in \mathcal{X}^{\gamma,\zeta} \setminus \{\xi^{\gamma,\zeta}\} \quad (6.19)$$

$$E_i^n = E_i^{a^n} + \delta_i^n \varpi_i X_0^n - \varpi_i d^n + \varpi_i L^n - Q_i^n \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{X}^{\gamma,\zeta} \setminus \{\xi^{\gamma,\zeta}\} \quad (6.20)$$

$$E_i^n = E_i^{a^n} + X_{i-I}^n - \varpi_i d^n + \varpi_i L^n \qquad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{X}^{\gamma,\zeta} \setminus \{\xi^{\gamma,\zeta}\} \quad (6.21)$$

$$E_{2I+1}^n = E_{2I+1}^{a^n} + X_{I+1}^n - d^n + L^n \qquad \forall n \in \mathcal{X}^{\gamma,\zeta} \setminus \{\xi^{\gamma,\zeta}\} \quad (6.22)$$

$$E_i^n - E_{I+i}^n \geq 0 \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{X}^{\gamma,\zeta} \quad (6.23)$$

$$E_i^n - \varpi_i E_{2I+1}^n \geq 0 \qquad \forall i \in \mathcal{I}_s, \forall n \in \mathcal{X}^{\gamma,\zeta} \quad (6.24)$$

$$E_{2I+1}^n \geq 0 \qquad \forall n \in \mathcal{X}^{\gamma,\zeta} \quad (6.25)$$

$$S_0^\ell = \textstyle\sum_{\beta\in\mathcal{B}} 2^\beta U_0^{\ell,\beta} \qquad \forall \ell \in \mathfrak{L}(\gamma,\zeta) \quad (6.26)$$

$$E_i^\ell = \textstyle\sum_{\beta\in\mathcal{B}} 2^\beta U_i^{\ell,\beta} \qquad \forall i \in \mathcal{I} \setminus \{0\}, \ell \in \mathfrak{L}(\gamma,\zeta) \quad (6.27)$$

$$S_0^n \geq 0, L^n \geq 0 \qquad \forall n \in \mathcal{X}^{\gamma,\zeta} \quad (6.28)$$

$$X_p^n \geq 0, Y_p^n \in \{0,1\} \qquad \forall p \in \mathcal{J}, \forall n \in \mathcal{X}^{\gamma,\zeta} \quad (6.29)$$

$$\hat{U}_i^{\xi^{\gamma,\zeta},\beta} \in [0,1] \qquad i \in \mathcal{I}, \forall \beta \in \mathcal{B} \quad (6.30)$$

$$U_i^{\ell,\beta} \in \{0,1\} \qquad \forall i \in \mathcal{I}, \ell \in \mathfrak{L}(\gamma,\zeta), \forall \beta \in \mathcal{B} \quad (6.31)$$

where U^n denotes the vector of binary variables $U^n = (U_0^{n0}, \ldots, U_0^{nB}, \ldots, U_{2I+1}^{n0}, \ldots, U_{2I+1}^{nB})$.

In this reformulation, Constraint (6.14)-(6.17) corresponds to the inventory balance at node $\xi^{\gamma,\zeta}$ in which the entering stock level E_i^m (resp. S_0^m) is computed using the auxiliary variables $\hat{U}_i^{\xi^{\gamma,\zeta}\beta}$ for each $i \in \mathcal{I} \setminus \{0\}$ (resp. for item $i = 0$). Equalities (6.18) are copy constraints ensuring that the value of each auxiliary variable $\hat{U}_i^{\xi^{\gamma,\zeta},\beta}$ is equal to the value of the corresponding state variable $U_i^{m,\beta}$ imposed by the parent node m. Constraints (6.19)-(6.22) ensure that the inventory balance is respected at each node of sub-tree $\mathcal{X}^{\gamma,\zeta}$ except the root node $\xi^{\gamma,\zeta}$. Constraints (6.26)-(6.27) define, for each leaf node $\ell \in \mathfrak{L}(\gamma,\zeta)$, the value of the binary variables $U_i^{\ell,\beta}$, which will be used to compute the future expected costs as $\mathcal{Q}^\gamma(U^\ell) = \sum_{\zeta'\in\mathcal{R}^{\gamma+1}} \mathscr{Q}^{\gamma+1,\zeta'}(U^\ell)$.

6.3.2 Sub-tree-based SDDiP algorithm

The dynamic programming reformulation (6.12)-(6.31) enables us to use the extSDDiP algorithm presented in Section 5.3. Recall that this algorithm solves problems such as (6.12)-(6.31) by iteratively building a piece-wise under-approximation of each expected cost-to-go function $\mathcal{Q}^\gamma, \gamma \in \mathcal{G}$. Each iteration of the sub-tree-based SDDiP algorithm comprises three steps:

- In the sampling step, a subset of W scenarios is sampled from the scenario tree.
- In the forward step, the algorithm proceeds stage-wise from $\gamma = 1$ to Γ by solving, for each sampled scenario ω^w and each macro-stage γ, the problem $P^\gamma(U^m, \mathcal{X}^{\gamma,\zeta^{w,\gamma}})$ with an approximate

expected cost-to-go function, where $m = \omega^w \cap \mathcal{V}^{t'(\gamma-1)}$ is the node in the sampled scenario ω^w belonging to the last period of γ.

- In the backward step, at each iteration v, the algorithm proceeds stage-wise from macro-stage Γ back to macro-stage 1 and solves, for each scenario $w = 1, \ldots, W$, each node $m \in \omega_v^w \cap \mathcal{V}^{t'(\gamma)}$ and each realization $\zeta \in \mathcal{R}^{\gamma+1}$, a suitable relaxation of $\hat{P}_v^{\gamma+1}(U_v^m, \psi_{v+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta})$, where $\psi_{v+1}^{\gamma+1}$ represents the current approximation of the expected cost-to-go function at the stage $\gamma+1$ and iteration $v+1$. The algorithm then adds supporting hyperplanes to the approximate cost-to-go functions of the previous stage.

Finally, the sub-problem solved at the root node provides a lower bound of the overall problem. The algorithm stops when the upper and lower bounds are close enough, according to a convergence criterion. The reader is referred to Section 5.3 for a detailed description of this algorithm.

6.4 Approximate partial decomposition approach

In this section, we describe the adaptations of the proposed extSDDiP algorithm to improve its computational efficiently at solving the multi-item multi-echelon lot-sizing problem under study. The proposed algorithmic enhancements are individually introduced in Sections 6.4.1, 6.4.3 and 6.4.4. Then, in Section 6.5.2, we describe how these enhancements work together in the approximate extSDDiP algorithm.

6.4.1 Sub-problem reformulation

The first adaptation is similar to the one investigated in Section 5.4.1. We namely propose to use a reformulation of each sub-problem in which the state variables are kept continuous and do not resort to a binary approximation of these variables. In this case, the finite convergence of the algorithm is no longer theoretically guaranteed. However, this approximation leads to a significant reduction of the computational effort required to solve the sub-problems (6.12)-(6.31) at each iteration of the algorithm and, hence, it may positively impact the solution quality in practice

This reformulation of problem $P^\gamma(S_0^m, E^m, \mathcal{X}^{\gamma,\zeta})$ involves continuous auxiliary variables $(\tilde{S}_0^{\xi^{\gamma,\zeta}}, \tilde{E}^{\xi^{\gamma,\zeta}})$. These variables $(\tilde{S}_0^{\xi^{\gamma,\zeta}}, \tilde{E}^{\xi^{\gamma,\zeta}})$ can be seen as a local copy of the inventory variables (S_0^m, E^m) at the parent node m in $P^\gamma(S_0^m, E^m, \mathcal{X}^{\gamma,\zeta})$. This results in the following reformulation of sub-problem $P^\gamma(S_0^m, E^m, \mathcal{X}^{\gamma,\zeta})$:

$$\mathscr{Q}^{\gamma,\zeta}(S_0^m, E^m) = \min \sum_{n \in \mathcal{X}^{\gamma,\zeta}} \rho^n \Big(\sum_{p \in \mathcal{J}} f_p^n Y_p^n + h_0^n S_0^n + \sum_{i \in \mathcal{I} \setminus \{0\}} e h_i^n E_i^n + l^n L^n + \sum_{i \in \mathcal{I}_r \cup \{0\}} q_i^n Q_i^n + g^n X_0^n \Big) + \sum_{\ell \in \mathfrak{L}(\gamma,\zeta)} \mathcal{Q}^\gamma(S_0^\ell, E^\ell) \quad (6.32)$$

$$S_0^{\xi^{\gamma,\zeta}} = \tilde{S}_i^{\xi^{\gamma,\zeta}} + r^{\xi^{\gamma,\zeta}} - X_0^{\xi^{\gamma,\zeta}} - Q_0^{\xi^{\gamma,\zeta}} \quad (6.33)$$

$$E_i^{\xi^{\gamma,\zeta}} = \tilde{E}_i^{\xi^{\gamma,\zeta}} + \delta_i^{\xi^{\gamma,\zeta}} \varpi_i X_0^{\xi^{\gamma,\zeta}} - \varpi_i d^{\xi^{\gamma,\zeta}} + \varpi_i L^{\xi^{\gamma,\zeta}} - Q_i^{\xi^{\gamma,\zeta}} \qquad \forall i \in \mathcal{I}_r \quad (6.34)$$

$$E_i^{\xi^{\gamma,\zeta}} = \tilde{E}_i^{\xi^{\gamma,\zeta}} + X_{i-I}^{\xi^{\gamma,\zeta}} - \varpi_i d^{\xi^{\gamma,\zeta}} + \varpi_i L^{\xi^{\gamma,\zeta}} \qquad \forall i \in \mathcal{I}_s \quad (6.35)$$

$$E_{2I+1}^{\xi^{\gamma,\zeta}} = \tilde{E}_{2I+1}^{\xi^{\gamma,\zeta}} + X_{I+1}^n - d^{\xi^{\gamma,\zeta}} + L^{\xi^{\gamma,\zeta}} \quad (6.36)$$

$$\tilde{S}_0^{\xi^{\gamma,\zeta}} = S_0^m \quad (6.37)$$

$$\tilde{E}_i^{\xi^{\gamma,\zeta}} = E_i^m \qquad \forall i \in \mathcal{I} \setminus \{0\} \quad (6.38)$$

$$\text{Constraints } (6.13), (6.19) - (6.25), (6.28) - (6.29) \quad (6.39)$$

In this reformulation, the expected cost-to-go function $\mathcal{Q}^\gamma(S_0^\ell, E^\ell) = \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} \mathcal{Q}^{\gamma+1,\zeta'}(S_0^\ell, E^\ell)$ is a function of the continuous state variables (S_0^ℓ, E^ℓ). We thus build an under-approximation of $\mathcal{Q}^\gamma(\cdot)$ through a set of linear cuts involving continuous variables (S_0^ℓ, E^ℓ) instead of binary variables U^ℓ.

6.4.2 Approximate sub-tree-based SDDiP algorithm

The approximate algorithm differs from the sub-tree-based SDDiP algorithm presented in Section 6.3 by two main aspects. First, it uses formulation (6.32)-(6.39) involving continuous state variables instead of formulation (6.12)-(6.31) involving binary state variables to generate cuts in the backward step. Second, it only generates *strengthened Benders' cuts* to approximate the expected cost-to-go functions whereas the extended SDDiP algorithm generates three types of cuts: *strengthened Benders' cuts*, *integer optimality cuts* and *Lagrangian cuts*. Namely, integer optimality cuts can only be generated when the state variables are binary. Moreover, the generation of Lagrangian cuts, even if possible, would imply to solve a series of dual Lagrangian problems through a sub-gradient algorithm. The updating of the Lagrangian multipliers by the sub-gradient algorithm requires to repeatedly solve medium-size mixed-integer linear programs, which is time consuming. This leads to a significant increase in the computation time needed to carry out one iteration of the SDDiP algorithm and negatively impacts its performance. In contrast, strengthened Benders' cuts can be generated with a limited computational effort.

More precisely, let $\tilde{\psi}_\upsilon^\gamma(\cdot)$ be the approximation of the expected cost-to-go function $\mathcal{Q}^\gamma(\cdot)$ available at iteration υ for macro-stage γ in the approximate SDDiP algorithm. We have:

$$\tilde{\psi}_\upsilon^\gamma(S_0^\ell, E^\ell) = \min\{\theta^{\gamma\ell} : \theta^{\gamma\ell} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\tilde{\nu}_u^{\gamma+1,\zeta'} + \tilde{\pi}_{u,0}^{\gamma+1,\zeta'} S_0^\ell + \sum_{i \in \mathcal{I}\setminus\{0\}} \tilde{\pi}_{u,i}^{\gamma+1,\zeta'} E_i^\ell) \quad \forall u \in \{1, ..., \upsilon - 1\}\} \tag{6.40}$$

where $\tilde{\nu}_u^{\gamma+1,\zeta'}$ and $\tilde{\pi}_{u,i}^{\gamma+1,\zeta'}$ are the coefficients of the cut generated at iteration $u < \upsilon$ by considering realization $\zeta' \in \mathcal{R}^{\gamma+1}$. This leads to the following approximate sub-problem $\tilde{P}_\upsilon^\gamma(S_\upsilon^m, E_\upsilon^m, \tilde{\psi}_\upsilon^\gamma, \mathcal{X}^{\gamma,\zeta})$:

$$\tilde{\mathscr{Q}}_\upsilon^{\gamma,\zeta}(S_0^m, E^m) = \min \sum_{n \in \mathcal{X}^{\gamma,\zeta}} \rho^n \Big(\sum_{p \in \mathcal{J}} f_p^n Y_p^n + h_0^n S_0^n + \sum_{i \in \mathcal{I}\setminus\{0\}} e h_i^n E_i^n + l^n L^n + \sum_{i \in \mathcal{I}_r \cup \{0\}} q_i^n Q_i^n + g^n X_0^n \Big) + \sum_{\ell \in \mathfrak{L}(\gamma,\zeta)} \theta^{\gamma\ell} \tag{6.41}$$

$$\theta^{\gamma\ell} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\tilde{\nu}_u^{\gamma+1,\zeta'} + \tilde{\pi}_{u,0}^{\gamma+1,\zeta'} S_0^\ell + \sum_{i \in \mathcal{I}\setminus\{0\}} \tilde{\pi}_{u,i}^{\gamma+1,\zeta'} E_i^\ell) \quad \forall u \in \{1, ..., \upsilon - 1\}, \quad \forall \ell \in \mathfrak{L}(\gamma,\zeta) \tag{6.42}$$

$$\text{Constraints} \ \ (6.33) - (6.39) \tag{6.43}$$

In the backward step of the approximate SDDiP algorithm, only strengthened Benders' cuts are generated. More precisely, for each macro-stage $\gamma = \Gamma - 1, \ldots, 1$, the updating of the approximation of $\mathcal{Q}^\gamma(\cdot)$ is carried out as follows.

For each node $m \in \Omega_\upsilon \cap \mathcal{V}^{t'(\gamma)}$ and each realization $\zeta' \in \mathcal{R}^{\gamma+1}$,

- We first solve the linear relaxation of $\tilde{P}_\upsilon^{\gamma+1}(S_0^m, E_\upsilon^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$ to optimality. We then get the dual value of each copy constraint (6.37)-(6.38) and use them to set the value of vector $\tilde{\pi}_\upsilon^{\gamma+1,\zeta'}$.
- We then solve the Lagrangian relaxation of $\tilde{P}_\upsilon^{\gamma+1}(S_0^m, E_\upsilon^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$ in which each copy constraint $\tilde{E}_i^{\xi^{\gamma,\zeta}} = E_i^m$ (resp. $\tilde{S}_0^{\xi^{\gamma,\zeta}} = S_0^m$) has been dualized with a Lagrangian multiplier set to $\tilde{\pi}_{\upsilon,i}^{\gamma+1,\zeta'}$ (resp. $\tilde{\pi}_{\upsilon,0}^{\gamma+1,\zeta'}$). We record the optimal value of this Lagrangian relaxation and set $\tilde{\nu}_\upsilon^{\gamma+1,\zeta'}$ to this value.

Once this procedure is carried out for all realization $\zeta' \in \mathcal{R}^{\gamma+1}$, we get the cut $\theta^{\gamma,m} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\tilde{\nu}_{\upsilon}^{\gamma+1,\zeta'} + \tilde{\pi}_{\upsilon,0}^{\gamma+1,\zeta'} S_0^m + \sum_{i \in \mathcal{I} \setminus \{0\}} \tilde{\pi}_{\upsilon,i}^{\gamma+1,\zeta'} E_i^m)$ to be added to $\tilde{\psi}_{\upsilon}^{\gamma}$ to get $\tilde{\psi}_{\upsilon+1}^{\gamma}$.

6.4.3 ε-optimal strengthened Benders' cuts

In this section, we propose to generate strengthened Benders' cuts based on sub-optimal solutions of the Lagrangian sub-problems to approximate the expected cost-to-go functions. This approach was recently introduced in [84] to reduce the computational burden needed to approximate recourse functions in the context of a Benders' decomposition algorithm. We thus investigate the use of ε-optimal cuts applied to the generation of strengthened Benders' cuts, i.e. the generation of strengthened Benders' cuts obtained by using a solution of each Lagrangian sub-problem which is at most ε units of the optimal value.

These ε-optimal strengthened Benders' cuts are generated as follows. Given an iteration υ of the sub-tree-based SDDiP algorithm, for each node $m \in \Omega_{\upsilon} \cap \mathcal{V}^{t'(\gamma)}$ and each realization $\zeta' \in \mathcal{R}^{\gamma+1}$,

- We first solve the linear relaxation of $\tilde{P}_{\upsilon}^{\gamma+1}(S_0^m, E_{\upsilon}^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$ to optimality. Then, we get the dual value of each copy constraint (6.37)-(6.38) and use them to set the value of vector $\tilde{\pi}_{\upsilon}^{\gamma+1,\zeta'}$.
- We then solve the Lagrangian relaxation of $\tilde{P}_{\upsilon}^{\gamma+1}(S_0^m, E_{\upsilon}^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$ in which each copy constraint $\tilde{E}_i^{\xi^{\gamma},\zeta} = E_i^m$ (resp. $\tilde{S}_0^{\xi^{\gamma},\zeta} = S_0^m$) has been dualized with a Lagrangian multiplier set to $\tilde{\pi}_{\upsilon,i}^{\gamma+1,\zeta'}$ (resp. $\tilde{\pi}_{\upsilon,0}^{\gamma+1,\zeta'}$), until the solution found is at most ε units from its optimal solution. We thus record in $\tilde{\nu}_{\upsilon}^{\gamma+1,\zeta'}(\varepsilon)$ the objective value of the Lagrangian relaxation of sub-problem $\tilde{P}_{\upsilon}^{\gamma+1}(S_0^m, E_{\upsilon}^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$ given by the ε-optimal solution and set $\underline{\nu}_{\upsilon}^{\gamma+1,\zeta'}(\varepsilon) = \tilde{\nu}_{\upsilon}^{\gamma+1,\zeta'}(\varepsilon) - \varepsilon$.
- Note that $\underline{\nu}_{\upsilon}^{\gamma+1,\zeta'}(\varepsilon)$ is a lower bound of the optimal value of the Lagrangian relaxation of sub-problem $\tilde{P}_{\upsilon}^{\gamma+1}(S_0^m, E_{\upsilon}^m, \tilde{\psi}_{\upsilon+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,\zeta'})$, i.e., $\tilde{\nu}_{\upsilon}^{\gamma+1,\zeta'} \geq \underline{\nu}_{\upsilon}^{\gamma+1,\zeta'}(\varepsilon)$. The following cut is thus a valid linear approximation of the corresponding expected cost-to-go function.

$$\theta^{\gamma\ell} \geq \sum_{\zeta' \in \mathcal{R}^{\gamma+1}} (\underline{\nu}_{\upsilon}^{\gamma+1,\zeta'}(\varepsilon) + \tilde{\pi}_{\upsilon,0}^{\gamma+1,\zeta'} S_0^{\ell} + \sum_{i \in \mathcal{I} \setminus \{0\}} \tilde{\pi}_{\upsilon,i}^{\gamma+1,\zeta'} E_i^{\ell}) \qquad \forall \ell \in \mathfrak{L}(\gamma, \zeta) \tag{6.44}$$

6.4.4 Generation of strengthened Benders' cuts using alternative MILP formulations

We now focus on improving the numerical performance of strengthened Benders' cuts. This is achieved through the generation of a wider set of these cuts by exploiting the alternative MILP formulations available for the sub-problems.

As explained above, to generate a strengthened Benders' cut to be added at a given iteration υ to the approximation of $\mathcal{Q}^{\gamma-1}(\cdot)$, the algorithm first solves, for each $r \in \mathcal{R}^{\gamma}$, the linear relaxation of problem $\tilde{P}_{\upsilon}^{\gamma}(S_0^m, E_{\upsilon}^m, \tilde{\psi}_{\upsilon}^{\gamma}, \mathcal{X}^{\gamma,\zeta})$, where m is a node belonging to $\mathcal{V}^{t'(\gamma-1)}$. This linear relaxation can be computed using different formulations of $\tilde{P}_{\upsilon}^{\gamma}(S_0^m, E_{\upsilon}^m, \tilde{\psi}_{\upsilon}^{\gamma}, \mathcal{X}^{\gamma,\zeta})$. A first option is to use the initial MILP formulation defined in Section 6.4.2. Other options consist in using the initial MILP formulation strengthened by (k, U) path inequalities (2.12) or the initial MILP formulation strengthened by (ℓ, k, U) path inequalities (3.38)-(3.39). The algorithm then collects the dual value of the copy constraint $\tilde{E}_i^{\xi^{\gamma},\zeta} = E_i^m$ (resp. $\tilde{S}_0^{\xi^{\gamma},\zeta} = S_0^m$) in the linear relaxation for setting coefficient $\tilde{\pi}_{\upsilon,i}^{\gamma,\zeta}$.

A key observation here is that the dual value of constraint $\tilde{E}_i^{\xi^{\gamma},\zeta} = E_i^m$ (resp. $\tilde{S}_0^{\xi^{\gamma},\zeta} = S_0^m$) in the linear relaxation will vary according to the MILP formulation used for $\tilde{P}_{\upsilon}^{\gamma}(S_0^m, E_{\upsilon}^m, \tilde{\psi}_{\upsilon}^{\gamma}, \mathcal{X}^{\gamma,\zeta})$. Hence,

for a given value of the entering stock described by the vector (S_0^m, E^m), by considering the three alternative MILP formulations available for $\tilde{P}_v^\gamma(S_0^m, E_v^m, \tilde{\psi}_v^\gamma, \mathcal{X}^{\gamma,\zeta})$, it is possible to generate three different strengthened Benders' cuts, each one corresponding to a different value of coefficients $\tilde{\nu}_v^{\gamma,\zeta}$ and $\tilde{\pi}_v^{\gamma,\zeta}$.

We point out here that, in general, there does not exist a dominance relationship between these three different cuts. In other words, a cut generated using a stronger formulation of $\tilde{P}_v^\gamma(S_0^m, E_v^m, \tilde{\psi}_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ does not necessarily lead to a better approximation of $\mathcal{Q}^{\gamma-1}(\cdot)$.

As in Section 5.4.2, we propose to study an adaptation of Algorithm 7 where strengthened Benders' cuts based on the three MILP formulations available for $\tilde{P}_v^\gamma(S_0^m, E_v^m, \tilde{\psi}_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ are sequentially generated.

