A partial nested decomposition approach for remanufacturing planning under uncertainty

Franco Quezada, Céline Gicquel, Safia Kedad-Sidhoum

A partial nested decomposition approach for remanufacturing planning under uncertainty

Franco Quezada[1,2], Céline Gicquel[3], Safia Kedad-Sidhoum[4]

[1] Sorbonne Université, LIP6, 75005 Paris, France
franco.quezada@lip6.fr
[2] Universidad de Santiago de Chile (USACH), LDSPS, Santiago, Chile
franco.quezada@usach.cl
[3] Université Paris Saclay, LRI, 91190 Gif-sur-Yvette, France
celine.gicquel@lri.fr
[4] CNAM, CEDRIC, 75003 Paris, France
safia.kedad_sidhoum@cnam.fr

Abstract. We seek to optimize the production planning of a three-echelon remanufacturing system under uncertain input data. We consider a multi-stage stochastic integer programming approach and use scenario trees to represent the uncertain information structure. We introduce a new dynamic programming formulation that relies on a partial nested decomposition of the scenario tree. We then propose a new extension of the recently published stochastic dual dynamic integer programming algorithm based on this partial decomposition. Our numerical results show that the proposed solution approach is able to provide near-optimal solutions for large-size instances with a reasonable computational effort.

Keywords: Stochastic lot-sizing with remanufacturing · Multistage stochastic integer programming · Stochastic dual dynamic programming.

1 INTRODUCTION

Remanufacturing is defined as a set of processes transforming used products into like-new finished products, mainly by rehabilitating damaged components. By reusing the materials and components embedded in used products, remanufacturing both contributes in reducing pollution emissions and natural resource consumption, making production processes more environment-friendly. However, remanufacturing systems involve several complicating characteristics, among which a high level of uncertainty in the input data needed to make planning decisions. This uncertainty mainly comes from a lack of control on the return flows of used products, both in terms of quantity and quality, and from the difficulty of forecasting the demand for remanufactured products.

The present work investigates production planning for a remanufacturing system involving three production echelons: disassembly of used products into parts, refurbishing of used parts and reassembly into like-new products. We consider uncertainties related to the quantity and quality of used products returned by customers, the demand for remanufactured products, and the production costs.

We propose to handle this problem through a multi-stage stochastic programming approach 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. We assume that the underlying stochastic input process has a finite probability space and we represent the information on the evolution of the uncertain parameters by a discrete scenario tree.

This problem was previously investigated in [2] and [3]. Quezada et al. [3] formulated the problem as a large-size mixed-integer linear program and proposed a customized branch-and-cut algorithm based on new valid inequalities to solve it. Quezada et al. [2] later investigated the use of the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm recently presented in [4] to solve the problem. This algorithm relies on a full decomposition of the problem into a large number of small deterministic sub-problems. Although the above mentioned solution approaches were successful at providing near optimal solutions for small to medium size instances, some numerical difficulties were encountered to solve instances involving large-size scenario trees.

The present work thus discusses a new solution approach for this problem which uses a partial nested decomposition. The main idea consists in partially decomposing the problem into a set of medium-size stochastic sub-problems, each one defined on a sub-tree of the initial scenario tree. A new extension of the SDDiP algorithm exploiting this partial decomposition is then proposed.

The remaining part of this paper is organized as follows. Section 2 describes the problem under study and introduces a mixed-integer linear programming formulation. Section 3 briefly presents the proposed partial nested decomposition approach. Computational results are reported in Section 4. Finally, conclusions and directions for further works are discussed in Section 5.

2 Problem description and modeling

Production system We consider a remanufacturing system comprising three main production echelons: disassembly, refurbishing and reassembly. We seek to plan the production activities in this system over a horizon comprising a discrete set $\mathcal{T} = \{1, .., T\}$ of periods. The system involves a set $\mathcal{I}$ of items. Among these ones, item $i = 0$ represents the used products returned by customers in limited quantity at each period. A used product is composed of I parts. Let α_i be the number of parts i embedded in a used product. The returned products are first disassembled to obtain a set $\mathcal{I}_r = \{1, ..., I\}$ of recoverable parts. Due to the usage state of the used products, some of the parts obtained during disassembly have to be discarded. 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 recoverable parts are then refurbished on dedicated refurbishing processes. The set of $\mathcal{I}_s = \{I + 1, ..., 2I\}$ of serviceable parts obtained after refurbishing are reassembled into remanufactured products which have the same bill-of-material

as the used products. These remanufactured products, indexed by $i = 2I + 1$, are used to satisfy the dynamic demand of customers.