It is important to note that valid inequalities generated for sub-problem $\tilde{P}_v^\gamma(S_0^m, E_v^m, \tilde{\psi}_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ at a given iteration v are valid for any sub-problem $\tilde{P}_u^\gamma(S_{u,0}^m, E_u^m, \tilde{\psi}_u^\gamma, \mathcal{X}^{\gamma,\zeta})$ to be solved at iteration $u \geq v$ as long as they only use local variables, i.e. do not use the entering stock $(S_{u,0}^m, E_u^m)$, but their local copy $(\tilde{S}_0^{\xi^{\gamma,\zeta}}, \tilde{E}^{\xi^{\gamma,\zeta}})$. Moreover, sub-problems $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$ and $\tilde{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$ only differ with respect to the objective function evaluation and have the same feasible space. Thus, any inequality valid for $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$ is also valid for $\tilde{P}_v^\gamma(\cdot, \psi_v^\gamma, \mathcal{X}^{\gamma,\zeta})$.

As mentioned above, we consider here two families of valid inequalities to strengthen the formulation of each sub-problem $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$.

The first one corresponds to the $(k, \mathcal{U})$ path inequalities investigated in Chapter 3. Thus, for a given sub-problem $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$, they can be expressed as follows.

Let $k \in \mathcal{X}^{\gamma,\zeta}$ and $\ell \in \mathfrak{L}(\gamma, \zeta)$. Let $\mathcal{U}_{k,\ell}$ be a subset of $\mathcal{P}(c_k^\ell, \ell)$. For each process $p \in \mathcal{J} \setminus \{0\}$, we have:

$$E_{p+I}^k \geq \varpi_p \sum_{\mu \in \mathcal{U}_{k,\ell}} \left[d^\mu (1 - \sum_{n \in \mathcal{P}(c_k^\ell, \mu)} Y_p^n) - L^\mu \right] \tag{6.45}$$

and for $p = 0$, we have

$$\min_{i \in \mathcal{I}_r} \left[\frac{E_i^k}{\varpi_i} \right] \geq \sum_{\mu \in \mathcal{U}_{k,\ell}} \left[d^\mu (1 - \sum_{n \in \mathcal{P}(c_k^\ell, \mu)} Y_p^n) - L^\mu \right] \tag{6.46}$$

Note that k may also take the value of $a^{\xi^{\gamma,\zeta}}$, in that case we replace E_i^k by $\tilde{E}^{\xi_i^{\gamma,\zeta}}$ in (6.45) and (6.46).

The second family of valid inequalities corresponds to the extension of valid inequalities (4.5) to a stochastic setting. We recall here that any valid inequality for deterministic problem remains valid for each path of the scenario tree in its stochastic variant. Thus, let k and v be two nodes in $\mathcal{X}^{\gamma,\zeta}$, such that $v \in \mathcal{P}(\xi^{\gamma,\zeta}, k)$. Let $\ell \in \mathfrak{L}(\gamma, \zeta)$ and $\mathcal{U}_{k,\ell}$ a subset of $\mathcal{P}(c_k^\ell, \ell)$.

The following inequalities are valid for each sub-problem (6.32)-(6.39):

$$S_0^v \hat{\delta}_i^{v,k^*} + \varpi_i^{-1} E_i^k + \sum_{\mu \in \mathcal{P}(c_k^\ell, k^*)} \phi_i^\mu Y_0^\mu \geq \sum_{\mu \in \mathcal{U}_{k,\ell}} (d^\mu - L^\mu) \qquad \forall i \in \mathcal{I}_r \tag{6.47}$$

$$S_0^v \hat{\delta}_i^{v,k^*} + \varpi_p^{-1}(E_p^v - E_{p+I}^v) + \varpi_p^{-1} E_{p+I}^k + \sum_{\mu \in \mathcal{P}(c_k^\ell, k^*)} \phi_p^\mu Y_p^\mu \geq \sum_{\mu \in \mathcal{U}_{k,\ell}} (d^\mu - L^\mu) \qquad \forall p \in \{1, ...I\} \tag{6.48}$$

$$S_0^v \hat{\delta}_i^{v,k^*} + (\varpi_i^{-1} E_i^v - E_{2I+I}^v) + E_{2I+1}^k + \sum_{\mu \in \mathcal{P}(c_k^\ell, k^*)} \phi_i^\mu Y_{I+1}^\mu \geq \sum_{\mu \in \mathcal{U}_{k,\ell}} (d^\mu - L^\mu) \qquad \forall i \in \mathcal{I}_r \tag{6.49}$$

with

$$\phi_i^\mu = \min \left\{ \sum_{n \in \mathcal{P}(c_v^\ell, \mu)} r^n \hat{\delta}_i^{n,\mu}, \sum_{n \in \mathcal{U}_{k,\ell} : n \leq \mu} d^n \right\}$$

where $\hat{\delta}_i^{n,\mu}$ denotes the maximum δ_i^v in the path $\mathcal{P}(n, \mu)$ and $k^* = \max\{v \in \mathcal{U}_{k,\ell}\}$.

6.5 Computational experiments

In this section, we focus on assessing the performance of the proposed approximate sub-tree based SDDiP algorithm by comparing it with the one of a stand-alone mathematical programming solver using the extensive formulation (3.1)-(3.11) and with the one of the original SDDiP algorithm proposed by Zou et al. [105].

We first describe the scheme used to randomly generate instances of the stochastic problem. We then present in more detail the experimental set-up used for our numerical experiments. The corresponding computational results are then discussed.

6.5.1 Instance generation

We randomly generated instances for the stochastic remanufacturing planning problem by considering various structures for the scenario tree and various values for the production-holding cost ratio g/h, the setup-holding cost ratio f/h and the returns-demand quantity ratio r/d.

Regarding the scenario tree structure, we used only balanced trees with $\Sigma \in \{4, 6, 8, 12\}$ stages, a constant number $b \in \{1, 2, 3, 5\}$ of time periods per stage and a constant number $R \in \{3, 5, 10, 20\}$ of equi-probable realizations per stage. We considered 8 possible combinations for these parameters, leading to scenario trees involving between 1000 and 3.2 million scenarios. Costs were generated by using a production-holding cost ratio $g/h \in \{2, 4\}$, a setup-holding cost ratio $f/h \in \{200, 400\}$ and a returns-demand quantity ratio $r/d \in \{1, 3, 5\}$. For each considered scenario tree structure and each possible combination of g/h, f/h and r/d, five random instances were generated, resulting in a total of 480 instances.

More precisely, for each instance, we randomly generated the input data relative to each node $n \in \mathcal{V}$ as follows.

- Demand d^n was uniformly distributed in the interval $[0, 100]$ and the returns quantity r^n was uniformly distributed in the interval $[0.8(r/d)\bar{d}, 1.2(r/d)\bar{d}]$, where $\bar{d} = \frac{1}{\mathcal{V}} \sum d^n$ is the average demand.
- The proportion of recoverable parts δ_i^n was uniformly distributed in the interval $[0.4, 0.6]$.
- The holding cost h_0^n for the returned product $i = 0$ was fixed to 1. The holding cost h_i^n for each recoverable item $i \in \mathcal{I}_r$ was randomly generated following a discrete uniform distribution over interval $[2, 7]$. Similarly, the holding cost h_i^n for each serviceable item $i \in \mathcal{I}_r$ was randomly generated following a discrete uniform distribution over interval $[7, 12]$. Finally, in order to ensure non negative echelon costs, we set the value of the inventory holding cost for the remanufactured product, h_{2I+1}^n, to $\sum_{i=1}^{I} \varpi_i h_{I+i}^n + \epsilon$, where ϵ follows a discrete uniform distribution over interval $[80, 100]$.
- The production cost g^n was uniformly distributed in the interval $[0.8(g/h)\bar{h}, 1.2(g/h)\bar{h}]$, where $\bar{h} = \frac{1}{\mathcal{V}} \sum h_{2I+1}^n$ is the average holding cost.
- The set-up cost f^n was uniformly distributed in the interval $[0.8(f/h)\bar{h}, 1.2(f/h)\bar{h}]$.
- Discarding costs $q_i^n = 0.8\bar{h}_i^n$, where $\bar{h}_i^n = \frac{1}{|\mathcal{V}(n)|} \sum_{v \in \mathcal{V}(n)} h_i^v$
- The unit penalty cost for lost sales, l^n, was fixed to 10000 per unit.

Moreover, the bill-of-materials coefficients ϖ_i were assumed to be deterministically known and thus node-independent. Their value was generated as follows. We set $\varpi_0 = \varpi_{2I+1} = 1$ and randomly generated the value of $\varpi_i = \varpi_{i+I}$, $i = 1...I$, following a discrete uniform distribution over $[1; 6]$.

6.5.2 Experimental setup

Each instance is first solved with the mathematical programming solver CPLEX 12.8 using the extensive MILP formulation (3.1)-(3.11). This solution method is denoted by CPX in what follows.

Each instance is then solved using the SDDiP algorithm proposed in [105]. This algorithm is based on the dynamic programming formulation (6.12)-(6.31). In our experiments, the binary approximation of the continuous state variables (S_0^n, E^n) is carried out as follows. For each instance, we compute an upper bound of the inventory level of item $i \in \mathcal{I} \setminus \{0\}$ at node n as $E_i^{max} = max_{\ell \in \mathcal{L}(1)} \varpi_i d^{1,\ell}$. The number B of binary variables $U_i^{n,\beta}$ is set to $B = \lceil log_2(\max_{i \in \mathcal{I} \setminus \{0\}} E_i^{max}) \rceil$.

Finally, each instance is solved using the proposed approximate sub-tree based SDDiP algorithm based on the dynamic programming formulation (6.32)-(6.39). Following the notation used in Chapter 5, we will refer to this algorithm as the extSDDiP-I algorithm. Several variants of this algorithm are considered in our numerical experiments. They differ from one another with respect to several aspects:

- First, we consider two ways of partially decomposing the scenario tree. We thus use a partition of the set of decision stages $\mathcal{S}$ in which each macro-stage corresponds to a constant number $G \in \{1, 2\}$ of stages.

- Second, ε-optimal strengthened Benders' cuts are generated in the backward step of the algorithm to build the piece-wise linear approximation of the expected cost-to-go functions. Based on the results of our preliminary experiments, we set the value of ε to 1%.

- Third, similar to the cut generation strategy used in Section 6.4.4, the algorithm sequentially adds strengthened Benders' cuts based on the three available MILP formulations for each sub-problem $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$, i.e., the initial MILP formulation (6.32)-(6.39), the MILP formulation strengthened by (k, U) path inequalities (6.45)-(6.46) and the MILP formulation strengthened by the (v, k, U) path inequalities (6.47)-(6.49). The strategy is thus based on three increasing levels of formulation.

  - In the first level ($\lambda = 0$), the algorithm (denoted by extSDDiP-I-0) generates only strengthened Benders' cuts based on the initial formulation. Then, it moves to the next level if the lower bound has not improved a threshold $\hat{\epsilon} = 0.1\%$ after a predefined number of consecutive iterations.
  - In the second level ($\lambda = 1$), the algorithm (denoted by extSDDiP-I-1) generates only strengthened Benders' cuts based on the initial formulation strengthened by (k, U) path inequalities (6.45)-(6.46), i.e., at each iteration, only (k, U) path inequalities (2.12) are added to each sub-problem $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$ using a single run of a cutting plane generation procedure proposed in Subsection 3.5.2. The algorithm moves to the next level if the lower bound has not improved a threshold $\hat{\epsilon} = 0.1\%$ after a predefined number of consecutive iterations.
  - In the third level ($\lambda = 2$), the algorithm (denoted by extSDDiP-I-2) generates only strengthened Benders' cuts based on the initial formulation strengthened by (ℓ, k, U) path inequalities (6.47)-(6.49), i.e., at each iteration, only (ℓ, k, U) path inequalities (2.12) are added to each sub-problem $P^\gamma(\cdot, \mathcal{X}^{\gamma,\zeta})$ using a single run of a heuristic cutting plane generation procedure proposed in Section 4.5, page 70. Finally, the algorithm stops if the lower bound has not improved a threshold $\hat{\epsilon} = 0.1\%$ after a predefined number of consecutive iterations.

Moreover, in all the SDDiP and extSDDiP-I algorithms, we sample $W = 1$ scenarios at each iteration and use the following stopping criteria: the algorithm stops when the lower bound LB does

not improve after 30 iterations, or when 1000 iterations have been carried out. Note that at this point, the upper bound UB is computed considering only $W = 1$ scenario and is not statistically representative. Thus, after the algorithm has stopped, we compute an updated statistical upper bound based on a larger number of scenarios as follows. We randomly sample $W' = 1000$ scenarios and compute a feasible solution for each of them using the final approximation of the expected cost-to-go functions to evaluate the objective function at each (macro)-stage. We then construct a 95% confidence interval and report the right endpoint of this interval as the statistical upper bound of the optimal value.

All the algorithms were implemented in C++ using the Concert Technology environment. The MILP and LP sub-problems embedded into the SDDiP and extSDDiP-I algorithms were solved using CPLEX 12.8. All computations have been carried out on the computing infrastructure of the Laboratoire d'Informatique de Paris VI (LIP6), which consists of a cluster of Intel Xeon Processors X5690. We set the cluster to use two 3.46GHz cores and 24GB RAM to solve each instance. We impose a time limit of 7200 seconds to method CPX to solve each instance. For the SDDiP and extSDDiP algorithms, we impose a time limit of 3600 seconds to compute a lower bound and 3600 seconds to compute the true or statistical upper bound.

6.5.3 Results

Tables 6.1-6.4 display the numerical results. Columns R and b describe the structure of the scenario tree when needed. The corresponding number of nodes in the scenario tree, $|\mathcal{V}|$, and the number of scenarios, $|\mathcal{L}(0)|$, are then provided. Column G indicates the number of stages per macro-stage in the partial decomposition of the scenario tree and Column `Method` indicates the algorithm used to solve each instance. Each line in the tables thus provides the average results of the indicated resolution method over the 60 instances corresponding to the given scenario tree structure but to various values of the r/d,f/h and g/h ratios. Column `Gap` displays the gap between the lower bound (LB) and the upper bound (UB) found by each method, i.e. $Gap = |UB - LB|/UB$. The average total computation time in seconds is reported in Column `Time(s)`, the average number of iterations in Column `#ite` and the total number of valid inequalities of type (6.45)-(6.46) or (6.47)-(6.49) generated is provided in Column `#VI`.

Results from Table 6.1 first show that, when using the extensive formulation (3.12)-(3.25), method CPX outperforms the other methods for the smallest considered instances, i.e. the instances corresponding to $\Sigma = 4$, $R = 10$ and $b = 1$, providing an average gap of 0.24% within the allotted time limit. When the number of realizations per stage increases, i.e. for the instances corresponding to

Table 6.1: Performance of each method at solving instances with $\Sigma = 4$ and $b = 1$

R	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
10	1,110	1,000	1	SDDiP	23.55	4,306.65	50	0
				extSDDiP-I-0	7.46	1,358.58	222	0
			2	extSDDiP-I-0	2.00	205.15	87	0
				extSDDiP-I-1	1.20	1,696.08	173	566
				extSDDiP-I-2	1.18	1,956.50	192	580
			4	CPX	0.24	6,513.00	0	0
20	8,420	8,000	1	SDDiP	22.80	4,202.29	29	0
				extSDDiP-I-0	6.98	1,974.58	267	0
			2	extSDDiP-I-0	4.89	1,336.35	112	0
				extSDDiP-I-1	5.55	3,342.38	156	1,403
				extSDDiP-I-2	4.70	3,463.68	165	1,518
			4	CPX	1.57	7,201.81	0	0

Table 6.2: Performance of each method at solving instances with $\Sigma = 6$ and $b = 1$

R	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
10	111,110	100,000	1	SDDiP	30.37	5,641.81	31	0
				extSDDiP-I-0	8.81	2,513.51	296	0
			2	extSDDiP-I-0	5.88	3,507.57	224	0
				extSDDiP-I-1	5.58	4,570.10	222	500
				extSDDiP-I-2	5.61	4,579.56	214	462
			4	CPX	68.27	7,217.53	0	0
20	3.36×10^6	3.2×10^6	1	SDDiP	35.67	5,608.54	19	0
				extSDDiP-I-0	9.83	3,664.12	395	0
			2	extSDDiP-I-0	8.45	4,599.68	131	0
				extSDDiP-I-1	7.66	4,894.02	109	84
				extSDDiP-I-2	7.59	4,814.77	111	84
			4	CPX	-	-	-	-

$\Sigma = 4$, $R = 20$ and $b = 1$, the relative performance of method CPX deteriorates but the average gap remains below 2%. However, when the number of stages, and consequently the size of the scenario tree, increases, the performance of method CPX strongly deteriorates. This can be seen from the results displayed in Tables 6.2-6.4: method CPX namely provides an average gap of 94% for the instances with $\Sigma = 8$ and $R = 5$ and $b = 2$ and is not able to find any feasible solution for the largest instances.

We also observe from the results displayed in Tables 6.2-6.4 that method SDDiP is able to provide feasible solutions for all the considered instances and that it consistently provides average gaps smaller than the ones obtained with method CPX for the medium to large-size instances. It thus clearly outperforms method CPX in terms of solution quality for these instances. However, the remaining gaps are still significant as it can be up to 55%.

Finally, these results show that the proposed approximate extSDDiP algorithm significantly outperforms methods CPX and SDDiP. Namely, the average gap over all considered instances is significantly decreased from more than 50% with method CPX (resp. from 38.56% with method SDDiP) to 5.19% with method extSDDiP-I-2 and $G = 2$.

We now deepen our analysis and seek to independently assess the impact of each proposed adaptation of the initial SDDiP algorithm.

We first note that a large part of the gap reduction can be obtained by using continuous (rather

Table 6.3: Performance of each method at solving instances with $\Sigma = 8$ and $R = 5$

b	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
2	195,311	78,125	1	SDDiP	47.76	7,151.78	24	0
				extSDDiP-I-0	9.11	3,929.52	403	0
			2	extSDDiP-I-0	6.31	3,501.71	220	0
				extSDDiP-I-1	6.15	5,233.31	226	1,217
				extSDDiP-I-2	5.64	4,965.82	252	1,529
			4	CPX	93.48	7,236.10	0	0
5	488,279	78,125	1	SDDiP	41.74	7,222.84	14	0
				extSDDiP-I-0	6.68	3,815.32	378	0
			2	extSDDiP-I-0	4.11	5,884.19	140	0
				extSDDiP-I-1	4.42	6,667.21	123	1,612
				extSDDiP-I-2	4.73	6,335.19	146	2,657
			4	CPX	-	-	-	-

Table 6.4: Performance of each method at solving instances with $\Sigma = 12$ and $R = 3$

b	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(0)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
1	265,719	177,147	1	SDDiP	51.04	7,207.72	38	0
				extSDDiP-I-0	11.00	3,313.90	356	0
			2	extSDDiP-I-0	7.69	2,567.41	402	0
				extSDDiP-I-1	6.34	3,807.25	424	514
				extSDDiP-I-2	6.29	3,595.10	505	685
			4	CPX	91.33	7,243.70	0	0
3	797,159	177,147	1	SDDiP	55.60	7,230.25	15	0
				extSDDiP-I-0	9.72	3,607.37	395	0
			2	extSDDiP-I-0	6.01	3,893.92	221	0
				extSDDiP-I-1	5.65	5,987.09	246	2,232
				extSDDiP-I-2	5.75	5,380.00	272	2,851
			4	CPX	-	-	-	-

than binary) state variables. This can be seen by looking at the results obtained with the extSDDiP-I-0 algorithm for $G = 1$. We observe that the gap is reduced from 38.58% with method SDDiP to 8.70% with method extSDDiP-I-0. This improvement is mainly explained by the fact that the sub-problems expressed with continuous state variables are much easier to solve than the ones expressed with binary state variables. This allows the extSDDiP-I-0 algorithm to carry out more iterations than the SDDiP algorithm within the allotted time and to build better approximations of the expected cost-to-go functions.

We then study the impact of using a partial decomposition rather than a full decomposition of the scenario tree by comparing the results obtained with extSDDiP-I-0 for $G = 1$ and $G = 2$. We thus observe that the average gap is reduced from 8.70% with the extSDDiP-I-0 algorithm based a full decomposition ($G = 1$) to 5.67 % with the extSDDiP-I-0 algorithm based on a partial decomposition involving $G = 2$ stages per macro-stage. This shows the practical interest of using a partial decomposition of the scenario tree.

Finally, we can evaluate the impact of using valid inequalities to strengthen the linear relaxation of each sub-problem in order to generate additional strengthened Benders' cuts at each iteration. Results in Tables 6.1-6.4 suggest that only a slight improvement in the performance of the extSDDiP algorithm is obtained by using improved MILP formulations for the sub-problems to generate strengthened Benders' cuts. The average gap is namely only reduced from from 5.67% with method extSDDiP-I-0 to 5.32% with method extSDDiP-I-1 and to 5.19% method extSDDiP-I-2, for a partial decomposition based on $G = 2$. Nonetheless, for some particular sets of instances, the use of valid inequalities can still provide better lower bound on average. For instance, in the case of $r/d = 5$ and $\Sigma = 12$, the method extSDDiP-I-1 is able to provide lower bounds 2.47% stronger and, with the method extSDDiP-I-2, 3.27% on average.

6.6 Conclusion and perspectives

We studied a production planning problem with remanufacturing under uncertain input data and investigated a multi-stage stochastic integer programming approach. We proposed to adapt the extSDDiP algorithm investigated in Chapter 5 in order to overcome the numerical difficulties arising from the size of the sub-problems to be solved in the backward step. Computational experiments carried out on large-size randomly generated instances suggested that the proposed approximate extSDDiP algorithm is capable of obtaining near-optimal solutions in practicable computation times and outperforms both the mathematical programming solver CPLEX 12.8 using an extensive MILP

formulation and the initial SDDiP algorithm proposed in [105].

From an algorithmic standpoint, we consider that several directions might be studied. First, at each iteration, in the backward step of the proposed algorithm, we generate a single strengthened Benders' cut per macro-stage. However, it would be possible to exploit the existence of alternative MILP formulations of the sub-problems to generate several strengthened Benders' cuts for the same macro-stage during a given iteration of the algorithm. This multi-cut variant of the algorithm will be worth investigating. Moreover, the question of how to identify the best cut within the pool of cuts that we can generate at each iteration remains unanswered. In this direction and given the nature of the sub-tree based algorithm, several solutions are available after solving a forward step. These solutions can be also used to solve the corresponding sub-problems in the backward step in order to generate different cuts. Hence, a selection solution strategy will be also worth investigating and it might positively impact the performance of the algorithm.

Another important issue that might be explored is how to efficiently implement a binary expansion of the state variables in order to generate Lagrangian and Integer Optimality cuts. These cuts are known to be tight and guarantee that the SDDiP algorithm converges to an optimal solution when the state variables are restricted to be binary. However, our numerical results carried out with method SDDiP show that a binary approximation leads to a prohibitive computations times where there are multiple items involved in the production system. Therefore, one direction could be to study a partial binarization of the state variables, where state variables of only a few leaf nodes of each sub-problem are subject to be binary. This will allow the generation of Integer Optimality and Lagrangian cuts for these sub-problems, improving the lower bound. Another direction would be to partially binarize each state variable, i.e., to represent only a part of the possible values of a state variable trough binary variables and the rest possible values remains represented by a continuous auxiliary variable. This also will allow the generation of Integer Optimality and Lagrangian cuts for these sub-problems, improving the lower bound.

In terms of problem modelling, we assumed uncapacitated production processes. Extending the present work in order to account for production resources with limited capacity could also be worth investigating.

Finally, it is worth noting that Chapters 3 and 6 focused on solving the same stochastic lot-sizing problem but with a different strategy. Chapter 3 presented a solution approach aiming at solving to optimality a deterministic equivalent problem formulated as a MILP, using small to medium-size scenario trees, i.e., using a coarse approximation of the stochastic process. In contrast, in this chapter, we sought to solve the same problem over large-size scenario trees, i.e., using a better approximation of the stochastic process, but the obtained solutions were of a lesser quality. This opens the following question. In practice, is it better to solve to optimality an instance with a scenario tree of reduced size or to solve with a good quality, but not optimal, an instance with a very large scenario tree? We believe that this question is worth investigating and should be addressed through an extensive simulation study comparing the solutions obtained with both approaches.

Part IV

Risk in stochastic lot-sizing

Chapter 7

On the risk-averse multistage stochastic uncapacitated lot-sizing problem

We present in this chapter an on-going exploratory work on risk-averse multi-stage stochastic lot-sizing. Namely, risk-neutral models based on the minimization of the expected costs such as the ones investigated in the previous chapters may lead to production plans displaying a good performance on average, but providing very poor results, i.e. production costs much higher than the expected value, in some unfavorable scenarios. In case the production manager is more concerned about these potential large monetary losses than about the average performance of the production plan, it might be useful to introduce risk measures in the problem modeling. We thus study several ways of incorporating risk aversion in the multi-stage stochastic uncapacitated lot-sizing (SULS) problem and show how the risk-averse SULS can be reformulated as a mixed-integer linear program in each case. We finally provide some preliminary results seeking to assess the practical usefulness of using a risk-averse formulation rather than a risk neutral formulation.

7.1 Introduction

In the previous chapters, we represented the probability distribution of the uncertain input data through a finite set of possible scenarios and focused on minimizing the expected cost of the production decisions based on this representation. This approach using expected costs in the objective function relies on two important assumptions. First, the production plans computed by the optimization model will be repeatedly implemented under similar conditions a number of times sufficiently large to allow the observed statistical average cost to coincide with the expected cost computed by the model. Second, the decision-maker is risk-neutral, i.e. is ready to accept that the production cost observed for certain unfavorable realizations of the input data is very high as long as this is offset by a smaller production cost in more favorable realizations. However, these two assumptions do not always hold in practice, namely, each computed production plan is implemented at most once and this within a rolling horizon framework, which means that only the decisions pertaining to the first periods of the planning horizon are actually implemented before a new production plan taking into account updated information on the input data is computed. This implies that the statistical average cost of the production decisions observed at a given point in time may significantly differ from the expected cost provided by the model. Second, the decision-maker may be risk-averse, i.e. may be more concerned about potential large monetary losses than about the average performance of the production plan, and may wish to limit the probability of occurrence and/or the magnitude of these large monetary losses.

One way to overcome these difficulties consists of incorporating risk measures in the problem modeling, either by including terms in the objective function that can measure exposition to risk and mitigate the effects of undesirable realizations ([88], [87], [92]) or by restricting the feasible solutions

space to limit the probability of occurrence or the magnitude of undesirable outcomes ([43], [42], [36]). Note that risk was initially defined as a scalar measure of the variability of outcomes and the first risk-averse models introduced in [71, 72] used the variance of the objective function as a risk measure. Now, the notion of risk measure has been somewhat extended and refers to a mathematical expression reflecting the decision-maker's preferences with respect to a set of random outcomes [9].

There exists a large variety of risk measures such as the *Conditional Value-at-Risk* (CVaR), the upper partial mean or the semi-deviation that can be used in a risk-averse model. A difficulty here comes from the fact that, as mentioned e.g. by Alem et al. [9], there does not seem to be an unrestrictedly recommendable risk measure for production lot-sizing problems or for any other class of problems. The justification for adopting one risk measure over another one usually relies on the preferences of the decision-maker, the tractability of the resulting optimization model or the theoretical properties of the risk measure. This latter point is addressed by the pioneer work of [15], which proposes a set of desirable theoretical properties that risk measures should fulfill: translation invariance, subadditivity, positive homogeneity and monotonicity. Measures satisfying these four properties are called *coherent* risk measures. An interesting feature of coherent risk measures is that any convex stochastic optimization problem minimizing an expected cost remains convex when its objective function is replaced by a coherent risk measure [77].

Among existing risk measures, the Conditional Value at Risk (CVaR) or expected shortfall is one of the most widely used: see e.g. [92], [8], [14], [11], [12], [70], [50]. Basically, the CVaR of a random variable representing a monetary loss (or a cost) is equal to the expected loss over the worst outcomes, i.e. over the outcomes for which the obtained loss is beyond a confidence level. The CVaR has two main advantages: it is a coherent risk measure and, when used in a stochastic (mixed-integer) linear programming model, it leads to the formulation of a computationally tractable (mixed-integer) linear program [88]. Moreover, several recent works (see [8], [9] and [14]) highlighted the practical interest of using CVaR in two-stage stochastic production planning in order to reduce exposition to risk. In the present work, we will thus focus on the use of CVaR as a measure to quantify the risk in a stochastic lot-sizing problem and seek to extend the previously published works by considering a multi-stage setting.

As explained e.g. in [54], when considering two-stage stochastic programming models, the extension of a risk-neutral to a risk-averse model does not pose major modeling difficulties. Namely, the objective function comprises a single random variable corresponding to the random second-stage/recourse costs and it is quite natural to replace the expectation of this second-stage costs by a risk measure such as the CVaR. Yet, the situation is different when dealing with multi-stage stochastic programming models. Namely, in a sequential decision making process, the objective function involves a series of random variables, each one corresponding to the random cost associated to a given decision stage, so that there does not seem to be one natural and obvious way of measuring risk [54]. Risk may be measured e.g. stage by stage, scenario by scenario or in a nested way. It might thus be interesting to understand and evaluate the benefits and drawbacks of each alternative way of measuring risk in a multi-stage stochastic lot-sizing problem.

In particular, one important issue encountered when dealing with risk-averse multi-stage stochastic programs is that of *time consistency*. Time-consistency is informally defined as follows: if you solve a multistage stochastic program today and find solutions for each node of a tree, you should find the same solutions if you re-solve the problem tomorrow given what was observed and decided today. Although this property may seem to be natural and desirable, it does not hold in general [93]. Specifically, Homem-de-Mello and Pagnoncelli [54] showed that two natural ways of measuring risk, based on the application of a CVaR risk measure scenario by scenario or stage by stage, lead to time-inconsistent models.

Time-consistent risk measures have been proposed among others in [90], [89] and [54]. However, despite the significant scientific discussions about the relevance and theoretical advantages of time-consistent risk measures over time-inconsistent ones, to the best of our knowledge, the question

of how good (if so) time-consistent optimal policies are in comparison to time-inconsistent optimal policies remains open. A first attempt to assess time-consistent vs time-inconsistent policies based on the CVaR measure is proposed by Alonso-Ayuso et. al. [12]. The authors of [12] study a multistage stochastic forest planning problem, in which uncertainty is in timber prices and demand, and evaluate three CVaR-based risk measures. They conclude that the time-consistent CVaR risk measure should be used when the main goal of the decision-maker is to obtain the highest profit, whereas the time-inconsistent CVaR risk measure should be preferred if the main goal of the decision-maker is to bring forward the profit to early time periods. However, as in most previous works, the authors of [12] assess the obtained optimal policies using a static framework, i.e. using a set of scenarios forming a single scenario tree. This approach assumes that the decisions relative to all the decision stages will be actually implemented, whereas, in practice, only the decisions relative to the first decision stage will be executed before a new plan is computed. Hence, the conclusions and practical insights provided in [12] should maybe be taken with some care.

In the present work, we thus study several ways of incorporating the risk aversion of the decision-maker in the multistage stochastic uncapacitated lot-sizing (SULS) problem and focus on the use of CVaR-based risk measures. Our main objective is to highlight and quantify the benefit of using a risk-averse model as compared to a more intuitive risk-neutral one and to understand the potential advantages and disadvantages of each studied risk-averse model. In order to more accurately assess the quality of the production plans provided by the risk-neutral and risk-averse models, we propose to use a dynamic framework relying on a rolling horizon simulation. More precisely, similar to what was done in Section 3.6.3 to estimate the value of the stochastic solution obtained for the stochastic remanufacturing planning problem, we iteratively solve a series of multistage stochastic models using a rolling horizon framework and evaluate, for the production plan computed at each iteration, the implementation of the first-stage decisions over a scenario representing the 'true' realization of the stochastic process.

Our contribution is thus twofold. First, we consider four different strategies based on the Conditional Value-at-Risk to measure risk in the multistage SULS problem. In particular, we investigate two time-inconsistent and two time-consistent risk measures, which have not been previously assessed in the context of multi-stage stochastic lot-sizing problems. We then formulate four mixed-integer linear programs of the multi-stage SULS problem, each one of these formulations corresponds to the case of one of the risk measures is used as an objective function for the SULS problem. These formulations serve as base models to investigate the practical relevance of each risk measure. Second, we carry out a large number of rolling-horizon simulations and provide a quantitative comparison of the production plans provided by the risk-neutral model, the time-inconsistent and time-consistent risk-averse models. To the best of our knowledge, the issue of how better (if so) the time-consistent policies may be in practice over time-inconsistent ones for multi-stage stochastic lot-sizing problems has not been previously addressed in the literature. We thus provide, in this chapter, some preliminary results that might help to better understand their difference.

The remaining part of this paper is organized as follows. Section 7.2 reviews some related works dealing with risk-averse stochastic lot-sizing problems. Section 7.3 provides some general background on risk measures, introducing in particular the definition of coherency and time consistency. Section 7.4 focuses on the risk-averse multi-stage SULS problem. Four mixed-integer linear programming formulations, based on four alternative multi-stage CVaR-based risk measures, are presented. Finally, the results of our simulations are provided and discussed in Section 7.6. Conclusions and directions for further works are given in Section 7.7.

7.2 Related works

We provide in this section a brief overview of the related literature, focusing on the papers dealing with risk-averse stochastic production planning and lot-sizing. We propose to classify these works

according to the number of stages (two or several) included in the decision process.

Risk management in two-stage stochastic production planning and lot-sizing problems was studied in several works. Macedo et al. [68] investigated a hybrid manufacturing/remanufacturing problem with stochastic demand, return and setup cost and compared a risk-averse formulation based on an upper partial mean risk measure to the risk-neutral formulation. Their numerical results showed that it is possible to obtain a large risk reduction through a minor increase of the expected total costs. Alem and Morabito [8] considered a coupled lot-sizing and cutting-stock problem and explored four strategies to handle risk aversion. They concluded that the CVaR measure is especially relevant for minimizing the losses in the worst scenarios and for reducing the standard deviation of the production costs. Moreover, from a computational standpoint, the authors of [8] noted that, for their particular problem, the risk-averse MILP formulation based on the CVaR measure could be solved faster than the risk-averse MILP formulations based on other risk measures. Amorim et al. [14] investigated two risk-averse formulations for mitigating crucial risks in the production planning of perishable food goods. They determined that, in their case, the CVaR measure provides a good trade-off between the amount of spoiled products and the expected profit loss. Alem et al. [9] studied a two-stage stochastic aggregated production planning problem. They considered four ways of incorporating the risk aversion of the decision-maker in the model: the first two approaches correspond to mean-risk models using either the semideviation or the conditional value at risk as a risk measure, the last two ones use first-order and second-order stochastic dominance constraints. Alem et al. [9] numerically compared the resulting four risk-averse formulations. In line with the above-mentioned works, their computational results showed that the CVaR measure is the one that reduces most dramatically the cost in the worst scenarios but they noted that this improvement was achieved at the expense of a non-negligible increase of the expected cost. Different versions of the risk-averse capacitated lot-sizing problem were addressed in [59], [98], [97] differing mainly in their production planning problem setting, risk measures and solution approaches but these works did not provide a numerical comparison of alternative risk-averse formulations.

As for risk management in a multistage stochastic setting, two recent works studied the risk-averse SULS problem. However, both works focused on developing a solution approach for the problem rather than on assessing the quality of the solution provided by the risk-averse model. Thus, Guo and Ryan [50] focused on extending a scenario decomposition approach previously known for risk-neutral models to solve risk-averse stochastic integer models involving expected conditional risk measures (ECRMs) and evaluated it among others on SULS instances. Similarly, Mahmutoğullari et al. [69] proposed a new methodology to find tight lower bounds for risk-averse multistage stochastic integer programs based on a nested CVaR measure and applied it on the SULS.

The work presented here is closely related to the one of [9] as, similar to [9], our main focus is on numerically comparing several risk-averse approaches on a given production planning problem (the SULS in our case). However, whereas authors of [9] compare several risk-averse approaches dealing with a single random variable (the random cost of the recourse actions in a two-stage stochastic program), we consider a single risk measure, namely the CVaR risk measure, and seek to assess several alternative ways of using it when the risk to take into account is related to a series of random variables (each one representing the random cost of the production plan implemented at a given decision stage). In particular, we focus on the comparison, in a dynamic decision setting, between time-consistent and time-inconsistent CVaR-based risk approaches. To the best of our knowledge, this is the first time such a numerical study is conducted for dynamic production planning and lot-sizing problems.

7.3 Preliminaries

This section recalls some basic definitions relative to risk measures and their properties in order to ease the understanding of the work presented in this chapter. We first discuss risk measure for a

single random variable and recall the definition of coherency. We then consider stochastic processes and recall several multi-stage risk functions investigated in the literature in order to measure risk in multi-stage stochastic programming programs. We end this section by introducing the concept of time consistency in multi-stage stochastic programming.

7.3.1 Risk measure for a single random variable

We consider a probability space $(\Omega, \mathcal{F}, \mathscr{P})$ associated to some random experiment. Ω represents the sample space, i.e. the set of all possible outcomes of the random experiment. $\mathcal{F}$ is a sigma-algebra over Ω, i.e. a collection of events or subsets of Ω. Finally, $\mathscr{P} : \mathcal{F} \to [0; 1]$ is a probability function assigning to each event in $\mathcal{F}$ a real number in $[0; 1]$ called its probability.

Let $Z : \Omega \to \mathbb{R}$ be a random variable, i.e. an $\mathcal{F}$-measurable function from Ω to the real numbers $\mathbb{R}$ associating a real value to each possible outcome of the random experiment. A risk measure ϱ is a mapping from a set of random variables to the real numbers: $\varrho(Z)$ thus denotes the risk associated to the random variable Z when evaluated using measure ϱ.

As defined in [15], a risk measure ϱ is said to be coherent if it satisfies the following properties:

- **Translation invariance**
  For any random variable Z and any $a \in \mathbb{R}$, $\varrho(Z + a) = a + \varrho(Z)$.
- **Positive homogeneity**
  For any random variable Z and any $c \in \mathbb{R}^*_+$, $\varrho(cZ) = c\varrho(Z)$.
- **Monotonicity**
  For any pair of random variables Z and W such that $Z \leq W$, i.e. such that $Z(\omega) \leq W(\omega)$ for all $\omega \in \Omega$, $\varrho(Z) \leq \varrho(W)$.
- **Subadditivity**:
  For any pair of random variables Z and W, $\varrho(Z + W) \leq \varrho(Z) + \varrho(W)$.

It is possible to gain some intuition about these properties by considering the case where the random variables Z and W represent a monetary loss or a cost. The translation invariance and positive homogeneity properties entail that the risk increases (or decreases) in the same proportion of the random variable. Monotonicity implies that, if a random variables W consistently leads to a monetary loss greater than the one obtained with random variable Z, then the risk associated to W is greater than the one associated to Z. Finally, subadditivity assures that the risk of a combined random variable is smaller than the sum of the risk of each individual random variable, which could be stated in the brisk form "a merger does not create extra risk".

Generalizations of coherent risk measures to different settings are discussed in the literature (see e.g. [63], [32], [31], [27], [90]). In particular, Föllmer and Schied [39] extended the notion of coherent risk measure to the one of convex risk measure by noting that the subadditivity and positive homogeneity properties imply convexity, i.e. imply that:

- **Convexity**: For any pair of random variables Z and W, for any $\lambda \in [0, 1]$, then $\varrho(\lambda Z + (1 - \lambda)W) \leq \lambda\varrho(Z) + (1 - \lambda)\varrho(W)$.

The coherence and convexity of a risk measure has an important impact when considering the incorporation of risk measures in stochastic optimization problems. Namely, any convex stochastic optimization problem minimizing an expected cost remains convex when its objective function is replaced by a coherent risk measure [77].

As mentioned in the introduction, a large variety of risk measures have been proposed to measure the risk associated to a single random variable. We focus here on the one which will be used in our computational study, namely the Conditional-Value-at-Risk or CVaR. The CVaR of a random

variable Z representing a cost or a monetary loss can intuitively be seen as the expected value of the costs in the $(1-\alpha)\cdot 100\%$ worst cases, where $\alpha \in]0;1[$ is a predefined confidence level. We follow the paper of Rockafellar and Uryasev [88] to define the CVaR.

Let F_Z be the cumulative probability distribution of a random variable Z:

$$F_Z(\varphi) = \mathbb{P}(\{\omega \in \Omega : Z(\omega) \leq \varphi\}) \tag{7.1}$$

The Value-at-Risk at level α, or VaR_α, of Z is defined by:

$$\text{VaR}_\alpha(Z) = \min\{\varphi \in \mathbb{R} : F_Z(\varphi) \geq \alpha\} \tag{7.2}$$

$\text{VaR}_\alpha(Z)$ thus corresponds to the α-percentile of the probability distribution of Z. In the case where Z represents a monetary cost, $\text{VaR}_\alpha(Z)$ can be interpreted as the largest possible cost that may be observed once the $1-\alpha\%$ worst potential outcomes have been excluded (see e.g. Fig 7.1).
The Conditional-Value-at-Risk at level α or CVaR_α is defined by:

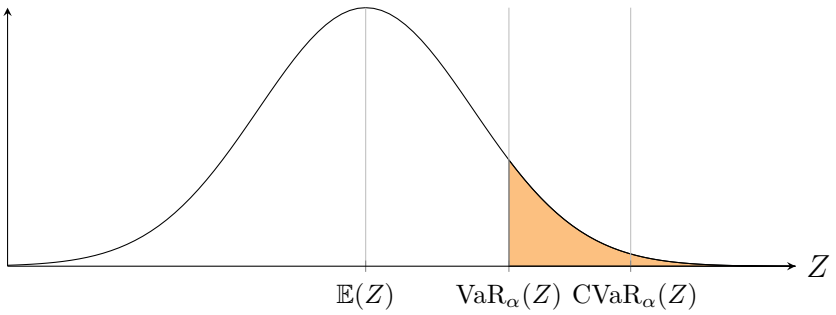
$$\text{CVaR}_\alpha(Z) = \frac{1}{1-\alpha}\int_\alpha^1 \text{VaR}_\gamma(Z)d\gamma \tag{7.3}$$

A key result in [87] is the proof that $\text{CVaR}_\alpha(Z)$ can be expressed as the optimal value of the following optimization problem:

$$\text{CVaR}_\alpha(Z) = \min_\varphi \left\{\varphi + \frac{1}{1-\alpha}\mathbb{E}[(Z-\varphi)^+]\right\} \tag{7.4}$$

where $(a)^+ = \max(a, 0)$. This reformulation is widely used to reformulate scenario-based stochastic integer programs involving a CVaR risk measure as mixed-integer linear programs. Namely, if we assume that the random variable Z has a finite and discrete set of K possible realizations, each one corresponding to a scenario Z^k having a probability ρ^k, the term $\mathbb{E}[(Z-\varphi)^+]$ in the CVaR formula (7.4) can be replaced by the weighted sum of possible outcomes exceeding the risk level φ, i.e $\sum_{k=1}^K \rho^k(Z^k-\varphi)^+$, which can easily be handled by adding a set of linear inequalities in the problem formulation.

Figure 7.1: Value-at-risk and Conditional Value-at-risk at confidence level $\alpha = 0.95$ for a normally distributed random variable



7.3.2 Risk function for a stochastic process

Let us now consider a stochastic process, i.e. a collection of random variables $\{Z_\sigma, \sigma = 1, ..., \Sigma\}$ defined on the same probability space $(\Omega, \mathcal{F}, \mathscr{P})$ and indexed by a set Σ of stages.

Recall that $\mathcal{F}$ is a sigma-algebra over Ω, i.e. a collection of subsets of Ω representing the set of all events that might be considered. In a dynamic multi-stage setting, we may be interested in the subsets of events that are *observable* at stage σ. An event is said to be observable at stage σ if it is

possible to determine the corresponding subset of outcomes in Ω using only the information available at (and including) stage σ or, in other words, if it is possible to determine whether the event has occurred or not at stage σ without waiting till the end of the considered horizon. We thus define a sub sigma-algebra over Ω, $\mathcal{F}_\sigma \subset \mathcal{F}$, corresponding to the subset of events observable at stage σ. The ordered set of sub sigma-algebras of $\mathcal{F}$, $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_\Sigma$, is called a *filtration*. Note that we have $\mathcal{F}_1 = \{\emptyset, \Omega\}$, $\mathcal{F}_\Sigma = \mathcal{F}$ and $\mathcal{F}_\sigma \subset \mathcal{F}_{\sigma'}$ for any pair of stages (σ, σ') such that $\sigma \leq \sigma'$.

The random variable Z_σ corresponding to stage σ of the above-mentioned stochastic process is thus an $\mathcal{F}_\sigma$-measurable function from Ω to $\mathbb{R}$. Let $\mathcal{Z}_\sigma$ denote the space of such $\mathcal{F}_\sigma$-measurable functions from Ω to $\mathbb{R}$ and $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_\Sigma$ be the cartesian product of these sets.

A multi-stage risk function $\mathbb{F}$ is defined as a mapping from $\mathcal{Z}$ to $\mathbb{R}$ which can be used to measure the risk associated to a stochastic process. In other words, if $(Z_1, Z_2, ..., Z_\Sigma)$ is a set of random variables belonging to $\mathcal{Z}$, $\mathbb{F}(Z_1, Z_2, ..., Z_\Sigma)$ is a real number measuring the total risk (over all decision stages) associated with the corresponding stochastic process.

As mentioned in Section 7.1, there is not a unique and natural way of defining a multi-stage risk function $\mathbb{F}$. In what follows, we provide a brief overview of four alternatives for $\mathbb{F}$, which have been previously proposed in the literature.

Aggregated risk function

The first natural way of measuring the risk of a sequence of random variables $\{Z_\sigma, \sigma \in \Sigma\}$ is to measure the risk of the unique random variable corresponding to the sum of all random variables. This can be done by using one of the risk measures ϱ defined for single random variables. This leads to an *aggregated multi-stage* risk function $\mathbb{F}$ defined by:

$$\mathbb{F}(Z_1, ..., Z_\Sigma) = \varrho(Z_1 + \cdots + Z_\Sigma) \tag{7.5}$$

where $\varrho : \mathcal{Z}_\Sigma \to \mathbb{R}$ is a single-stage risk measure. ϱ is a mapping from the set of $\mathcal{F}_\Sigma$-measurable functions, $\mathcal{Z}_\Sigma$, to the real numbers and provides information about the riskiness of the random variable equal to $Z_1 + \cdots + Z_\Sigma$.

This multi-stage risk function displays an important drawback. Again, to gain some intuition about this drawback, let us assume that $\{Z_\sigma, \sigma \in \Sigma\}$ represents some monetary loss. $\mathbb{F}$ focuses on the risk linked to the total random loss at the end of the planning horizon and does not take into account the point in time at which this loss may occur. It may thus assign the same risk value to a stochastic process in which the loss is likely to display strong time variations (e.g. large losses at the beginning of the planning horizon followed by large gains at the end) and to a stochastic process in which the loss will be more stable in time. Yet, a decision-maker may consider the first case a lot riskier than the second one.

Stage-wise risk function

One way to overcome this drawback is to measure the risk stage-wise by applying a *one-period risk measure* $\varrho_\sigma : \mathcal{Z}_\sigma \to \mathbb{R}$ to each random variable $Z_\sigma, \sigma = 2, \ldots, \Sigma$, separately. This leads to the following definition of $\mathbb{F}$:

$$\mathbb{F}(Z_1, ..., Z_\Sigma) = Z_1 + \varrho_2(Z_2) + \cdots + \varrho_\Sigma(Z_\Sigma) \tag{7.6}$$

Note that Z_1 is assumed to be deterministic so that no risk measure is applied to it.

In this case, the risk function $\mathbb{F}$ explicitly considers the dynamic aspect of the stochastic process and takes into account the risk level that may be observed at each stage of the process. But it does not exploit any potential offsetting of the risk between stages and thus may overestimate the actual risk perceived by the decision-maker. As a consequence, when used in a stochastic optimization problem, it may lead to an overly conservative optimal policy and to a significant increase in the expected cost of this policy.

Nested risk function

Another approach, which can be seen as an intermediate solution between the aggregated and the stage-wise risk functions, consists of using nested risk functions.

This approach relies on the concept of *conditional risk mapping*, or *one-stage conditional risk measure*, introduced in [90] and further studied in [89] and [93]. Formally, a one-stage conditional risk measure $\varrho_{\mathcal{Z}_\sigma|\mathcal{Z}_{\sigma-1}} : \mathcal{Z}_\sigma \to \mathcal{Z}_{\sigma-1}$ is a mapping from the space of $\mathcal{F}_\sigma$-measurable functions to the space of $\mathcal{F}_{\sigma-1}$-measurable functions defined with respect to a given realization $Z_{[\sigma-1]} = (Z_1, \ldots, Z_{\sigma-1})$ of the stochastic process up to stage $\sigma - 1$. The idea underlying the definition of $\varrho_{\mathcal{Z}_\sigma|\mathcal{Z}_{\sigma-1}}$ is that the evaluation of the future risk relative to stage σ carried out at stage $\sigma-1$ should take into account the information about the history of the stochastic process available at that time. Thus, at a given stage $\sigma - 1$, the future risk associated to a random variable $Z_\sigma \in \mathcal{Z}_\sigma$ is described as a random variable $\varrho_{\mathcal{Z}_\sigma|\mathcal{Z}_{\sigma-1}}(Z_\sigma) \in \mathcal{Z}_{\sigma-1}$ whose definition takes into account the realization of the stochastic processes observed up to stage $\sigma - 1$. Note that $\varrho_{\mathcal{Z}_\sigma|\mathcal{Z}_{\sigma-1}}(Z_\sigma) : \Omega \to \mathbb{R}$ can be understood as a random variable associating to each outcome ω a real value measuring the future risk relative to stage σ as seen from stage $\sigma - 1$.

Ruszczynski and Shapiro [90] proposed to nest conditional risk mappings to define a multi-stage risk function as follows:

$$\mathbb{F}(Z_1, ..., Z_\Sigma) = Z_1 + \varrho_2\Big(Z_2 + \varrho_{\mathcal{Z}_3|\mathcal{Z}_2}\Big(Z_3 + \cdots + \varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\Sigma-1}}(Z_\Sigma)\Big)\Big) \tag{7.7}$$

Homem-de-Mello and Pagnoncelli [54] showed that, if each conditional mapping $\varrho_{\mathcal{Z}_\sigma|\mathcal{Z}_{\sigma-1}}, \sigma = 2, ...T$, displays convexity, translation-invariance and monotonicity properties, $\mathbb{F}$ is as time-consistent risk measure for multi-stage stochastic programming problems (see Theorem 4.1 in [54]). However, the main drawback of such risk measures is that they are in general difficult to handle in a stochastic optimization problem from an algorithmic and computational viewpoints (see e.g. [93] and [77]).

Stage-wise composite risk function

Homem-de-Mello and Pagnoncelli [54] recently introduced a new class of stage-wise composite risk functions, called the *expected conditional risk measure*, which uses a concept of conditional risk mapping similar to the concept of one-stage conditional measure defined above. However, this class of risk measure aims at being more computational tractable as well as displaying relevant properties such as time-consistency. In this context, Alonso-Ayuso et al. [12] investigated a particular case of the expected conditional risk measures introduced in [54]. Formally, they defined an one-stage conditional risk measure $\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\sigma-1}} : \mathcal{Z}_\Sigma \to \mathcal{Z}_{\sigma-1}$ as a mapping from the space of $\mathcal{F}_\Sigma$-measurable functions to the space of $\mathcal{F}_{\sigma-1}$-measurable functions defined with respect to a given realization $Z_{[\sigma-1]} = (Z_1, \ldots, Z_{\sigma-1})$ of the stochastic process up to stage $\sigma - 1$. Similar to the previous case, the idea underlying the definition of $\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\sigma-1}}$ is that the evaluation of the future risk relative to the forthcoming stages $\sigma, \ldots, \Sigma$ carried out at stage $\sigma - 1$ should take into account the information about the history of the stochastic process available at that time. Thus, their multi-period risk function $\mathbb{F}$ is defined as follows:

$$\begin{aligned}\mathbb{F}(Z_1, ..., Z_\Sigma) = \varrho(Z_1 + \cdots + Z_\Sigma) + \mathbb{E}_{Z_{[1]}}[\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_1}(Z_1 + \cdots + Z_\Sigma)] + \mathbb{E}_{Z_{[2]}}[\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_2}(Z_1 + \cdots + Z_\Sigma)] \\ + \cdots + \mathbb{E}_{Z_{[\sigma-1]}}[\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\sigma-1}}(Z_1 + \cdots + Z_\Sigma)] + \cdots + \mathbb{E}_{Z_{[\Sigma-1]}}[\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\Sigma-1}}(Z_1 + \cdots + Z_\Sigma)]\end{aligned} \tag{7.8}$$

where the subscript in $\mathbb{E}$ indicates that the expectation is with respect to the corresponding variables. Note how the definition of $\mathbb{F}$ involves the random variable $\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\sigma-1}}(Z_1 + \cdots + Z_\Sigma)$ representing the future risk relative to stages $\sigma, \ldots, \Sigma$ as seen from $\sigma - 1$ and uses the expected value of $\mathbb{E}_{Z_{[\sigma-1]}}[\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\sigma-1}}(Z_1 + \cdots + Z_\Sigma)]$ over all possible realizations $Z_{[\sigma-1]}$ of the stochastic process up to $\sigma - 1$ to compute a real-valued risk relative to stage σ.

Similar to the case of nested risk functions, Homem-de-Mello and Pagnoncelli [54] showed that, if each conditional risk mapping $\varrho_{\mathcal{Z}_\Sigma|\mathcal{Z}_{\sigma-1}}(Z_\sigma)$, $\sigma = 2, \ldots, \Sigma$, displays convexity, translation-invariance and monotonicity properties, $\mathbb{F}$ is as time-consistent risk measure for multi-stage stochastic programming problems (see Theorem 5.1 in [54]). We refer the reader to [54] and [12] for further detail about this expected conditional multi-period risk function.

7.3.3 Time consistency in risk-averse multi-stage stochastic programming

Let us now consider the following risk-averse multi-stage stochastic optimization problem:

$$\begin{aligned} &\min_{x_1,\ldots,x_\sigma} \mathbb{F}(f_1(x_1,\xi_1),\ldots,f_\Sigma(x_\sigma,\xi_\Sigma)) && (\mathcal{K}) \\ &\text{s.t. } x_\sigma \in \mathcal{X}_\sigma(x_{[\sigma-1]},\xi_{[\sigma]}) && \sigma = 1,\ldots,\Sigma. \end{aligned}$$

This problem involves Σ decision stages. Let $x_\sigma \in \mathbb{R}^{n_\sigma}$ denote the decision made in stage σ and ξ_σ be an m_σ-dimensional random vector representing the uncertainty observed in stage σ, i.e. ξ_σ is an $\mathcal{F}_\sigma$-measurable mapping from Ω to $\mathbb{R}^{m_\sigma}$. We let $\mathcal{X}_\sigma(x_{[\sigma-1]},\xi_{[\sigma]})$ denote the feasibility set in stage σ, which may depend on the decisions $x_{[\sigma-1]} = [x_1, x_2, ..., x_{\sigma-1}]$ made up to the previous stage as well as on the uncertainty $\xi_{[\sigma]} = [\xi_1, \xi_2, ..., \xi_\sigma]$ observed up to stage σ. Thus, f_σ is a function from $\mathbb{R}^{n_\sigma} \times \mathbb{R}^{m_\sigma}$ to $\mathbb{R}$ that computes the cost of decision x_σ given the observed uncertainty $\xi_\sigma(\omega)$ in that stage. More precisely, $f_\sigma(x_\sigma,\xi_\sigma)$ can be seen as a random variable $Z_\sigma \in \mathcal{Z}_\sigma$ associating to each outcome $\omega \in \Omega$ a real value $Z_\sigma(\omega) = f_\sigma(x_\sigma,\xi_\sigma(\omega))$ corresponding to the cost of decision x_σ given the realization of the uncertain parameters described by vector $\xi_\sigma(\omega)$. Finally, $\mathbb{F} : \mathcal{Z} \to \mathbb{R}$ is a multi-stage risk function computing the total risk $\mathbb{F}(Z_1,\ldots,Z_\Sigma)$ associated with the stochastic process $\{Z_\sigma = (f_\sigma(x_\sigma,\xi_\sigma), \sigma = 1\ldots\Sigma\}$ representing the sequence of random cost observed at each stage. Note that it is clear from the above definition that any feasible solution $x_{[\sigma]} = [x_1,\ldots,x_\sigma]$ for $(\mathcal{K})$ is such that each $x_{[\sigma]}$ is actually a function of $\xi_1,\ldots,\xi_\sigma$, though we make that dependence explicit only when necessary to avoid cluttering the notation.

As mentioned in Section 7.1, an important issue that arises when modeling risk-averse multistage stochastic programs is that of time consistency: see e.g. [93], [94], [76] and [54]. Definitions of time consistency differ in the literature by their objective and the particular properties of the risk measure under study. In this work, we focus on the recent definition given in [54]. They informally define time consistency as follows. If you solve a multistage stochastic program today and find solutions for each node of a tree, you should find the same solutions if you re-solve the problem tomorrow given what was observed and decided today. We refer the reader to examples in Section 7.5.2, which might help to ease the understanding of the time-consistency concept.

In what follows, we recall the formal definition of time consistency provided in [54].

Let us consider the problem of solving $(\mathcal{K})$ at a given stage σ, when all the information relative to the previous stages, i.e. the past decisions $\hat{x}_{[\sigma-1]}$ and the past observations up to stage σ, $\hat{\xi}_{[\sigma]}$, is known. This optimization problem, denoted by $\mathcal{K}_\sigma$, is formulated as follows:

$$\begin{aligned} &\min_{x_\sigma,\ldots,x_\sigma} \mathbb{F}^{\hat{\xi}_{[\sigma]}}(f_1(x_1,\xi_1),\ldots,f_\Sigma(x_\sigma,\xi_\Sigma)) && (\mathcal{K}_\sigma) \\ &\text{s.t. } x_\tau \in \mathcal{X}_\tau(\hat{x}_{[\sigma-1]}, x_\sigma,\ldots,x_{\tau-1},\hat{\xi}_{[\sigma]},\xi_{\sigma+1},\ldots,\xi_\tau) && \tau = \sigma,\ldots,\Sigma. \end{aligned}$$

In the above, the notation $\mathbb{F}^{\hat{\xi}_{[\sigma]}}$ indicates a conditional multi-period risk function, i.e., the multi-period risk function $\mathbb{F}$ in $(\mathcal{K})$ applied to the random vector $f_1(x_1,\xi_1),\ldots,f_\Sigma(x_\sigma,\xi_\Sigma)$, conditional on a given realization $\hat{\xi}_1,\ldots,\hat{\xi}_\sigma$ (and implicitly on $\hat{x}_{[\sigma-1]}$). Note that under such conditions, $f_1(\hat{x}_1,\hat{\xi}_1),\ldots,f_{\sigma-1}(\hat{x}_{\sigma-1},\hat{\xi}_{\sigma-1})$ are constant, and so is $\hat{\xi}_\sigma$.

Let $[\bar{x}_\tau^{\sigma,\hat{x}_{[\sigma-1]},\hat{\xi}_{[\sigma]}} : \tau = \sigma,\ldots,\Sigma]$ denote an optimal solution of $(\mathcal{K}_\sigma)$. We include $\sigma, \hat{x}_{[\sigma-1]}$ and $\hat{\xi}_{[\sigma]}$ as superscripts to emphasize that such a solution is calculated at time σ, given the previous stages

decisions $\hat{x}_1, \ldots, \hat{x}_{\sigma-1}$ and conditional on a given realization $\hat{\xi}_1, \ldots, \hat{\xi}_\sigma$. Moreover, as mentioned earlier, each $\bar{x}_\tau^{\sigma, \hat{x}_{[\sigma-1]}, \hat{\xi}_{[\sigma]}}$, $\tau = \sigma, \ldots, \Sigma$, is a function of $\xi_{\sigma+1}, \ldots, \xi_\tau$.

We now introduce the inherited optimality property, which will serve to define the time consistency property as defined in [54].

Definition 1

The Inherited Optimality Property (hence-forth called IOP) holds for an instance of problem $(\mathcal{K})$ if, given any stage σ such that $1 < \sigma \leq \Sigma$ and any realization $\hat{\xi}_1, \ldots, \hat{\xi}_\sigma$, there exists an optimal solution x^* of $(\mathcal{K})$ such that the solution "inherited" from x^* at $\hat{\xi}_2, \ldots, \hat{\xi}_\sigma$ (denoted as $[x_\tau^*(\hat{\xi}_2, \ldots, \hat{\xi}_\sigma, \cdot) : \tau = \sigma, \ldots, T]$, where "$(\cdot)$" indicates this is a function of $\xi_{\sigma+1}, \ldots, \xi_\tau$) coincides with an optimal solution of $(\mathcal{K}_\sigma)$ for those $\sigma, \hat{\xi}$, and $\hat{x} = x^*$.

It is important to observe in the above definition that, in general, the problem $(\mathcal{K})$ may have multiple optimal solutions. When solving a problem $(\mathcal{K}_\sigma)$, one might hit upon an optimal solution which is different from the one inherited from period 1. The IOP only requires that one of the optimal solutions of $(\mathcal{K}_\sigma)$ coincides with some inherited solution from period 1. Of course, when $(\mathcal{K})$ and $(\mathcal{K}_\sigma)$ have unique optimal solutions, then the optimal solution of $(\mathcal{K}_\sigma)$ must be the solution "inherited" from the optimal solution in period 1. We thus now introduce the definition of time consistency as is done in [54]:

Definition 2

We say that the multi-period risk measure $\mathbb{F}$ is consistent for a problem of the form $(\mathcal{K})$ if the IOP holds for any particular instance of that problem.

Note that the definition of consistency in [54] is independent of any particular instance of the problem; for example, when working with scenario trees, a multi-period risk measure $\mathbb{F}$ is consistent no matter which tree we use to represent the input process. It should be mentioned that if the time consistency property does not hold, it does not mean that the corresponding policies are not implementable. We only would like to point out that there is an additional consideration, associated with a chosen optimality criterion, which is worthwhile to keep in mind [93]. We refer the reader to Section 7.5.2 for examples of time-consistent and time-inconsistent solutions.

In what follows, we present four mathematical formulations of the risk-averse multi-stage SULS problem, considering the four different ways of measuring the risk of a stochastic process discussed in 7.3.2.

7.4 Mathematical formulations

We now focus on the multi-stage SULS problem and investigate how to incorporate the risk aversion of the production planner into the problem.

Following the definitions introduced in Subsection 7.3.3, the multi-stage SULS problem involves Σ decision stages, each one corresponding to a single production planning period. The decisions made at stage σ correspond to a vector $(X_\sigma, Y_\sigma, S_\sigma)$ describing the production quantity, setup state and inventory level at this stage. The uncertainty observed at stage σ is described by a random vector $\xi_\sigma = (d_\sigma, f_\sigma, g_\sigma, h_\sigma)$ giving the realization of the demand and costs at this stage. The feasibility space for the decisions relative to stage σ is given by $\mathcal{X}_\sigma(X_{[\sigma-1]}, Y_{[\sigma-1]}, S_{[\sigma-1]}, \xi_{[\sigma]}) = \{(X_\sigma, Y_\sigma, S_\sigma) | X_\sigma \leq Y_\sigma, S_\sigma = S_{\sigma-1} + X_\sigma - d_\sigma, X_\sigma \geq 0, S_\sigma \geq 0, Y_\sigma \in \{0, 1\}\}$. Finally, the random cost relative to stage σ is given by $Z_\sigma(\omega) = f_\sigma(\omega)Y_\sigma + g_\sigma(\omega)X_\sigma + h_\sigma(\omega)S_\sigma$.

Using this notation, we can formulate the risk-averse SULS investigated here as:

$$\min \mathbb{F}(Z_1, \ldots, Z_\Sigma) \tag{7.9}$$

$$(X_\sigma, Y_\sigma, S_\sigma) \in \mathcal{X}_\sigma(X_{[\sigma-1]}, Y_{[\sigma-1]}, S_{[\sigma-1]}, \xi_{[\sigma]}) \qquad \sigma = 1, \ldots, \Sigma \tag{7.10}$$

$$Z_\sigma = f_\sigma Y_\sigma + g_\sigma X_\sigma + h_\sigma S_\sigma \qquad \sigma = 1, \ldots, \Sigma. \tag{7.11}$$

The objective function (7.9) aims at minimizing a general risk measure over the random variables $Z_1, \ldots, Z_\Sigma$.

In order to obtain a tractable mathematical program, we assume as done in Chapter 5 that the stochastic process described by ξ has a finite probability space and that its evolution can be described by a scenario tree $\mathcal{V}$. Let d^n, f^n, h^n and g^n denote the value of the demand and cost parameters at node $n \in \mathcal{V}$. As in Chapter 5, we introduce the following decision variables:

- X^n: quantity produced at node $n \in \mathcal{V}$,
- $Y^n = 1$ if a setup for production is carried out at node $n \in \mathcal{V}$, $Y^n = 0$ otherwise,
- S^n: inventory level at node $n \in \mathcal{V}$.

We now discuss how to reformulate Problem (7.9)-(7.11) into a mixed-integer linear program depending on the expression chosen for the risk function $\mathbb{F}$. We provide a reformulation for each expression of $\mathbb{F}$ introduced in Subsection 7.3.2. Moreover, we use the Conditional Value-at-Risk (CVaR) as single-stage risk measure. Namely, as mentioned in the introduction of this chapter, the CVaR displays many advantages, both from a theoretical and practical point of view. It is a coherent risk measure and its value can be computed rather easily in a stochastic optimization problem thanks to the reformulation (7.4). Moreover, its practical interest for two-stage stochastic production planning problems has been highlighted by several recent works (see [8], [9] and [14]).

7.4.1 Aggregated scenario-wise risk formulation

We first consider the case where $\mathbb{F}$ is an aggregated risk function measuring the global risk as the CVaR_α of a single random variable corresponding to the total cost over the whole planning horizon. $\mathbb{F}$ is thus defined as:

$$\mathbb{F}(Z_1, ..., Z_\Sigma) = \text{CVaR}_\alpha(Z_1 + \cdots + Z_\Sigma) \tag{7.12}$$

Using the reformulation (7.4) of the CVaR and expressing the expected value as the weighted sum over the set of scenarios described by $\mathcal{L}(1)$, we have:

$$\mathbb{F}(Z_1, ..., Z_\Sigma) = \min_\varphi \left\{ \varphi + \frac{1}{1-\alpha} \mathbb{E}[(Z_1 + \cdots + Z_\Sigma - \varphi)^+] \right\} \tag{7.13}$$

$$= \min_\varphi \left\{ \varphi + \frac{1}{1-\alpha} \sum_{\ell \in \mathcal{L}(1)} \rho^\ell \Big(\sum_{n \in \mathcal{P}(1,\ell)} (f^n Y^n + h^n S^n + g^n X^n) - \varphi \Big)^+ \right\} \tag{7.14}$$

This allows us to reformulate Problem (7.9)-(7.11) as a mixed-integer linear program.
We introduce the following additional decision variables to measure the risk:

- φ: risk level or value-at-risk for confidence level α,
- V^ℓ: risk of scenario $\ell \in \mathcal{L}(1)$, i.e. overcost of scenario ℓ with respect to the value-at-risk φ.

This leads to the following MILP formulation:

$$\min \varphi + \frac{1}{1-\alpha} \sum_{\ell \in \mathcal{L}(1)} \rho^\ell V^\ell \tag{7.15}$$

$$\sum_{n \in \mathcal{P}(1,\ell)} (f^n Y^n + h^n S^n + g^n X^n) - \varphi \leq V^\ell \qquad \forall \ell \in \mathcal{L}(1) \tag{7.16}$$

$$X^n \leq M^n Y^n \qquad \forall n \in \mathcal{V} \tag{7.17}$$
$$S^n + d^n = X^n + S^{a^n} \qquad \forall n \in \mathcal{V} \tag{7.18}$$
$$X^n, S^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{7.19}$$
$$Y^n \in \{0,1\} \qquad \forall n \in \mathcal{V} \tag{7.20}$$
$$V^\ell \geq 0 \qquad \forall \ell \in \mathcal{L}(1) \tag{7.21}$$

The objective function (7.15) comprises the sum of the value-at-risk φ and the expected overcost over all scenarios $\sum_{\ell \in \mathcal{L}(1)} \rho^\ell V^\ell$ multiplied by factor $\frac{1}{1-\alpha}$. Constraints (7.16) ensure that each variable V^ℓ is equal to the overcost of scenario ℓ with respect to the value-at-risk φ if this one is positive, and to 0 otherwise. Constraints (7.17) link the production and setup variables. Note that the value of constant M^n can be set by using an upper bound on the quantity to be processed at node n, usually defined as the maximum future demand as seen from node n, i.e. $M^n = \max_{\ell \in \mathcal{L}(n)} d^{n\ell}$, where $d^{n\ell} = \sum_{m \in \mathcal{P}(n,\ell)} d^m$. Constraints (7.18) are the inventory balance constraints. Constraints (7.19)-(7.21) provide the decision variables domains.

7.4.2 Stage-wise risk formulation

Second, we consider the case where $\mathbb{F}$ is a stage-wise risk function in which we compute the risk at each stage separately using CVaR_α as the risk measure and then sum over all stages the value of the risk measured at each stage. This leads to the following definition of $\mathbb{F}$:

$$\mathbb{F}(Z_1, ..., Z_\Sigma) = Z_1 + \text{CVaR}_\alpha(Z_2) + \cdots + \text{CVaR}_\alpha(Z_\Sigma) \tag{7.22}$$

Using the reformulation (7.4) of the CVaR and expressing the expected value as the weighted sum over all nodes belonging to stage σ , we have:

$$\mathbb{F}(Z_1, ..., Z_\Sigma) = Z_1 + \sum_{\sigma=2}^{\Sigma} \min_{\varphi_\sigma} \left\{ \varphi_\sigma + \frac{1}{1-\alpha} \mathbb{E}[(Z_\sigma - \varphi_\sigma)^+] \right\} \tag{7.23}$$
$$= f^1 Y^1 + h^1 S^1 + g^1 X^1 + \sum_{\sigma=2}^{\Sigma} \min_{\varphi_\sigma} \left\{ \varphi_\sigma + \frac{1}{1-\alpha} \sum_{n \in \mathcal{V}_\sigma} \rho^n \Big(f^n Y^n + h^n S^n + g^n X^n - \varphi_\sigma \Big)^+ \right\} \tag{7.24}$$

In order to reformulate Problem (7.9)-(7.11) as a mixed-integer linear program, we introduce the following decision variables.

- φ_σ: risk level or value-at-risk at stage σ,
- V^n: risk of node $n \in \mathcal{V}$, i.e. overcost at node n with respect to the value-at-risk φ_{σ^n} corresponding to its stage σ^n.

This leads to the following MILP formulation:

$$\min f^1 Y^1 + h^1 S^1 + g^1 X^1 + \sum_{\sigma=2}^{\Sigma} (\varphi_\sigma + \frac{1}{1-\alpha} \sum_{n \in \mathcal{V}_\sigma} \rho^n V^n) \tag{7.25}$$

$$f^n Y^n + h^n S^n + g^n X^n - \varphi_{\sigma^n} \leq V^n \qquad \forall n \in \mathcal{V} \tag{7.26}$$
$$V^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{7.27}$$
$$\text{Constraints } (7.17) - (7.20) \tag{7.28}$$

The objective function (7.25) computes the risk as the sum, over all stages, of the Conditional Value-at-Risk of the random cost at each stage. The one is equal to the sum of the Value-at-Risk φ_σ for each stage σ and of the expected overcost over all nodes belonging to stage σ multiplied by coefficient $\frac{1}{1-\alpha}$. Note that the first stage cost is assumed to be deterministic so that the objective function only comprises the production, set-up and inventory holding cost of its production decision. Similar to the previous case, equations (7.26)-(7.27) ensure that V^n accurately computes the overcost (if any) at node n with respect to the Value-at-Risk φ_σ corresponding to its stage σ^n.

7.4.3 Nested risk formulation

We now present a MILP formulation for the case where $\mathbb{F}$ is a nested risk function based on the use of conditional risk mappings. These ones measure the future risk relative to each stage σ as seen from the previous stage $\sigma-1$ taking into account the realization of the stochastic process up to stage $\sigma-1$.

We focus here on the particular case where these conditional risk mappings are computed using a CVaR risk function. Recall that, in our context of multi-stage stochastic optimization, the random variable Z_σ represents the cost at stage σ and is a function of the production decisions at stage σ and of the realizations $\xi_{[\sigma]}$ of the stochastic input parameters up to stage σ. For each stage $\sigma-1$, the conditional risk mapping $\varrho_{Z_\sigma|Z_{\sigma-1}}$ thus corresponds to the conditional one-stage CVaR denoted by $\text{CVaR}_\alpha^{\xi_{[\sigma-1]}}$. Using the reformulation (7.4) of the CVaR risk measure, $\text{CVaR}_\alpha^{\xi_{[\sigma-1]}}$ can be expressed as:

$$\text{CVaR}_\alpha^{\xi_{[\sigma-1]}}(Z_\sigma) = \min_{\varphi_\sigma^{\xi_{[\sigma-1]}}} \left\{ \varphi_\sigma^{\xi_{[\sigma-1]}} + \frac{1}{1-\alpha}\mathbb{E}_{\xi_\sigma}\left[\left(Z_\sigma - \varphi_\sigma^{\xi_{[\sigma-1]}}\right)^+ \middle| \xi_{[\sigma-1]}\right] \right\} \tag{7.29}$$

When the evolution of the stochastic process ξ is described by a scenario tree, the realizations of the stochastic process up to stage $\sigma-1$ corresponds to a node n belonging to $\mathcal{V}_{\sigma-1}$. Thus, the Value-at-Risk at stage σ given a realization $\xi_{[\sigma-1]}$, $\varphi_\sigma^{\xi_{[\sigma-1]}}$, can be denoted by φ_σ^n. Similarly, the conditional expectation over all possible realizations of ξ_σ given a realization $\xi_{[\sigma-1]}$, $\mathbb{E}_{\xi_\sigma}\left[\cdots\middle|\xi_{[\sigma-1]}\right]$, can be computed as a weighted sum using the conditional probability $\rho^{m|n}$ giving the probability of a transition from node n to its child node m in the scenario tree. This gives: $\mathbb{E}_{\xi_\sigma}\left[\cdots\middle|\xi_{[\sigma-1]}\right] = \sum_{m\in\mathcal{C}(n)}\rho^{m|n}(\cdots)$. We thus have, for each node $n\in\mathcal{V}_{\sigma-1}$:

$$\begin{aligned}\text{CVaR}_\alpha^{\xi_{[\sigma-1]}}(Z_\sigma) &= \text{CVaR}_\alpha^n(Z_\sigma)\\ &= \min_{\varphi_\sigma^n}\left\{\varphi_\sigma^n + \frac{1}{1-\alpha}\sum_{m\in\mathcal{C}(n)}\rho^{m|n}(f^mY^m + h^mS^m + g^mX^m - \varphi_\sigma^n)^+\right\}\end{aligned} \tag{7.30}$$

Let us now consider the risk function $\mathbb{F}$ defined by nesting conditional one-stage CVAR risk measures as follows:

$$\begin{aligned}\mathbb{F}(Z_1,...,Z_\Sigma) = Z_1 + \text{CVaR}_\alpha(Z_2 + [\text{CVaR}_\alpha^{\xi_{[2]}}(Z_3 + [\text{CVaR}_\alpha^{\xi_{[3]}}(Z_4\\ + \cdots + [\text{CVaR}_\alpha^{\xi_{[\Sigma-1]}}(Z_\Sigma)])])])\end{aligned} \tag{7.31}$$

The multi-stage risk function (7.31) can be integrated within a mixed-integer linear programming formulation as follows.

We first compute the conditional one-stage CVaR at node $m\in\mathcal{V}^{\Sigma-1}$, $\text{CVaR}_\alpha^{\xi_{[\Sigma-1]}}(Z_\Sigma)$ by using (7.30) and introducing continuous variable V^μ for each node $\mu\in\mathcal{C}(m)$ as follows.

$$\text{CVaR}_\alpha^m(Z_\Sigma) = \min_{\varphi_\Sigma^m}\left\{\varphi_\Sigma^m + \frac{1}{1-\alpha}\sum_{\mu\in\mathcal{C}(m)}\rho^{\mu|m}V^\mu\right\} \tag{7.32}$$

$$f^\mu Y^\mu + g^\mu X^\mu + h^\mu S^\mu - \varphi_\sigma^m \leq V^\mu \qquad \forall \mu \in \mathcal{C}(m) \tag{7.33}$$

We then compute the conditional one-stage CVaR at node $n \in \mathcal{V}^{\Sigma-2}$, $\text{CVaR}_\alpha^{\xi_{[\Sigma-2]}}[Z_{\Sigma-1} + \text{CVaR}_\alpha^{\xi_{[\Sigma-1]}}(Z_\Sigma)]$, as follows:

$$\begin{aligned}
&\text{CVaR}_\alpha^n[Z_{\Sigma-1} + \text{CVaR}_\alpha^{\xi_{[\Sigma-1]}}(Z_\Sigma)] \\
&\quad = \min_{\varphi_{\Sigma-1}^n} \Big\{ \varphi_{\Sigma-1}^n + \frac{1}{1-\alpha} \sum_{m \in \mathcal{C}(n)} \rho^{m|n} \Big(f^m Y^m + g^m X^m + h^m S^m \\
&\qquad\qquad + \min_{\varphi_\Sigma^m} \{ \varphi_\Sigma^m + \frac{1}{1-\alpha} \sum_{\mu \in \mathcal{C}(m)} \rho^{\mu|m} V^\mu \} - \varphi_{\Sigma-1}^n \Big)^+ \Big\} \qquad (7.34) \\
&f^\mu Y^\mu + h^\mu S^\mu + g^\mu X^\mu - \varphi_\sigma^{a^\mu} \leq V^\mu \qquad \forall \mu \in \mathcal{C}(m) \qquad (7.35)
\end{aligned}$$

By moving the inner minimization term out of the brackets and introducing a continuous variable V^m for each node $m \in \mathcal{C}(n)$, we get:

$$\begin{aligned}
&\text{CVaR}_\alpha^n[Z_{\Sigma-1} + \text{CVaR}_\alpha^{\xi_{[\Sigma-1]}}(Z_\Sigma)] \\
&\quad = \min_{\varphi_{\Sigma-1}^n, \varphi_\Sigma^m} \Big\{ \varphi_{\Sigma-1}^n + \frac{1}{1-\alpha} \sum_{m \in \mathcal{C}(n)} \rho^{m|n} V^m \Big\} \qquad (7.36) \\
&f^\mu Y^\mu + h^\mu S^\mu + g^\mu X^\mu - \varphi_\sigma^{a^\mu} \leq V^\mu \qquad \forall \mu \in \mathcal{C}(m), m \in \mathcal{C}(n) \qquad (7.37) \\
&f^m Y^m + g^m X^m + h^m S^m + \varphi_\Sigma^m + \frac{1}{1-\alpha} \sum_{\mu \in \mathcal{C}(m)} \rho^{\mu|m} V^\mu - \varphi_{\Sigma-1}^n \leq V^m \qquad \forall m \in \mathcal{C}(n) \qquad (7.38) \\
&0 \leq V^\mu \qquad \forall \mu \in \mathcal{C}(m), m \in \mathcal{C}(n) \qquad (7.39) \\
&0 \leq V^m \qquad \forall m \in \mathcal{C}(n) \qquad (7.40)
\end{aligned}$$

We then repeat this unnesting operation stage by stage, starting from stage $\Sigma - 2$ and going up to stage 1, and introduce the following set of decision variables:

- φ_σ^m: risk level or Value-at-Risk at stage σ measured at node $m \in \mathcal{V}^{\sigma-1}$.
- V^n: risk at node $n \in \mathcal{V} \setminus \{1\}$, i.e. overcost at node n with respect to the risk level $\varphi_\sigma^{a^n}$ fixed by its parent node a^n in the scenario tree.

This leads to the following MILP formulation:

$$\min(f^1 Y^1 + h^1 S^1 + g^1 X^1) + (\varphi_1^2 + \frac{1}{1-\alpha} \sum_{n \in \mathcal{C}(1)} \rho^{1|n} V^n) \tag{7.41}$$

$$\begin{aligned}
&f^n Y^n + h^n S^n + g^n X^n \\
&\qquad + \varphi_n^{\sigma^n+1} + \frac{1}{1-\alpha} \sum_{m \in \mathcal{C}(n)} \rho^{m|n} V^m - \varphi_{a^n}^{\sigma^n} \leq V^n \qquad \forall n \in \mathcal{V}^\sigma, \sigma \in \{2, ..., \Sigma - 1\} \qquad (7.42) \\
&f^n Y^n + h^n S^n + g^n X^n - \varphi_{a^n}^{\sigma^n} \leq V^n \qquad \forall n \in \mathcal{V}^\Sigma \qquad (7.43) \\
&V^n \geq 0 \qquad \forall n \in \mathcal{V} \qquad (7.44) \\
&\text{Constraints } (7.17) - (7.20) \qquad (7.45)
\end{aligned}$$

Similar to the two previous cases, the objective function (7.41) comprises two terms: the first one aims at minimizing the cost over the first stage, which is assumed deterministic, and the second one seeks to minimize a multi-stage risk measure. This later computes the risk in a nested way. In the objective function, we thus have a term $\varphi_1^2 + \frac{1}{1-\alpha}\sum_{n\in\mathcal{C}(1)}\rho^{1n}V^n$ corresponding to the evaluation of the risk at stage 1. Note how this evaluation takes into account the risk at all the subsequent stages thanks to inequalities (7.42) which link together the variables computing the risk level at two subsequent stages.

7.4.4 Expected conditional risk formulation

Finally, we present a MILP formulation for the case where $\mathbb{F}$ is a stage-wise composite risk function based on the Expected Conditional Risk Measure (ECRM) introduced in [54]. The ECRM measures the risk stage by stage in a composite way. It namely computes the risk at stage $\sigma-1$ as the expected value of a conditional risk mapping measuring the future risk relative to stages $\sigma\cdots\Sigma$ as seen from stage $\sigma-1$ taking into account the realization of the stochastic process up to stage $\sigma-1$. We thus focus on the Expected Conditional Value-at-Risk measure (ECVaR) presented in [12], which belongs to the family of ECRMs investigated in [54].

The multi-stage risk function $\mathbb{F}$ can be written as:

$$\mathbb{F}(Z_1,...,Z_\Sigma) = \sum_{\sigma=1}^{\Sigma}\mathbb{E}_{\xi_{[\sigma-1]}}\Big[\mathrm{CVaR}_\alpha^{\xi_{[\sigma-1]}}(Z_1+\cdots+Z_\Sigma)\Big] \tag{7.46}$$

$$= \sum_{\sigma=1}^{\Sigma}\sum_{m\in\mathcal{V}_{\sigma-1}}\rho^m\Big(\min_{\varphi_\sigma^m}\Big\{\varphi_\sigma^m + \frac{1}{1-\alpha}\sum_{\ell\in\mathcal{L}(m)}\rho^{\ell|m}\Big(\sum_{n\in\mathcal{P}(1,\ell)} f^nY^n + h^nS^n + g^nX^n - \varphi_\sigma^m\Big)^+\Big\}\Big) \tag{7.47}$$

$$= \sum_{\sigma=1}^{\Sigma}\sum_{m\in\mathcal{V}_{\sigma-1}}\Big(\min_{\varphi_\sigma^m}\Big\{\rho^m\varphi_\sigma^m + \frac{1}{1-\alpha}\sum_{\ell\in\mathcal{L}(m)}\rho^{\ell}\Big(\sum_{n\in\mathcal{P}(1,\ell)} f^nY^n + h^nS^n + g^nX^n - \varphi_\sigma^m\Big)^+\Big\}\Big) \tag{7.48}$$

In order to formulate Problem (7.9)-(7.11) with the stage-wise composite risk measure (7.48) as a mixed-integer linear program, we introduce the following decision variables:

- φ_σ^m: Value-at-Risk at stage $\sigma\in\mathcal{S}$, given the realization of the stochastic process up to stage $\sigma-1$ corresponding to node $m\in\mathcal{V}^{\sigma-1}$,
- V^ℓ: risk of scenario $\ell\in\mathcal{L}(1)$, i.e. overcost of scenario ℓ with respect to the value-at-risk $\varphi_{a^n}^{\sigma_n}$ defined for its stage σ^n and by its ancestor node a^n.

This leads to the following MILP formulation:

$$\min\sum_{\sigma=1}^{\Sigma}\sum_{m\in\mathcal{V}_{\sigma-1}}\Big(\rho^m\varphi_\sigma^m + \frac{1}{1-\alpha}\sum_{n\in\mathcal{C}(m)}\rho^nV^n\Big) \tag{7.49}$$

$$\sum_{n\in\mathcal{P}(1,\ell)} f^nY^n + h^nS^n + g^nX^n - \varphi_{\sigma^m}^{a^m} \le V^\ell \qquad \forall\ell\in\mathcal{L}(m), m\in\mathcal{V} \tag{7.50}$$

$$V^n \ge 0 \qquad \forall n\in\mathcal{V} \tag{7.51}$$

Constraints (7.17) − (7.20) (7.52)

In this case, the objective function (7.49) seeks to minimize a stage-wise composite risk measure. Thus, for each stage $\sigma = 1, ..., \Sigma$, the risk function computes the expected value, over all nodes $m \in \mathcal{V}_{\sigma-1}$ belonging to the previous stage, of $\text{CVaR}_\alpha^m(Z_1 + \cdots + Z_\Sigma)$, i.e the CVaR of the random cost $Z_1 + \cdots + Z_\Sigma$ at stage σ conditional to the realization of the stochastic input process corresponding to node m. For each stage σ and each node $m \in \mathcal{V}_{\sigma-1}$, $\text{CVaR}_\alpha^m(Z_1 + \cdots + Z_\Sigma)$ is computed as the minimum of the sum of the Value-at-Risk φ_σ^m evaluated at node m and of the expected overcost over the scenarios in the associated subtree rooted in node m multiplied by coefficient $\frac{1}{1-\alpha}$. Equations (7.50) ensure that V^ℓ accurately computes the overcost (if any) corresponding to scenario $\mathcal{P}(1, \ell)$ with respect to the Value-at-Risk $\varphi_{\sigma^n}^{a^n}$ corresponding to its stage σ^n and its ancestor node a^n.

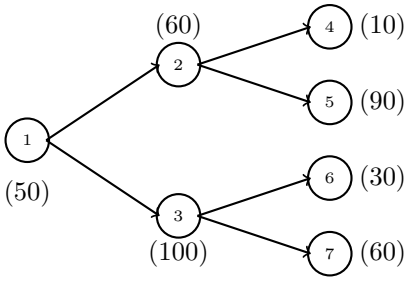
7.5 Small illustrative examples

Before providing our preliminary computational results, we present two small examples. The first one illustrates how the production plan may vary according to the risk function used in the objective function. The second shows that in some cases, the obtained production plan does not display the inherited optimality property.

7.5.1 Comparison of production plans obtained with risk-neutral and risk-averse formulations

Consider a decision-maker trying to plan the production over a three time-period horizon. For the sake of simplicity, all costs are assumed to be deterministic and time invariant. The unit production cost is set to $g = 1$ and the unit inventory holding cost to $h = 1$. A setup cost $f = 100$ is to be paid in each time period where there is a strictly positive production. The stochastic demand at time period 2 is revealed at the beginning of time period 2 and the stochastic demand d at time period 3 is revealed at the beginning of time period 3. Suppose that the stochastic process representing the evolution of the demand can be described by the following scenario tree, in which the conditional probability of transition from a node to any of its child is set to 0.5. The confidence level used in the CVaR computation is set to $\alpha = 0.95$.

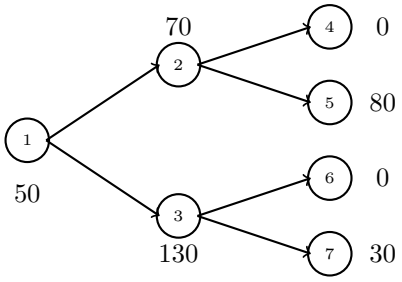
Figure 7.2: Small illustrative example. Values between brackets represent the demand d^n a each node of the scenario tree.



Risk-neutral production plan Let us first consider the production plan obtained while using the risk-neutral formulation (5.1)-(5.4) and provided in Figure 7.3. This production plan corresponds to an expected optimal cost of 447.5. Note in particular how the production plan partially anticipates the demand in stage 3 by increasing the production in stage 2 to avoid carrying out some costly setups at nodes 4 and 6. However, in the obtained production plan, the total production costs

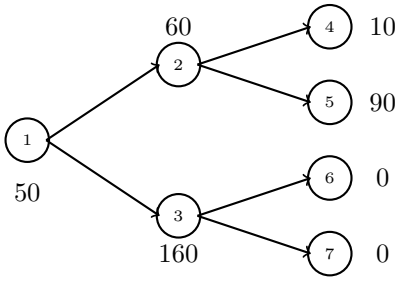
in the worst-case scenarios, which correspond to the paths $\mathcal{P}(1,7)$ and $\mathcal{P}(1,5)$ in the scenario tree, are equal to 540 and 510 respectively, which is significantly higher than the expected value of 447.5. This may not be satisfying for a risk-averse production planner.

Figure 7.3: Optimal production plan for the illustrative example using the risk-neutral formulation



Risk-averse production plan using a scenario-wise risk function In order to propose a production plan taking into account this risk aversion, we first solve the SULS using the formulation (7.15)-(7.21) based on the scenario-wise risk function. Figure 7.4 describes the obtained production plan.

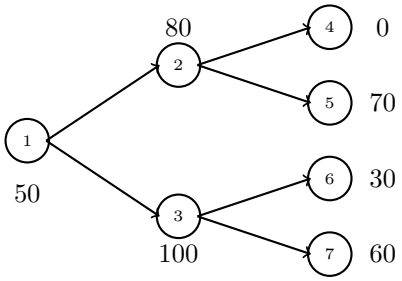
Figure 7.4: Optimal production plan given by the risk-averse formulation based on a scenario-wise risk function



This solution corresponds to an optimal aggregated risk equal to $\text{CVaR}_{0.95}(Z_1 + Z_2 + Z_3) = 500$. This value is computed using a value-at-risk equal to $\varphi = 500$ and an expected overcost, over the 4 scenarios, equal to 0. Namely, the cost of each individual scenario $\mathcal{P}(1,\ell), \ell \in \{4,5,6,7\}$, is equal to $\{420, 500, 500, 470\}$, which gives $v^\ell = 0$ for each ℓ in $\{4,5,6,7\}$. Note in particular how the total cost in scenario $\mathcal{P}(1,7)$ has been reduced from 540 in the previous solution to 470 in the current one through a full anticipation at node 3 of the future demand at node 7. Similarly, the production quantity at node 2 has been reduced from 70 to 60 in order to reduce the inventory holding costs in scenario $\mathcal{P}(1,5)$. However, these two changes negatively impact the costs on the two remaining scenarios, $\mathcal{P}(1,4)$ and $\mathcal{P}(1,6)$, as there is now some useless leaving inventory at node 6 and an additional setup at node 4. This translates in an increase of the expected total cost from 447.5 in the risk-neutral solution to 472.5 in this solution, which illustrates among others how the risk reduction may come at the expense of an increase in the expected cost of a production plan.

Risk-averse production plan using a stage-wise risk function Let us now consider the production plan obtained by solving the risk averse formulation (7.25)-(7.28) based on a stage-wise risk function. Figure 7.5 describes the obtained production plan.

Figure 7.5: Optimal production plan given by the risk-averse formulation based on a stage-wise risk function

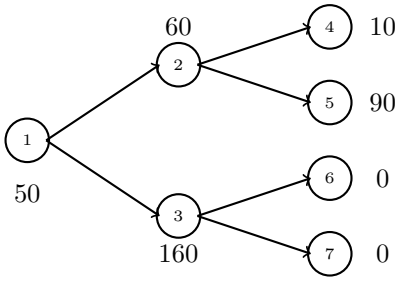


This solution gives a total risk value equal to $Z_1 + \text{CVaR}_{0.95}(Z_2) + \text{CVaR}_{0.95}(Z_3) = 520$ obtained as follows. At stage 1, the risk corresponds to the production cost at node 1, which is equal to 150. We then have $\text{CVaR}_{0.95}(Z_2) = 200$ with $\varphi_2 = 200$, $v^2 = v^3 = 0$. Finally, we have $\text{CVaR}_{0.95}(Z_3) = 170$ with $\varphi_3 = 170$ and $v^4 = v^5 = v^6 = v^7 = 0$. Note how the risk value at stage 2 is mainly driven by the production costs at node 3 whereas the risk value at stage 3 is determined by the production costs at node 5, which is not in the sub-tree rooted at node 3. As the stage-wise risk function considers the risk stage by stage, it overlooks the fact that the largest demand at stage 2 does not happen in the same scenario as the largest demand at stage 3. This allows to gain some intuition about the reason why the risk value provided by this model is larger than the one provided by the model based on the scenario-wise risk measure.

Moreover, as in the previous risk-averse model, we note that the expected cost of the production plan displayed in Figure 7.5 is larger (467.5) than the one obtained with the risk-neutral model (447.5).

Risk-averse production plan using a nested risk function Finally, we provide the production plan obtained with the risk-averse formulation (7.41)-(7.45) based on a nested risk function. This one is displayed in Figure 7.6

Figure 7.6: Optimal production plan given by the risk-averse formulation based on a nested risk function



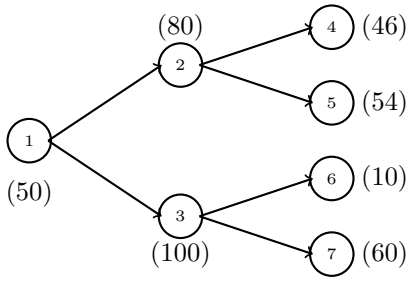
This solution corresponds to a total risk value equal to $Z_1 + \text{CVaR}_{0.95}\Big(Z_2 + \text{CVaR}^{\xi_{[2]}}_{0.95}(Z_3)\Big) = 500$. This value is obtained as follows. The production cost at stage 1 is equal to 150. The aggregated risk value $\text{CVaR}_{0.95}\Big(Z_2 + \text{CVaR}^{\xi_{[2]}}_{0.95}(Z_3)\Big)$ is equal to 350, with $\varphi^1_1 = 350$, $v^2 = v^3 = 0$. Moreover, the conditional one-stage CVaR at node 2, $\text{CVaR}^2_{0.95}(Z_3)$, is equal to 190 with $\varphi^2_3 = 190$ and $v^4 = v^5 = 0$ and the conditional one-stage CVaR at node 3, $\text{CVaR}^3_{0.95}(Z_3)$, is equal to 190 with $\varphi^2_3 = 30$ and $v^6 = v^7 = 0$.

We note in particular that, in the formulation (7.41)-(7.45) , the overcost value computed at node 3, v^3, does not depend on the value of conditional one-stage CVaR at node 2, $\text{CVaR}^2_{0.95}(Z_3)$. This allows the model to build a production plan in the sub-tree rooted in 3 without taking into account the large value of the demand arising at node 5 and to provide a production plan with an overall risk value smaller than the one obtained with the model based on a stage-wise risk measure.

7.5.2 Illustration of time inconsistent production plans

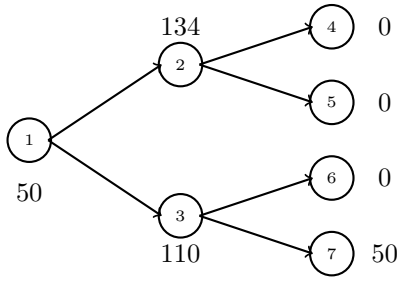
In order to illustrate the time inconsistency issues that may arise with some risk-averse formulations, we now study a second small instance of the SULS. This one is similar to the one discussed above and only differs with respect to the value of the demand at each node (see Figure 7.7) and the value of the confidence level which is set to $\alpha = 0.5$.

Figure 7.7: Small illustrative example. Value between brackets represent the demand d^n a each node of the scenario tree.



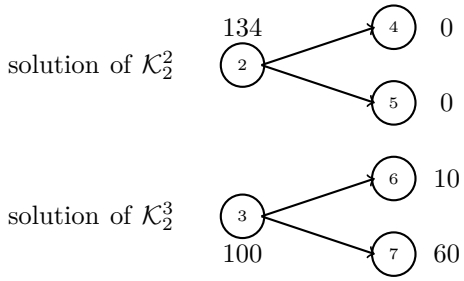
Risk-averse production plan using a scenario-wise risk function We first solve the production planning problem at the beginning of period 1 using the formulation (7.15)-(7.21) based on the scenario-wise risk function and obtain the production plan presented in Figure 7.8. It corresponds to an optimal aggregated risk equal to $\text{CVaR}_{0.95}(Z_1 + Z_2 + Z_3) = 483$. This value is obtained with a value-at-risk $\varphi = 446$ corresponding to the total cost of scenario $\mathcal{P}(1, 4)$ and overcosts equal to $v^4 = v^5 = v^6 = 0$ and $v^7 = 74$.

Figure 7.8: Optimal production plan computed at the beginning of period 1 using a scenario-wise risk measure



Let us now move forward one time period and consider the production planning problem at the beginning of time period 2, once the demand for period 2 has been observed. Since $\hat{X}^1 = 50$, there is no stock at the beginning of period 2 and we have to determine how many units to produce at time periods 2 and 3 in order to satisfy the demand. Figure 7.9 describes the production plans obtained in case $d_2 = 80$ (Problem $\mathcal{K}^2_2$) and in case $d_2 = 100$ (Problem $\mathcal{K}^3_2$).

Figure 7.9: Optimal production plans computed at the beginning of period 2 using a scenario-wise risk function



We note that the solution of Problem $\mathcal{K}_2^2$ coincides with the one of Problem $\mathcal{K}$ given in Figure 7.8. The optimal production plan computed at the beginning of period 2 is the same as the optimal production plan computed at the beginning of period 1 for this part of scenario tree.

This is however not the case for the solution of Problem $\mathcal{K}_2^3$ which completely differs from the one of Problem $\mathcal{K}$ for this part of the scenario tree. The solution of $\mathcal{K}_2^3$ corresponds to a risk value equal to $\text{CVaR}_{0.95}(Z_2 + Z_3) = 360$ with a value-at-risk equal to 310 and an overcost at node 6 equal to 50. We note in particular that:

- The production plan computed at the beginning of period 1 for the sub-tree $\mathcal{V}(3)$ is not optimal for Problem $\mathcal{K}_2^3$ as it leads to a higher risk value equal to 370.
- The production plan computed at the beginning of period 2 for $\mathcal{V}(3)$ is not optimal for the initial problem $\mathcal{K}$ as it leads to an overvall risk value equal to 485.

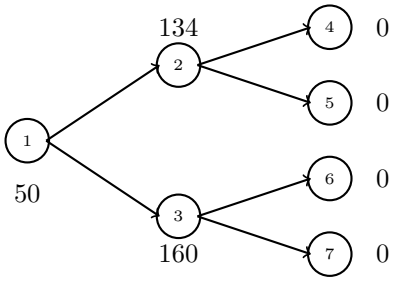
The IOP thus does not hold in this case. This is intuitively explained as follows. In the solution of Problem $\mathcal{K}$ displayed in Figure 7.8, the value of the production quantity at node 3 is mainly determined by the fact that we seek to limit the overcost in the scenarios $\mathcal{P}(1,6)$ and $\mathcal{P}(1,7)$ with respect to a value-at-risk φ whose value is determined by the cost of scenario $\mathcal{P}(1,4)$. Yet, when we are at node 3, we already know that the state described by 4 will not happen as node 4 does not belong to $\mathcal{V}(3)$. We are thus letting a production decision relative to node 3 depend on states of the stochastic demand process that cannot happen in the future. This is contradiction with the principle of time consistency defined in [93] which defines that at every state of the system, the optimal decision should not depend on scenarios which we already know cannot happen in the future.

Risk-averse production plan using a stage-wise risk function We now illustrate that the same issue may arise when using a stage-wise risk function. We first solve the production planning problem at the beginning of period 1 using the formulation (7.25)-(7.28) based on a stage-wise risk function. Figure 7.10 describes the obtained production plan. This one corresponds to a total risk of $Z_1 + \text{CVaR}_{0.95}(Z_2) + \text{CVaR}_{0.95}(Z_3) = 499$ obtained as follows. The cost relative to stage 1 is equal to 150. The risk at stage 2 is equal to $\text{CVaR}_{0.95}(Z_2) = 320$ with $\varphi_2 = 320$ and $v^2 = v^3 = 0$. The risk a stage 3 is equal to $\text{CVaR}_{0.95}(Z_3) = 29$ with $\varphi_3 = 8$ and $v^5 = v^7 = 0$ and $v^6 = 42$.

Let us now move forward one time period and consider the production planning problem at the beginning of period 2, once the demand for period 2 has been observed. Since $\hat{X}^1 = 50$, there is no stock at the beginning of period 2 and we have to determine how many units to produce at time periods 2 and 3 in order to satisfy the demand. Figure 7.11 describes the production plans obtained in case $d_2 = 80$ (Problem $\mathcal{K}_2^2$) and in case $d_2 = 100$ (Problem $\mathcal{K}_2^3$).

We note that the solution of Problem $\mathcal{K}_2^2$ coincides with the one of Problem $\mathcal{K}$ given in Figure 7.10. This is however not the case for the solution of Problem $\mathcal{K}_2^3$ which differs from the production plan computed for $\mathcal{V}(3)$ by Problem $\mathcal{K}$. This production plan corresponds to a risk value of $Z_2 +$

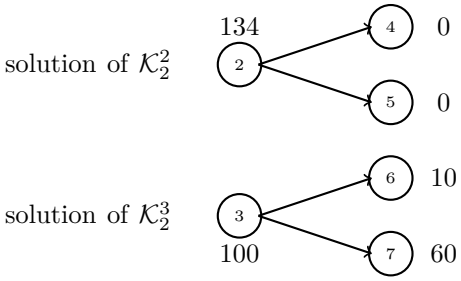
Figure 7.10: Optimal production plan computed at the beginning of period 1 using a stage-wise risk function



$\text{CVaR}_{0.95}(Z_3) = 360$ with a cost equal to 200 at stage 2 and a risk value relative to stage 3 equal to $\text{CVaR}_{0.95}(Z_3) = 160$. This latter is obtained with a value-at-risk equal to $\varphi_3 = 160$ and overcosts $v^6 = 0$ and $v^7 = 0$. We note in particular that:

- The production plan computed at the beginning of period 1 for the sub-tree $\mathcal{V}(3)$ is not optimal for Problem $\mathcal{K}_2^3$ as it leads to a higher risk value equal to 370.
- The production plan computed at the beginning of period 2 for $\mathcal{V}(3)$ is not optimal for the initial problem $\mathcal{K}$ as it leads to an overall risk value equal to 573.

Figure 7.11: Optimal production plans computed at the beginning of period 2 using a stage-wise risk function

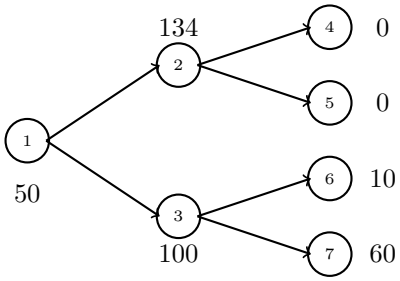


The IOP thus does not hold in this second case. The explanation is similar to the one provided above for the scenario-wise risk function. Namely, in the production plan presented in Figure 7.10, the production planning decisions at node 3, and in particular the fact that the production of all the demand relative to nodes 6 and 7 is anticipated at node 3, is driven by the need to decrease as much as possible the overcosts at nodes 6 and 7 with respect to the value-at-risk $\varphi_3 = 8$ whose value is determined by the cost observed at node 4. As node 4 does not belong to $\mathcal{V}(3)$, it means that we are again letting a production decision at node 3 depend on states of the stochastic demand process that cannot happen in the future as seen from node 3. This goes against the time consistency principle as enunciated in [93].

Risk-averse production plan using a nested risk function Finally, we solve the same production planning problem using with the formulation (7.41)-(7.45) based on a nested risk function. This one is displayed in Figure 7.12

This solution corresponds to a total risk value equal to $Z_1 + \text{CVaR}_{0.95}\Big(Z_2 + \text{CVaR}_{0.95}^{\xi_{[2]}}(Z_3)\Big) = 510$. This value is obtained as follows. The production cost at stage 1 is equal to 150. The aggregated

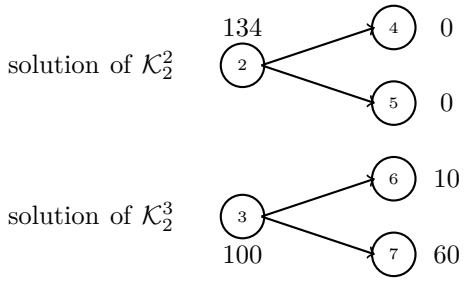
Figure 7.12: Optimal production plan given by the risk-averse formulation based on a nested risk function



risk value $\text{CVaR}_{0.95}\Big(Z_2 + \text{CVaR}^{\xi_{[2]}}_{0.95}(Z_3)\Big)$ is equal to 360, with $\varphi^1_2 = 296$ and $v^2 = 0$ and $v^3 = 64$. Moreover, the conditional one-stage CVaR at node 2, $\text{CVaR}^2_{0.95}(Z_3)$, is equal to 180 with $\varphi^2_3 = 0$ and $v^4 = 8$ and $v^5 = 0$ and the conditional one-stage CVaR at node 3, $\text{CVaR}^3_{0.95}(Z_3)$, is equal to 160 with $\varphi^3_3 = 160$, $v^6 = 0$ and $v^7 = 0$.

Let us now move forward one time period and consider the production planning problem at the beginning of period 2, once the demand for period 2 has been observed. Since $\hat{X}^1 = 50$, there is no stock at the beginning of period 2 and we have to determine how many units to produce at time periods 2 and 3 in order to satisfy the demand. Figure 7.13 describes the production plans obtained in case $d_2 = 80$ (Problem $\mathcal{K}^2_2$) and in case $d_2 = 100$ (Problem $\mathcal{K}^3_2$).

Figure 7.13: Optimal production plans computed at the beginning of period 2 using a nested risk function



We note that the solutions of Problems $\mathcal{K}^2_2$ and $\mathcal{K}^3_2$ coincide with the one of Problem $\mathcal{K}$ given in Figure 7.13. This means that the IOP holds in the present case, which is line with the fact that the nested risk measure used in the formulation is time-consistent.

Finally, note that we omit the illustration of the production plans provided by the expected conditional risk measure (7.8) as it provides the same solution as the nested risk measure (7.7) for the considered small numerical example.

7.6 Computational experiments

In this section, we present some preliminary computational results aiming at assessing the relative performance of the production plans provided by each risk-averse model when implemented within a rolling horizon framework. This assessment is carried out through simulation and focuses on evaluating both the obtained risk reduction, i.e. the decrease of the cost observed in the worst cases, and the potential deterioration of the average cost.

We first present the rolling horizon framework and the experimental setup used in our simulations. We then discuss our preliminary computational results.

7.6.1 Rolling horizon simulation

In order to obtain a quantitative assessment of the performance of the risk averse models discussed in this chapter, we rely on a rolling horizon simulation similar to the one used in [23].

Each simulation run consists in first generating a scenario representing a possible evolution of the stochastic input process over a simulation horizon involving Σ' stages. We then iteratively simulate, for each simulated stage $\sigma' = 1...\Sigma'$, the application of the first-stage decisions computed by a risk-averse or risk-neutral model which uses a production planning horizon involving Σ stages. At each simulated stage σ', we record the cost $C^{true}(\sigma')$ incurred by the application of the first-stage planning decisions over the simulated scenario: note that this cost does not correspond to the objective function of the optimization model but to the true cost of implementing the first-stage decisions computed by the optimization model when the realization of the stochastic parameters are equal to the value corresponding to the stage σ' of the simulated scenario. We finally compute the total 'true' cost over all simulated stage $C^{true} = \sum_{\sigma'=1}^{\Sigma'} C^{true}(\sigma')$.

More precisely, each simulation run consists of the following steps:

1. Generate a simulated 'true' scenario involving Σ' stages which will represent the actual evolution of the stochastic process over the simulation horizon.

2. For each $\sigma' = 1...\Sigma'$:

   2.1 Generate a scenario tree involving Σ stages and $|\mathcal{V}|$ nodes in which the nodes relative to the first stage of the scenario tree correspond to the same realization of the stochastic parameters as the nodes corresponding to stage σ' of the simulated scenario.

   2.2 For each considered production planning model:

   - solve the corresponding MILP formulation over the scenario tree $\mathcal{V}$ taking into account a non-negative entering inventory at node 1.
   - simulate the implementation of the production planning decisions corresponding to the first stage of $\mathcal{V}$ in the situation described by stage σ' of the 'true' scenario. Then, compute the corresponding true cost $C^{true}(\sigma')$ and record the non-negative leaving inventory.

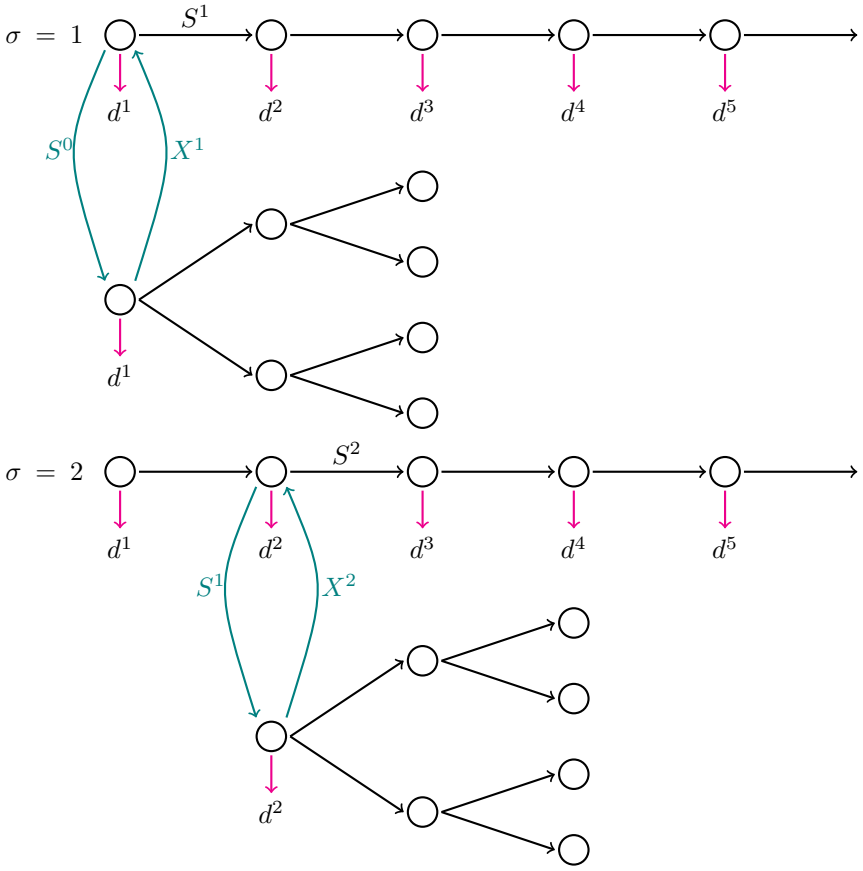
3. Compute the total simulated cost $C^{true} = \sum_{\sigma'=1}^{\Sigma'} C^{true}(\sigma')$ for each considered production planning model.

Note that, at each iteration of the rolling horizon simulation, each production planning model is solved using exactly the same scenario tree $\mathcal{V}$ to represent the possible future evolution of the stochastic process over the next Σ stages. Thus, any difference in the first-stage solutions provided by the various production planning models may only be due to the fact that they differ with respect to their (risk-neutral or risk-averse) objective function. Figure 7.14 illustratives this simulation procedure in case $\Sigma' = 5$, $\Sigma = 3$ and $|\mathcal{V}| = 7$.

7.6.2 Experimental setup

We use a simulation horizon involving $\Sigma' \in \{6, 12, 18\}$ stages with a single period per stage. Each simulated true scenario is randomly generated as follows. We consider deterministic and time-invariant costs and set the setup cost to $f = 1200$, the production cost to $g = 2$ and the inventory holding cost to $h = 1$. The demand $d_{\sigma'}, \sigma' = 1...\Sigma'$, is assumed to be uniformly distributed in the

Figure 7.14: Illustration of a rolling horizon simulation



interval [0,100]. For each simulation horizon length, we generated 400 true scenarios, resulting in a total of 1200 true scenarios

At each iteration of a simulation run, i.e. at each simulation stage σ', we generate a scenario tree to represent the future evolution of the stochastic process over the next stages. This scenario tree involves $\Sigma = 7$ stages, $b = 1$ time period per stage and $c = 3$ children for each non-leaf node, which corresponds to $|\mathcal{L}(1)| = 729$ individual scenarios and $|\mathcal{V}| = 1093$ nodes. The demand at node 1 of this scenario tree, d^1, is set to the value $d_{\sigma'}$ observed at stage σ' of the simulated scenario. The demand for nodes 2 to $|\mathcal{V}|$ is generated following the same procedure as the one used to generate the simulated true scenarios.

Finally, we consider two different values, $\alpha \in \{0.5, 0.95\}$, for the confidence level used to define the CVaR-based risk functions used in the various risk-averse production planning models.

For each generated true scenario and each considered value of α, we carried out one simulation run. Note that the corresponding computational effort is very significant. Namely, each simulation run requires to solve Σ' mixed-integer linear programs, and this for each of the 5 studied production planning models. This means that we solve a total of 144000 mixed-integer linear programs, each one corresponding to a SULS problem formulated on a scenario tree involving $|\mathcal{V}| = 1093$ nodes.

This simulation procedure was implemented in C++. All mixed-integer linear programs were defined using the Concert Technology environment and solved using the MILP solver CPLEX 12.8. All the parameters of the solver were kept to their default value, except for the computation time limit which was set to 600 seconds and the relative MIP gap tolerance which was set to 1%. All tests were run on the computing infrastructure of the Laboratoire d'Informatique de Paris VI (LIP6), which consists of a cluster of Intel Xeon Processors X5690. We set the cluster to use two 3.46GHz cores and 12GB RAM to solve each instance.

7.6.3 Preliminary computational results

Tables 7.1-7.4 display the preliminary computational results of our rolling horizon simulations. In each table, Column `Model` indicates the corresponding production planning model: `RN` represents the risk-neutral formulation (5.1)-(5.4), `A-CVaR` the risk-averse formulation (7.25)-(7.28) using an aggregated CVaR risk measure, `S-CVaR` the risk-averse model (7.15)-(7.21) using a stage-wise CVaR risk measure, `N-CVaR` the risk-averse model (7.41)-(7.45) using a nested CVaR risk measure and `E-CVaR` the risk-averse model (7.49)-(7.52) using an expected CVaR risk measure.

Table 7.1 focuses on assessing the quality of the production plans obtained by solving the MILP formulation corresponding to each considered production planning problem. This quality is evaluated by looking at the optimality gap, i.e. at the relative difference between the best found integer feasible solution and the best known lower bound when the computation time limit is reached. We thus report, for each production planning model and each value of the confidence level α, the average optimality gap over all corresponding solved MILPs in Column `Ave.MIP`$_{Gap}$ together with the standard deviation in Column `Std.Dev.` and the maximum observed gap in Column `Max.MIP`$_{Gap}$. Overall, these results show that the solutions of the production planning problems solved over the course of a simulation run are of good quality as they are on average within less than 1.5% of the optimal solution value. Thus, even if each production planning problem is not solved exactly in our simulation procedure to reduce the total computational effort, the obtained sub-optimal solutions seem to be of a quality sufficiently high to avoid introducing large bias in our analysis.

Table 7.1: Quality of the solutions obtained for each production planning problem within the allowed computation time

α	Model	Ave. MIP$_{Gap}$	Std. Dev.	Max. MIP$_{Gap}$
0.50	RN	1.54	0.27	2.56
	A-CVaR	2.28	0.43	4.39
	S-CVaR	2.65	0.48	4.68
	N-CVaR	1.77	0.84	3.86
	E-CVaR	1.40	0.26	2.79
0.95	RN	1.55	0.29	2.65
	A-CVaR	0.87	0.10	1.53
	S-CVaR	0.98	0.02	1.00
	N-CVaR	1.69	0.76	3.67
	E-CVaR	0.44	0.12	0.86

We now focus on assessing the ability of each considered risk-averse production planning model at actually reducing the risk when implemented in a rolling horizon framework. This ability is measured by comparing, for each risk-averse model, $C_{RA}^{\mathtt{x}\%}$, the cost over the $\mathtt{x}\%$ most costly true scenarios obtained with this model, with $C_{RN}^{\mathtt{x}\%}$, the cost over the $\mathtt{x}\%$ most costly true scenarios obtained with the risk-neutral model. Tables 7.2-7.4 thus report the value of the relative difference $RD^{\mathtt{x}\%} = 100(C_{RA}^{\mathtt{x}\%} - C_{RN}^{\mathtt{x}\%})/C_{RN}^{\mathtt{x}\%}$ for $x \in \{5, 10, 20, 30, 40, 50, 100\}$ for each risk-averse model and each value of α.

Results from Tables 7.2-7.4 first suggest that using the studied risk-averse production planning models in a rolling horizon dynamic framework does not always lead to the risk reduction one may have hoped for when formulating them in a static setting. This can be seen e.g. by looking at the results displayed in Column $RD^{50\%}$ of these tables for the simulations carried out with $\alpha = 0.5$. Indeed, we note that $RD^{50\%}$ takes an average positive value of 2.5%, which means that the average cost over the 50% worst-case true scenarios increases when using a risk-averse model instead of a risk-neutral model to plan production, and this even if the risk-averse models relied on CVaR$_{0.50}$-based risk functions seeking to reduce the risk over the 50% worst-case true scenarios.

Table 7.2: Performance of each risk-averse model within a rolling horizon simulation with $\Sigma' = 6$.

α	Model	$RD^{5\%}$	$RD^{10\%}$	$RD^{20\%}$	$RD^{30\%}$	$RD^{40\%}$	$RD^{50\%}$	$RD^{100\%}$
0.5	A-CVaR	2.61	-1.78	-10.67	-7.13	-4.37	-2.47	1.87
	S-CVaR	2.78	-5.13	-11.76	-7.20	-4.05	-1.74	3.47
	N-CVaR	1.80	0.58	-2.77	1.37	3.70	5.31	6.15
	E-CVaR	2.47	0.22	-7.90	-4.02	-1.25	0.79	4.55
0.95	A-CVaR	-7.19	-6.22	-5.16	2.60	7.30	10.12	2.89
	S-CVaR	-6.06	-6.73	-7.40	-0.45	3.89	6.97	3.70
	N-CVaR	-4.89	-4.91	-3.99	3.53	7.93	9.94	3.01
	E-CVaR	-2.31	-2.76	-2.82	4.24	8.01	9.01	3.23

However, the results obtained for a confidence level of 95% are more encouraging. Namely, the average value of $RD^{5\%}$ over all simulations carried out with $\alpha = 0.95$ is equal to -2.7%, which means that the average cost over the 5% worst-case true scenarios decreases when using a risk-averse model instead of a risk-neutral model to plan production. We observe that this reduction is obtained at the expense of an increase in the expected cost, which can be measured by the fact that the average value of $RD^{100\%}$, over all the simulation runs corresponding to $\alpha = 0.95$, is equal to 0.84%.

It is also relevant to mention that the cost reduction in the worst-case scenarios is smaller for the planning horizon with 12 and 18 stages than the one obtained with a shorter planning horizon involving only 6 stages. This might be explained by the size of the scenario tree used to approximate the stochastic process. We only consider scenario trees with $\Sigma = 7$ stages and 3 children par stage. Hence, the risk-averse models might provide first-stage solutions too conservative because they might not be able to see that they could compensate their first-stage decisions in further stages.

Additional computational experiments and a thorough in-depth analysis of the production planning decisions made by each type of model are thus needed to better understand the behavior observed in these preliminary results.

Table 7.3: Performance of each risk-averse model within a rolling horizon simulation with $\Sigma' = 12$.

α	Model	$RD^{5}\%$	$RD^{10\%}$	$RD^{20\%}$	$RD^{30\%}$	$RD^{40\%}$	$RD^{50\%}$	$RD^{100\%}$
0.5	A-CVaR	-2.86	-5.86	-3.57	-0.91	0.63	1.63	3.35
	S-CVaR	-4.81	-6.11	-2.64	0.27	1.99	3.09	4.91
	N-CVaR	0.71	-0.61	1.67	4.06	5.23	5.82	4.73
	E-CVaR	-2.94	-4.82	-1.82	0.81	2.37	3.31	4.04
0.95	A-CVaR	-4.10	-4.77	-1.47	1.69	3.24	4.10	-1.34
	S-CVaR	2.02	1.18	0.69	1.92	2.62	3.25	0.23
	N-CVaR	-3.10	-3.64	-0.50	2.50	3.93	4.60	-0.76
	E-CVaR	-1.42	-2.38	0.32	2.93	3.84	3.96	-0.02

7.7 Conclusion and perpsectives

In this chapter, we presented four alternative risk-averse formulations of the stochastic single-echelon uncapacitated lot-sizing problem. All these formulations sought to minimize the risk that the actual cost of the production plan might be much higher than the expected cost computed by the model. They differed with respect to the way risk may be measured in a multi-stage setting. We focused on assessing the performance in terms of risk reduction of the four studied production planning models when implemented in a rolling horizon framework. Our preliminary results suggest that the risk reduction observed in a rolling horizon dynamic setting may not always be as significant as one may

Table 7.4: Performance of each risk-averse model within a rolling horizon simulation with $\Sigma' = 18$

α	Model	$RD^{5\%}$	$RD^{10\%}$	$RD^{20\%}$	$RD^{30\%}$	$RD^{40\%}$	$RD^{50\%}$	$RD^{100\%}$
0.5	A-CVaR	-4.13	-4.09	-0.61	1.04	2.01	2.58	1.62
	S-CVaR	-2.93	-2.58	1.15	2.71	3.63	4.21	2.30
	N-CVaR	0.83	0.39	3.12	4.09	4.42	4.43	2.25
	E-CVaR	-3.01	-2.72	0.89	2.43	3.20	3.63	1.49
0.95	A-CVaR	0.09	-1.38	0.53	1.44	1.99	2.05	-1.10
	S-CVaR	1.27	1.73	3.82	3.00	2.48	2.07	0.36
	N-CVaR	0.09	-0.80	1.21	2.07	2.57	2.55	-0.47
	E-CVaR	0.26	0.28	2.34	2.90	3.00	2.69	0.45

have hoped for when considering these risk-averse models in a static setting. Additional research work is thus needed to better understand this behavior, to determine which risk function may have the best practical performance and assess the potential advantages and disadvantages of using a time-consistent rather than a time-inconsistent risk function. We also consider that extending this analysis to more complex stochastic lot-sizing problem, such as the lot-sizing problem with remanufacturing studied in Chapters 3 and 6 is also worth investigating.

Chapter 8

Conclusion and perspectives

8.1 Conclusion

In this thesis, we studied two lot-sizing problems: a multi-echelon lot-sizing problem with remanufacturing and lost sales and the uncapacitated single-echelon lot-sizing problem. Our main purpose was to develop models and solution approaches for the first problem. But the latter provided us a useful basis to develop novel algorithms and explore new risk-averse formulations.

For these two combinatorial optimization problems, we considered a stochastic environment in which the input data 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 large size mixed-integer linear programs, which cannot be solved directly by a state-of-the-art commercial MILP solver. We therefore investigated advanced MILP based solution approaches to solve these challenging stochastic lot-sizing problems.

We first investigated the underlying polyhedral structure of the multi-echelon stochastic problem with remanufacturing and lost sales. We proposed a new class of valid inequalities (the tree inequalities (2.13) presented in Chapter 3) that, when used as cutting-planes in a branch-and-cut algorithm, are able to solve medium-size instances of the multi-echelon stochastic lot-sizing problem with remanufacturing to optimality. It is worth noting that the proposed tree inequalities are valid not only for the stochastic lot-sizing problems with remanufacturing under study here but also for a larger class of capacitated stochastic lot-sizing problems with lost sales. These results were published in *Computers & Operations Research* and can be found in [83].

The production system structure of the multi-echelon lot-sizing problem with remanufacturing is then further investigated in Chapter 4. We proposed several new families of valid inequalities in order to strengthen the formulation of the problem. In particular, we introduced two new classes of valid inequalities, Inequalities (4.33), (4.34) and (4.41) and Inequalities (4.43)-(4.46), which take into account, at each production echelon, the limitations on the produced quantities coming from the limited availability of the returned products. The results of our computational experiments showed that a branch-and-cut algorithm based on these new valid inequalities performs well as compared to the generic branch-and-cut algorithm of CPLEX 12.8 solver and to the branch-and-cut algorithm based on the path inequalities (2.12) investigated in Chapter 3. It is worth mentioning that Inequalities (2.12) can be seen as a particular case of Inequalities (4.43),(4.44),(4.46) and that the proposed new inequalities thus extend previous results on lot-sizing problem with lost sales. These results will be presented as a full paper at the *International Conference on Computational Logistics 2021* and will be published in Springer's *Lecture Notes in Computer Science*.

Although our computational results showed that the proposed branch-and-cut algorithms were able to solve to near-optimality medium-size instances of the problem, some numerical difficulties could be observed for larger instances. This motivated the development of decomposition-based solution approaches such as the recently proposed SDDiP algorithm [105].

In view of the complexity and novelty of this algorithm, we first focused on applying the SDDiP algorithm on the stochastic uncapacitated single-echelon lot-sizing problem (SULS) which is much simpler than our original problem. We proposed in Chapter 5 a new extension of this algorithm wich mainly relies on a partial decomposition of the scenario tree into sub-trees and on the exploitation of existing knowledge on the polyhedral structure of the SULS problem. Our computational results showed that the proposed extended SDDiP algorithm significantly outperforms the initial SDDiP algorithm in terms of solution quality on large-size instances. A preliminary version of this work was presented at the *International Conference on Automated Planning and Scheduling 2019* [82] and the completed version has been accepted for publication in *INFORMS Journal on Computing*.

We then came back to the multi-echelon lot-sizing problem with remanufacturing and lost sales and proposed in Chapter 6 a dual dynamic decomposition approach to solve (very) large size instances of this problem. In particular, we adapted the extSDDiP algorithm introduced in Chapter 5 with the main objective of reducing the computational burden linked to the resolution of the sub-problems obtained after the partial decomposition of the original problem. Numerical results showed that the algorithm is capable of obtaining near-optimal solutions in practicable computation times and outperforms both the mathematical programming solver CPLEX 12.8 and the initial SDDiP algorithm. A preliminary version of this work obtained a best paper award at the *6^{th} International Conference on Control, Decision and Information Technologies (CoDIT 2019)* [81]. More recent results have been accepted for presentation at the *International Conference on Advances in Production Management Systems (APMS 2021)*.

Finally, we presented an on-going exploratory work on risk-averse multi-stage stochastic lot-sizing. In this work, we studied several ways of incorporating the risk aversion of the decision-maker in the multistage stochastic uncapacitated lot-sizing (SULS) problem and focused on the use of CVaR-based risk measures. For each considered multi-stage risk function, we showed how to reformulate the problem as a computationally tractable mixed-integer linear program. We then presented some preliminary computational results obtained by carrying out rolling horizon simulations. These results indicated that using a risk-averse model rather than a risk-neutral model does not always appear to lead to a clear risk reduction when it is implemented in a rolling horizon framework. Additional research work is thus needed to better understand this behavior.

8.2 Perspectives

The work presented in this thesis opens several research directions.

Regarding the cutting-plane generation based approaches, it would first be interesting to develop an exact separation algorithm for the (ℓ, k, U) inequalities introduced in Chapter 4. It may namely further improve their performance when used in a branch-and-cut algorithm. Second, these inequalities have shown their usefulness to solve the problem in a deterministic setting. It might thus be worth investigating whether they could also be helpful to solve its stochastic variant. On a longer term perspective, we could seek to exploit the idea underlying the expression of these inequalities to develop valid inequalities for a larger class of lot-sizing problems with returns, such as e.g. lot-sizing problems with backlogging, safety stocks or minimum production levels.

As for the decomposition-based approaches, several interesting research directions might be studied.

First, at each iteration of the proposed extSDDiP algorithm, we generate a single strengthened Benders' cut per macro-stage during the backward step. However, it would be possible to exploit the existence of alternative MILP formulations of the sub-problems to generate several strengthened Benders' cuts for the same macro-stage during a given iteration of the algorithm. This multi-cut variant of the extSDDiP algorithm is worth investigating as it might lead to a faster convergence. Similarly, in the sub-tree based algorithm, the solution obtained at each macro-stage provides a value of the leaving inventory at each leaf nodes of the considered sub-tree. Currently, only the value of

the leaving inventory at the leaf node belonging to the sampled scenario is used in the backward step to generate cuts. However, it would be possible to generate more cuts by considering the value of the leaving inventory at all or parts of the leaf nodes of the sub-tree. This would also lead to a multi-cut variant of the extSDDiP algorithm. Furthermore, in both cases, the question of how to identify the best cut within the pool of cuts that we can generate at each iteration remains unanswered. Hence, a cut selection strategy will be also worth investigating as it might positively impact the performance of the algorithm.

Another important issue that might be explored is how to efficiently implement a binary expansion of the state variables in order to generate Lagrangian and Integer Optimality cuts. These cuts are known to be tight and guarantee that the SDDiP algorithm converges to an optimal solution when the state variables are restricted to be binary. However, our numerical results carried out with the SDDiP algorithm showed that a binary approximation leads to prohibitive computations times when there are multiple items involved in the production system. Therefore, one direction could be to study a partial binarization of the state variables, in which we would use a binary representation of the state variables for only a few leaf nodes of each sub-problem. Another direction would be to carry out a partial binarization of each state variable, i.e. to use only the first digits of the base 2 representation in the binarization of each stage variables and to use an auxiliary continuous variable for the remaining value. These two ideas would make it possible to generate Integer Optimality and Lagrangian cuts in the backward step of the algorithm, which may potentially improve the quality of the lower bound.

Furthermore, it is worth noting that Chapters 3 and 6 focused on solving the same stochastic lot-sizing problem but used solution approaches relying on different strategies. Chapter 3 presented a solution approach aiming at solving to optimality a deterministic equivalent problem formulated as an MILP, using small to medium-size scenario trees, i.e. using a coarse approximation of the stochastic process. In contrast, in Chapter 6, we sought to solve the same problem using large-size scenario trees providing a better approximation of the stochastic input process, but the obtained solutions were of a lesser quality. This opens the following question. In practice, is it better to solve the problem to near-optimality using a scenario tree of reduced size or to solve it with a good but not optimal quality using a very large scenario tree? We believe that this question is worth investigating and should be addressed through an extensive simulation study comparing the solutions obtained with both approaches.

Finally, there are also some interesting research perspectives regarding the modeling of the remanufacturing planning problem. Many additional realistic features may namely be added in our model in order to further reduce the gap between the academic state of the art and the industrial need. For instance, extending the present work in order to account for production resources with a limited capacity and for stochastic processing times could also be worth investigating. We may also try to consider multiple types of used and/or remanufactured products and additional processes such as sorting and inspecting the used products before disassembling them or hybrid manufacturing/remanufacturing on the reassembly resource.

Bibliography

[1] Nabil Absi, Boris Detienne, and Stéphane Dauzère-Pérès. Heuristics for the multi-item capacitated lot-sizing problem with lost sales. *Computers & Operations Research*, 40(1):264–272, 2013. 56

[2] Nabil Absi, Safia Kedad-Sidhoum, and Stéphane Dauzère-Pérès. Uncapacitated lot-sizing problem with production time windows, early productions, backlogs and lost sales. *International Journal of Production Research*, 49(9):2551–2566, 2011. 56

[3] A. Aggarwal and J. K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41:549–571, 1993. 14, 82

[4] Shabbir Ahmed, Alan J King, and Gyana Parija. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24, 2003. 82

[5] Hyung-Dae Ahn, Dong-Ho Lee, and Hwa-Joong Kim. Solution algorithms for dynamic lot-sizing in remanufacturing systems. *International Journal of Production Research*, 49(22):6729–6748, 2011. 31, 41, 43

[6] Kerem Akartunalı and Ashwin Arulselvan. Economic lot-sizing problem with remanufacturing option: complexity and algorithms. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 132–143. Springer, 2016. 54

[7] Unai Aldasoro, Laureano F Escudero, María Merino, Juan F Monge, and Gloria Pérez. On parallelization of a stochastic dynamic programming algorithm for solving large-scale mixed 0–1 problems under uncertainty. *Top*, 23(3):703–742, 2015. 84

[8] Douglas Alem and Reinaldo Morabito. Risk-averse two-stage stochastic programs in furniture plants. *OR spectrum*, 35(4):773–806, 2013. 130, 132, 139

[9] Douglas Alem, Fabricio Oliveira, and Miguel Carrión Ruiz Peinado. A practical assessment of risk-averse approaches in production lot-sizing problems. *International Journal of Production Research*, 58(9):2581–2603, 2020. 130, 132, 139

[10] Sharifah Aishah Syed Ali, Mahdi Doostmohammadi, Kerem Akartunalı, and Robert van der Meer. A theoretical and computational analysis of lot-sizing in remanufacturing with separate setups. *International Journal of Production Economics*, 203:276–285, 2018. 54

[11] Antonio Alonso-Ayuso, Felipe Carvallo, Laureano F Escudero, Monique Guignard, Jiaxing Pi, Raghav Puranmalka, and Andrés Weintraub. Medium range optimization of copper extraction planning under uncertainty in future copper prices. *European Journal of Operational Research*, 233(3):711–726, 2014. 130

[12] Antonio Alonso-Ayuso, Laureano F Escudero, Monique Guignard, and Andres Weintraub. Risk management for forestry planning under uncertainty in demand and prices. *European Journal of Operational Research*, 267(3):1051–1074, 2018. 130, 131, 136, 137, 143

[13] Mohamed Ali Aloulou, Alexandre Dolgui, and Mikhail Y Kovalyov. A bibliography of non-deterministic lot-sizing models. *International Journal of Production Research*, 52(8):2293–2310, 2014. 14

[14] Pedro Amorim, Douglas Alem, and Bernardo Almada-Lobo. Risk management in production planning of perishable goods. *Industrial & Engineering Chemistry Research*, 52(49):17538–17553, 2013. 130, 132, 139

[15] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999. 130, 133

[16] Imre Barany, Tony Van Roy, and Laurence A Wolsey. Uncapacitated lot-sizing: The convex hull of solutions. In *Mathematical programming at Oberwolfach II*, pages 32–43. Springer, 1984. 14, 17, 86

[17] Imre Barany, Tony J Van Roy, and Laurence A Wolsey. Strong formulations for multi-item capacitated lot sizing. *Management Science*, 30(10):1255–1261, 1984. 94

[18] Joseph Begnaud, Saif Benjaafar, and Lisa A Miller. The multi-level lot-sizing problem with flexible production sequences. *IIE Transactions*, 41(8):702–715, 2009. 54

[19] Emre Berk, Ayhan Özgür Toy, and Öncü Hazır. Single item lot-sizing problem for a warm/cold process with immediate lost sales. *European Journal of Operational Research*, 187(3):1251–1267, 2008. 56

[20] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011. 8, 14

[21] Nadjib Brahimi, Nabil Absi, Stéphane Dauzère-Pérès, and Atle Nordli. Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, 263(3):838–863, 2017. 14, 54, 82

[22] Nadjib Brahimi, Stéphane Dauzere-Peres, Najib M Najid, and Atle Nordli. Single item lot sizing problems. *European Journal of Operational Research*, 168(1):1–16, 2006. 14

[23] Paolo Brandimarte. Multi-item capacitated lot-sizing with demand uncertainty. *International Journal of Production Research*, 44(15):2997–3022, 2006. 49, 151

[24] Lisbeth Buschkühl, Florian Sahling, Stefan Helber, and Horst Tempelmeier. Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *Or Spectrum*, 32(2):231–261, 2010. 14

[25] Victor CB Camargo, Franklina MB Toledo, and Bernardo Almada-Lobo. Hops–hamming-oriented partition search for production planning in the spinning industry. *European Journal of Operational Research*, 234(1):266–277, 2014. 82

[26] Santiago Cerisola and Andrés Ramos. Node aggregation in stochastic nested Benders decomposition applied to hydrothermal coordination. In *PMAPS2000: 6th International Conference on Probabilistic Methods Applied to Power Systems*, volume 1, 2000. 84

[27] Patrick Cheridito, Freddy Delbaen, and Michael Kupper. Coherent and convex monetary risk measures for unbounded cadlag processes. *Finance and Stochastics*, 9(3):369–387, 2005. 133

[28] M Pilar Cristobal, Laureano F Escudero, and Juan F Monge. On stochastic dynamic programming for solving large-scale planning problems under uncertainty. *Computers & Operations Research*, 36(8):2418–2428, 2009. 84

[29] Jesus O Cunha, Ioannis Konstantaras, Rafael A Melo, and Angelo Sifaleras. On multi-item economic lot-sizing with remanufacturing and uncapacitated production. *Applied Mathematical Modelling*, 50:772–780, 2017. 54

[30] Jesus O Cunha and Rafael A Melo. A computational comparison of formulations for the economic lot-sizing with remanufacturing. *Computers & Industrial Engineering*, 92:72–81, 2016. 66

[31] Freddy Delbaen. Coherent risk measures. *Blätter der DGVFM*, 24(4):733–739, 2000. 133

[32] Freddy Delbaen. Coherent risk measures on general probability spaces. In *Advances in finance and stochastics*, pages 1–37. Springer, 2002. 133

[33] Meltem Denizel, Mark Ferguson, et al. Multiperiod remanufacturing planning with uncertain quality of inputs. *IEEE Transactions on Engineering Management*, 57(3):394–404, 2010. 28

[34] Marco Di Summa and Laurence A Wolsey. Lot-sizing on a tree. *Operations Research Letters*, 36(1):7–13, 2008. 29, 82

[35] Andreas Drexl and Alf Kimms. Lot sizing and scheduling—survey and extensions. *European Journal of operational research*, 99(2):221–235, 1997. 14

[36] Laureano F Escudero, Juan Francisco Monge, and Dolores Romero Morales. An sdp approach for multiperiod mixed 0–1 linear programming models with stochastic dominance constraints for risk management. *Computers & Operations Research*, 58:32–40, 2015. 130

[37] Laureano F Escudero, Juan Francisco Monge, and Dolores Romero Morales. On the time-consistent stochastic dominance risk averse measure for tactical supply chain planning under uncertainty. *Computers & Operations Research*, 100:270–286, 2018. 84

[38] Chang Fang, Xinbao Liu, Panos M Pardalos, Jianyu Long, Jun Pei, and Chao Zuo. A stochastic production planning problem in hybrid manufacturing and remanufacturing systems with resource capacity planning. *Journal of Global Optimization*, 68(4):851–878, 2017. 29, 48, 50

[39] Hans Föllmer and Alexander Schied. Convex measures of risk and trading constraints. *Finance and stochastics*, 6(4):429–447, 2002. 133

[40] Francesco Gaglioppa, Lisa A Miller, and Saif Benjaafar. Multitask and multistage production planning and scheduling for process industries. *Operations Research*, 56(4):1010–1025, 2008. 54

[41] Ali Ghamari and Hadi Sahebi. The stochastic lot-sizing problem with lost sales: A chemical-petrochemical case study. *Journal of Manufacturing Systems*, 44:53–64, 2017. 82

[42] Ralf Gollmer, Uwe Gotzes, and Rüdiger Schultz. A note on second-order stochastic dominance constraints induced by mixed-integer linear recourse. *Mathematical Programming*, 126(1):179–190, 2011. 130

[43] Ralf Gollmer, Frederike Neise, and Rüdiger Schultz. Stochastic programs with first-order dominance constraints induced by mixed-integer linear recourse. *SIAM Journal on Optimization*, 19(2):552–571, 2008. 130

[44] Yongpei Guan, Shabbir Ahmed, Andrew J Miller, and George L Nemhauser. On formulations of the stochastic uncapacitated lot-sizing problem. *Operations Research Letters*, 34(3):241–250, 2006. 18

[45] Yongpei Guan, Shabbir Ahmed, and George L Nemhauser. Cutting planes for multistage stochastic integer programs. *Operations research*, 57(2):287–298, 2009. 18, 29, 36, 37, 38, 72, 82, 86, 94, 95, 96

[46] Yongpei Guan, Shabbir Ahmed, George L Nemhauser, and Andrew J Miller. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming*, 105(1):55–84, 2006. 17, 29, 37, 82

[47] Yongpei Guan and Andrew J Miller. Polynomial-time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research*, 56(5):1172–1183, 2008. 82, 96

[48] V Daniel R Guide. Production planning and control for remanufacturing: industry practice and research needs. *Journal of operations Management*, 18(4):467–483, 2000. 28

[49] V Daniel R Guide, Vaidyanathan Jayaraman, and Rajesh Srivastava. Production planning and control for remanufacturing: a state-of-the-art survey. *Robotics and Computer-Integrated Manufacturing*, 15(3):221–230, 1999. 28

[50] GC Guo and SM Ryan. Progressive hedging lower bounds for time consistent risk-averse multistage stochastic mixed-integer programs. *URL https://works. bepress. com/sarah_m_ryan/93*, 2017. 130, 132

[51] Nir Halman, Diego Klabjan, Mohamed Mostagir, Jim Orlin, and David Simchi-Levi. A fully polynomial-time approximation scheme for single-item stochastic inventory control with discrete demand. *Mathematics of Operations Research*, 34(3):674–685, 2009. 82

[52] Timo Hilger, Florian Sahling, and Horst Tempelmeier. Capacitated dynamic production and remanufacturing planning under demand and return uncertainty. *OR spectrum*, 38(4):849–876, 2016. 29, 48, 50

[53] Martin N Hjelmeland, Jikai Zou, Arild Helseth, and Shabbir Ahmed. Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming. *IEEE Transactions on Sustainable Energy*, 10(1):481–490, 2018. 84, 91

[54] Tito Homem-de Mello and Bernardo K Pagnoncelli. Risk aversion in multistage stochastic programming: A modeling and algorithmic perspective. *European Journal of Operational Research*, 249(1):188–199, 2016. 130, 136, 137, 138, 143

[55] Zhengyang Hu and Guiping Hu. A two-stage stochastic programming model for lot-sizing and scheduling under uncertainty. *International Journal of Production Economics*, 180:198–207, 2016. 82

[56] Mehmet Ali Ilgin and Surendra M Gupta. Environmentally conscious manufacturing and product recovery (ecmpro): a review of the state of the art. *Journal of environmental management*, 91(3):563–591, 2010. 28

[57] Raf Jans and Zeger Degraeve. Modeling industrial lot sizing problems: a review. *International Journal of Production Research*, 46(6):1619–1643, 2008. 14

[58] Vaidyanathan Jayaraman. Production planning for closed-loop supply chains with product recovery and reuse: an analytical approach. *International Journal of Production Research*, 44(5):981–998, 2006. 41, 43

[59] Ruiwei Jiang and Yongpei Guan. Risk-averse two-stage stochastic program with distributional ambiguity. *Operations Research*, 66(5):1390–1405, 2018. 132

[60] Masoumeh Kazemi Zanjani, Mustapha Nourelfath, and Daoud 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):4701–4723, 2010. 28

[61] Onur A Kilic. A mip-based heuristic for the stochastic economic lot sizing problem with remanufacturing. *IFAC Proceedings Volumes*, 46(9):742–747, 2013. 29, 48, 50

[62] Onur A Kilic, Huseyin Tunc, and S Armagan Tarim. Heuristic policies for the stochastic economic lot sizing problem with remanufacturing under service level constraints. *European Journal of Operational Research*, 267(3):1102–1109, 2018. 29, 48, 50, 82

[63] Shigeo Kusuoka. On law invariant coherent risk measures. In *Advances in mathematical economics*, pages 83–95. Springer, 2001. 133

[64] Muris Lage Lage Junior and Moacir Godinho Filho. Production planning and control for remanufacturing: literature review and analysis. *Production Planning & Control*, 23(6):419–435, 2012. 28

[65] Congbo Li, Fei Liu, Huajun Cao, and Qiulian Wang. A stochastic dynamic programming based model for uncertain production planning of re-manufacturing system. *International Journal of Production Research*, 47(13):3657–3668, 2009. 28

[66] Marko Loparic, Yves Pochet, and Laurence A Wolsey. The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming*, 89(3):487–504, 2001. 9, 11, 30, 36, 37, 39, 56, 73

[67] Robert T Lund and Banco Mundial. *Remanufacturing: the experience of the United States and implications for developing countries*, volume 31. World Bank, 1984. 7, 27

[68] Pedro Belluco Macedo, Douglas Alem, Maristela Santos, Muris Lage Junior, and Alfredo 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):1241–1257, 2016. 29, 48, 82, 132

[69] Ali İrfan Mahmutoğulları, Özlem Çavuş, and M Selim Aktürk. Bounds on risk-averse mixed-integer multi-stage stochastic programming problems with mean-cvar. *European Journal of Operational Research*, 266(2):595–608, 2018. 132

[70] Ali İrfan Mahmutoğulları, Özlem Çavuş, and M Selim Aktürk. An exact solution approach for risk-averse mixed-integer multi-stage stochastic programming problems. *Annals of Operations Research*, pages 1–22, 2019. 130

[71] Harry Markowitz. Portfolio selection, 1959. 130

[72] Harry Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. YALE University Press, 1971. 130

[73] Alfredo Moreno, Douglas Alem, Deisemara Ferreira, and Alistair Clark. An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. *European Journal of Operational Research*, 269(3):1050–1071, 2018. 82

[74] Mohd Arshad Naeem, Dean J Dias, Rupak Tibrewal, Pei-Chann Chang, and Manoj Kumar Tiwari. Production planning optimization for manufacturing and remanufacturing system in stochastic environment. *Journal of Intelligent Manufacturing*, pages 1–12, 2013. 29, 48, 50

[75] Mario VF Pereira and Leontina MVG Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, 1991. 18, 83

[76] Georg Ch Pflug and Alois Pichler. *Multistage stochastic optimization*. Springer, 2014. 137

[77] Andy Philpott, Vitor de Matos, and Erlon Finardi. On solving multistage stochastic programs with coherent risk measures. *Operations Research*, 61(4):957–970, 2013. 130, 133, 136

[78] Yves Pochet and Laurence A Wolsey. Polyhedra for lot-sizing with wagner—whitin costs. *Mathematical Programming*, 67(1):297–323, 1994. 54

[79] Yves Pochet and Laurence A Wolsey. *Production planning by mixed integer programming*. Springer Science & Business Media, 2006. 34, 60, 71

[80] Yves Pochet and Laurence A. Wolsey. *Production planning by mixed-integer programming*. Springer, 2006. 101

[81] Franco Quezada, Céline Gicquel, and Safia Kedad-Sidhoum. Stochastic dual dynamic integer programming for a multi-echelon lot-sizing problem with remanufacturing and lost sales. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1254–1259. IEEE, 2019. 158

[82] Franco Quezada, Céline Gicquel, and Safia Kedad-Sidhoum. A stochastic dual dynamic integer programming for the uncapacitated lot-sizing problem with uncertain demand and costs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 353–361, 2019. 83, 84, 91, 158

[83] Franco Quezada, Céline Gicquel, Safia Kedad-Sidhoum, and Dong Quan Vu. A multi-stage stochastic integer programming approach for a multi-echelon lot-sizing problem with returns and lost sales. *Computers & Operations Research*, 116:104865, 2020. 157

[84] Ragheb Rahmaniani, Shabbir Ahmed, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The benders dual decomposition method. *Operations Research*, 68(3):878–895, 2020. 112, 118

[85] Ronald L Rardin and Laurence A Wolsey. Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research*, 71(1):95–109, 1993. 54

[86] Mathijn J Retel Helmrich, Raf Jans, Wilco van den Heuvel, and Albert PM Wagelmans. Economic lot-sizing with remanufacturing: complexity and efficient formulations. *IIE Transactions*, 46(1):67–86, 2014. 54, 66

[87] R Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002. 129, 134

[88] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000. 129, 130, 134

[89] Andrzej Ruszczyński. Risk-averse dynamic programming for markov decision processes. *Mathematical programming*, 125(2):235–261, 2010. 130, 136

[90] Andrzej Ruszczyński and Alexander Shapiro. Conditional risk mappings. *Mathematics of operations research*, 31(3):544–561, 2006. 130, 133, 136

[91] Herbert Scarf. The optimality of (5, 5) policies in the dynamic inventory problem. *Optimal pricing, inflation, and the cost of price adjustment*, 1959. 82

[92] Rüdiger Schultz and Stephan Tiedemann. Conditional value-at-risk in stochastic programs with mixed-integer recourse. *Mathematical programming*, 105(2-3):365–386, 2006. 129, 130

[93] Alexander Shapiro. On a time consistency concept in risk averse multistage stochastic programming. *Operations Research Letters*, 37(3):143–147, 2009. 130, 136, 137, 138, 148, 149

[94] Alexander Shapiro. Minimax and risk averse multistage stochastic programming. *European Journal of Operational Research*, 219(3):719–726, 2012. 137

[95] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2014. 14

[96] Horst Tempelmeier. Stochastic lot sizing problems. In *Handbook of Stochastic Models and Analysis of Manufacturing System Operations*, pages 313–344. Springer, 2013. 14

[97] Saravanan Venkatachalam and Arunachalam Narayanan. Two-stage absolute semi-deviation mean-risk stochastic programming: An application to the supply chain replenishment problem. *Computers & Operations Research*, 106:62–75, 2019. 132

[98] Saravanan Venkatachalam and Lewis Ntaimo. Absolute semi-deviation risk measure for ordering problem with transportation cost in supply chain. *arXiv preprint arXiv:1605.08391*, 2016. 132

[99] Albert Wagelmans, Stan Van Hoesel, and Antoon Kolen. Economic lot sizing: an $\mathcal{O}(n \log n)$ algorithm that runs in linear time in the wagner-whitin case. *Operations Research*, 40(1-supplement-1):S145–S156, 1992. 14, 82

[100] Harvey M Wagner and Thomson M Whitin. Dynamic version of the economic lot size model. *Management science*, 5(1):89–96, 1958. 14, 81

[101] Hsiao-Fan Wang and Yen-Shan 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):2414–2432, 2013. 29, 48, 50

[102] Minjiao Zhang, Simge Küçükyavuz, and Saumya Goel. A branch-and-cut method for dynamic decision making under joint chance constraints. *Management Science*, 60(5):1317–1333, 2014. 29

[103] Minjiao Zhang, Simge Küçükyavuz, and Hande Yaman. A polyhedral study of multiechelon lot sizing with intermediate demands. *Operations Research*, 60(4):918–935, 2012. 54

[104] Chaoyue Zhao and Yongpei Guan. Extended formulations for stochastic lot-sizing problems. *Operations Research Letters*, 42(4):278–283, 2014. 82

[105] Jikai Zou, Shabbir Ahmed, and Xu Andy Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1-2):461–502, 2019. 10, 18, 22, 83, 84, 86, 87, 88, 90, 91, 96, 98, 101, 111, 114, 120, 121, 125, 157

List of Figures

Notations

In this chapter, we summarize the notation used throughout the manuscript. The notations are grouped by subject as follows. We first recall the general notation used throughout the manuscript. We then recall the notations linked to the multi-echelon lot-sizing problem with remanufacturing and lost sales and the notation used for describing the proposed valid inequalities in Part II. Later, we describe the notation related to the decomposition approaches used in Part III. Finally, notation related to the risk-averse approach in Part IV is recalled.

General notation

Basis notation

T	denotes the number of time periods in the planning horizon.
Σ	denotes the number of decision stages in the stochastic process.
$\mathcal{S}$	denotes the set of stages of the stochastic process, i.e., $\mathcal{S} = \{1, \ldots, \Sigma\}$.
$\mathcal{T}$	denotes the set of time periods, i.e, $\mathcal{T} = \{1, \ldots, T\}$.
$\mathcal{T}^\sigma$	denotes the set of time periods belonging to stage $\sigma \in \mathcal{S}$.
ξ_σ	denotes a realization of the stochastic process at stage $\sigma \in \mathcal{S}$.
$\xi_{[\sigma]}$	denotes a sequence of realizations of the stochastic process from stage 1 up to stage $\sigma \in \mathcal{S}$.
$\xi_{[\sigma',\sigma]}$	denotes a sequence of realizations of the stochastic process from stage σ' up to stage $\sigma \in \mathcal{S}$.
$\boldsymbol{\xi}_{[\sigma,\sigma']}$	denotes the sequence of random data vectors corresponding to stages σ through σ'

Scenario tree related notation

$\mathcal{V}$	denotes a set of nodes in a scenario tree.
$\mathcal{V}^t$	denotes the set of nodes belonging to time period $t \in \mathcal{T}$.
$\mathcal{V}^\sigma$	denotes the set of nodes belonging to stage $\sigma \in \mathcal{S}$.
$\mathcal{V}(n)$	denotes the set of nodes belonging to the sub-tree of $\mathcal{V}$ rooted in n.
n	denotes a node in the scenario tree, i.e. $n \in \mathcal{V}$.
ρ^n	denotes the probability associated with the state represented by node n.
ρ^{nm}	denotes the transition probability associated with the transition from the state represented by node n to node m.
t^n	denotes the time period $t \in \mathcal{T}$ of node $n \in \mathcal{V}$.
σ^n	denotes the stage $\sigma \in \mathcal{S}$ of node $n \in \mathcal{V}$.
a^n	denotes the predecessor node of a node n in the scenario tree.
$\mathcal{C}(n)$	denotes the set of immediate children of node $n \in \mathcal{V}$.
$\mathcal{P}(n,m)$	denotes the set of nodes on the path from node n to node m.
c_m^n	denotes a the immediate successor of node n belonging to the set $\mathcal{P}(n,m)$.
$\mathcal{L}(n)$	denotes the set of leaf nodes belonging to $\mathcal{V}(n)$.

Multi-echelon lot-sizing problem related notation

Production system related notation

i denotes an item or part of a returned product.
I denotes the number of part involved in one product
$\mathcal{I}$ denotes the set of all products involved in the remanufacturing production system.
$\mathcal{I}_r$ denotes the set of recoverable parts provided by the disassembly process.
$\mathcal{I}_s$ denotes the set of serviceable parts provided by the refurbishing processes.
$\mathcal{J}$ denotes the set of production processes.
ϖ_i denotes the number of parts i embedded in a returned/remanufactured product.

Stochastic parameters

d^n denotes the customers' demand at node $n \in \mathcal{V}$.
d^{nm} denotes the sum of demand in the path from node n to node m.
f^n denotes the setup cost in a node $n \in \mathcal{V}$.
g^n denotes the unit cost for discarding the unrecoverable parts obtained while disassembling one unit of returned product at node $n \in \mathcal{V}$.
h^n denotes the inventory holding cost in a node $n \in \mathcal{V}$.
eh^n denotes the unit echelon inventory holding cost in a node $n \in \mathcal{V}$.
q^n denotes the unit cost for discarding a recoverable part or a returned product at node $n \in \mathcal{V}$.
r^n denotes the quantity of used products (returns) collected at node $n \in \mathcal{V}$
l^n denotes the unit lost-sales penalty cost at node $n \in \mathcal{V}$
δ_i^n denotes the proportion of recoverable parts $i \in \mathcal{I}_r$ obtained by disassembling one unit of returned product at node $n \in \mathcal{V}$.

Decision Variables

E_i^n denotes a echelon stock decision variable for items $i \in \mathcal{I}$ at node $n \in \mathcal{V}$.
S_i^n denotes a decision variable determining the inventory level at node $n \in \mathcal{V}$.
L^n denotes a decision variable determining the lost sales of remanufactured products at node $n \in \mathcal{V}$.
Q_i^n denotes a decision variable determining the quantity discarded of item $i \in \mathcal{I}_r \cup \{0\}$ at node $n \in \mathcal{V}$.
X_p^n denotes a decision variable determining the quantity to be produced for each process $p \in \mathcal{J}$ at node $n \in \mathcal{V}$.
Y_p^n denotes a binary decision variable that takes the value 1 is a setup cost must be paid at node $n \in \mathcal{V}$ for each process $p \in \mathcal{J}$.

Valid inequalities related notation

d^{nm}	denotes the sum of demand in the path from node n to node m.
$\mathscr{S}$	denotes the subset of nodes in a path inequality, i.e., $\mathscr{S} \subseteq \mathcal{P}(1,n)$ for a node $n \in \mathcal{V}$.
$\bar{\mathscr{S}}$	denotes the subset of node in $\mathcal{P}(1,\ell) \setminus \mathscr{S}$ for a node $n \in \mathcal{V}$.
$\mathcal{U}$	denotes a subset of node from a sub-tree rooted in k, i.e., $\mathcal{U} \subset \mathcal{V}(k)$
$\mathcal{U}_{k,\ell}$	denotes a subset of node from the path between a node c_k^ℓ and ℓ, i.e., $\mathcal{U}_{c_k^\ell,\ell} \subset \mathcal{P}(k,\ell)$
$\mathscr{O}$	denotes a subset of node in a tree inequality, i.e., $\mathscr{O} \subseteq \mathcal{V}$.
$\mathscr{S}_{\mathscr{O}}$	denotes the subset of time periods in a tree inequality, i.e., $\mathscr{S} \subseteq \cup_{n\in\mathscr{O}}\mathcal{P}(1,n)$.
$\mathfrak{s}$	denotes a sequence $\{\mathfrak{s}_1, ..., \mathfrak{s}_{\lvert\mathcal{L}(k)\rvert}\}$ of leaf nodes belonging to $\mathcal{L}(k)$ in increasing order of cumulative demand $\sum_{\mu\in\mathcal{U}_{k,\ell}} d^\mu$: $\sum_{\mu\in\mathcal{U}_{k,\mathfrak{s}_1}} d^\mu \leq ... \leq \sum_{\mu\in\mathcal{U}_{k,\mathfrak{s}_\iota}} d^\mu \leq ... \leq \sum_{\mu\in\mathcal{U}_{k,\mathfrak{s}_{\lvert\mathcal{L}(k)\rvert}}} d^\mu$.
$\Delta_n(\mathscr{O})$	denotes the $\sum_{m_o\in\mathscr{O}\cap\mathcal{V}(n)}(d^{1m_o} - d^{1m_{o-1}})$.
χ	denotes a set of constraints
ϕ	denotes the coefficient associated with the decision variable Y

Decomposition approach related notation

SDDiP algorithm related notation

W	denotes the number of sampled scenarios in the forward step.
Ω_υ	denotes the set of sampled scenarios in the forward step of the SDDiP algorithm at iteration υ.
ω_υ^w	denotes a sampled scenario in Ω_υ at iteration υ
P^n	denotes a sub-problem defined by the constraint and cost parameters at node n
$\mathscr{Q}^n(\cdot)$	denotes the optimal objective value of a problem $\mathscr{P}^n(\cdot)$ for a node $n \in \mathcal{V}$.
$\mathcal{Q}^n(\cdot)$	denotes the expected cost-to-go function at a node $n \in \mathcal{V}$ and it is defined as $\mathcal{Q}^n(\cdot) = \sum_{m \in \mathcal{C}(n)} \rho^{nm} \mathscr{Q}^m(\cdot)$.
$\mathcal{Q}^\sigma(\cdot)$	denotes the expected cost-to-go function at stage $\sigma \in \mathcal{S}$.
$\psi_\upsilon^\sigma(\cdot)$	denotes the approximation of the expected cost-to-go function available at iteration υ for stage σ.
$\underline{\mathscr{Q}}_\upsilon^n(\cdot)$	denotes the lower bound of a problem $\mathscr{P}^n(\cdot)$ for a node $n \in \mathcal{V}$ with an approximate cost-to-function $\psi_\upsilon^\sigma(\cdot)$.
ν_υ^n	denotes a cut coefficient obtained from solving a problem at node n at iteration υ.
π_υ^n	denotes a Lagrangian multiplier from solving a Lagrangian relaxation of problem P_υ^n at iteration υ
B	denotes the number of values used to generate a binary approximation of a continuous decision variable.
$\mathcal{B}$	denotes the set of possible values to generate a binary approximation of a continuous decision variable.

Sub-tree-based decomposition

Γ	denotes the number of macro-stages.
$\mathcal{G}$	denotes a set of macro-stage, which corresponds a partition of the set of decision stages $\mathcal{S}$.
γ	denotes a specific macro-stage in $\mathcal{G}$.
$\mathcal{S}(\gamma)$	denotes a number of consecutive stages denoted for macro-stage $\gamma \in \mathcal{G}$.
$t(\gamma)$	denotes the first time period belonging to macro-stage γ.
$t'(\gamma)$	denotes the last time period belonging to macro-stage γ.
η	denotes the root node of a sub-tree defined by the set of nodes $\mathcal{W}^\eta$.
$\mathcal{W}^\eta$	denotes the set of nodes of the sub-tree $\mathcal{V}(\eta)$ belonging to macro-stage γ, i.e., $\mathcal{W}^\eta = \cup_{t=t(\gamma),\ldots,t'(\gamma)} \mathcal{V}^t \cap \mathcal{V}(\eta)$.
$\mathfrak{L}(\eta)$	denotes the set of leaf nodes of sub-tree $\mathcal{W}^\eta$, i.e, $\mathfrak{L}(\eta) = \mathcal{W}^\eta \cap \mathcal{V}^{t'(\gamma)}$.
$\mho$	denotes the set of sub-tree root nodes induced by $\mathcal{G}$, i.e, $\mho = \cup_{\gamma \in \mathcal{G}} \mathcal{V}^{t(\gamma)}$.

Decision Variables

$U^{n\beta}$	denotes a binary decision variable used to compute the binary approximation of a continuous decision variable S^n, for $\beta \in \mathcal{B}$.
U^n	denotes a vector of binary decision variable used to compute the binary approximation of a continuous decision variable S^n.
$\hat{U}^n$	denotes an auxiliary continuous decision variable used to represent the value of a parent state decision variable U.
$Z^{n,\beta}$	is an auxiliary decision variable representing the value of the state variable at the parent node of n.
θ	denotes a decision variable that takes the minimum value among a set of available linear supporting hyper-planes.

Risk-averse related notation

Ω	denotes the sample space, i.e. the set of all possible outcomes of the random experiment.
$\mathcal{F}$	denotes a sigma-algebra over Ω, i.e. a collection of events or subsets of Ω.
$\mathcal{F}_\sigma \subset \mathcal{F}$	denotes the subset of events observable at stage σ.
$\mathscr{P}$	denotes a probability function assigning to each event in $\mathcal{F}$, i.e., a real number in $[0;1]$ called its probability.
Z	denotes a random variable, i.e. a $\mathcal{F}$-measurable function from Ω to the real numbers $\mathbb{R}$ associating a real value to each possible outcome of the random experiment.
$\mathcal{Z}_\sigma$	denotes the space of $\mathcal{F}_\sigma$-measurable functions from Ω to $\mathbb{R}$.
ϱ	denotes a risk measure, i.e., a mapping from a set of random variables to the real numbers.
$\varrho_{\mathcal{Z}_\sigma \mid \mathcal{Z}_{\sigma-1}}$	denotes a mapping from the space of $\mathcal{F}_\sigma$-measurable functions to the space of $\mathcal{F}_{\sigma-1}$-measurable functions defined with respect to a given realization $Z_{[\sigma-1]} = (Z_1, \ldots, Z_{\sigma-1})$ of the stochastic process up to stage $\sigma - 1$.
$\mathbb{F}$	denotes a multi-stage risk function defined as a mapping from $\mathcal{Z}$ to $\mathbb{R}$ which can be used to measure the risk associated to a stochastic process.
$\mathbb{F}^{\hat{\xi}_{[\sigma]}}$	denotes a conditional multi-period risk function.
F_Z	denotes the cumulative probability distribution of a random variable Z.
VaR_α	denotes the Value-at-Risk at level α and corresponds to the α-percentile of the probability distribution of Z.
x_σ	denotes a decision made in stage σ
$\mathcal{X}_\sigma$	denotes the feasibility set in stage σ, which may depend on the decisions $x_{[\sigma-1]} = [x_1, x_2, ..., x_{\sigma-1}]$ made up to the previous stage as well as on the uncertainty $\xi_{[\sigma]} = [\xi_1, \xi_2, ..., \xi_\sigma]$ observed up to stage σ.
f_σ	denotes a function from $\mathbb{R}^{n_\sigma} \times \mathbb{R}^{m_\sigma}$ to $\mathbb{R}$ that computes the cost of decision x_σ given the observed uncertainty $\xi_\sigma(\omega)$ in that stage.
$(\mathcal{K})$	denotes a generic risk-averse multi-stage stochastic optimization problem.
$(\mathcal{K}_\sigma)$	denotes a generic risk-averse multi-stage stochastic optimization problem for a given stage σ, when all the information relative to the previous stages, i.e. the past decisions $\hat{x}_{[\sigma-1]}$ and the past observations up to stage σ, $\hat{\xi}_{[\sigma]}$, is known.

Decision Variables

φ	denotes a decision variable determining the risk level or value-at-risk for confidence level α.
V^ℓ	denotes a decision variable determining the risk of scenario $\ell \in \mathcal{L}(1)$, i.e. overcost of scenario ℓ with respect to the value-at-risk φ.