The system comprises a set $\mathcal{P} = \{0, ..., P+1\}$ of production processes. Here, $p = 0$ corresponds to the disassembly process, $p \in \{1, ..., P\}$ corresponds to the process refurbishing the recoverable part indexed by $i = p$ into the serviceable part indexed by $i + I$ and $p = P + 1$ corresponds to the reassembly process. Note that the system comprises one individual refurbishing process for each item embedded in the used product and, hence, we have $P = I$. All these 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, some used products and recoverable parts are allowed to be discarded. This option might be useful in case more used products are returned that what is needed to satisfy the demand for remanufactured products and in case there is a strong unbalance between the part-dependent disassembly yields leading an unnecessary accumulation in inventory of the easy-to-recover parts.

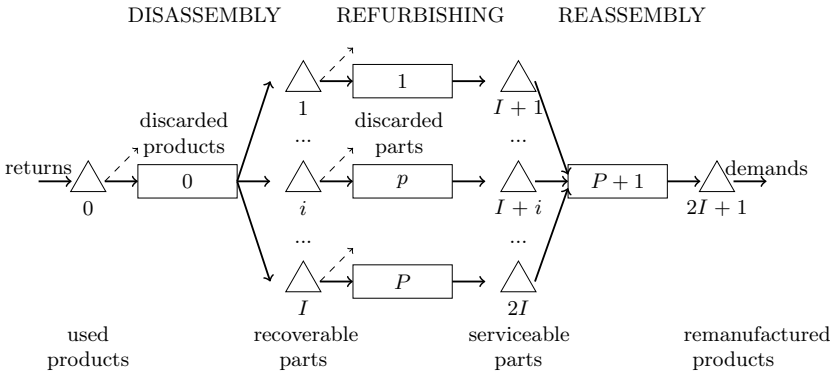

Fig. 1. Illustration of studied remanufacturing system

Uncertainty We focus on the situation in which the input data needed to optimize the production plan for this system are subject to uncertainty and propose to handle this stochastic problem using a multi-stage stochastic integer programming approach. We assume that the evolution of the uncertain parameters can be represented by a discrete scenario tree $\mathcal{V}$ comprising a set $\mathcal{S} = \{1, \ldots, \Sigma\}$ of decision stages. A decision stage σ may correspond to one or several planning periods: let $\mathcal{T}^\sigma$ be the set of time periods belonging to stage σ. Each node $n \in \mathcal{V}$ corresponds to a single period t^n and a single stage σ^n. Let $\mathcal{V}^t$ be the set of nodes belonging to period t. Each node n has a unique predecessor node denoted a^n and represents the state of the system that can be distinguished by the information unfolded up to stage σ^n. At any non-terminal node of the tree, there

are one or several branches to indicate future possible outcomes of the random variables from the current node. Let $\mathcal{C}(n)$ be the set of children of node n. The probability associated with the state represented by the node n is denoted by ρ^n. A scenario is defined as a path in the tree from the root node to a leaf node and represents a possible outcome of the stochastic input parameters over the whole planning horizon.

Each node $n \in \mathcal{V}$ corresponds to a realization of the stochastic input parameters. Let r^n be the quantity of collected used products, d^n be the customers demand and π_i^n be the proportion of recoverable parts $i \in \mathcal{I}_r$ obtained by disassembling one unit of returned product at node $n \in \mathcal{V}$. As for the costs, we have the setup cost f_p^n for process $p \in \mathcal{P}$, the unit inventory cost h_i^n for part $i \in \mathcal{I}$, the unit lost-sales penalty cost l^n, the unit cost q_i^n for discarding item $i \in \mathcal{I}_r \cup \{0\}$ and the unit cost g^n for discarding the unrecoverable parts obtained while disassembling one unit of returned product at node $n \in \mathcal{V}$.

Mixed-integer linear programming formulation In order to build a mathematical model for the problem, we introduce the following decision variables at each node $n \in \mathcal{V}$: x_p^n the quantity of parts processed on process $p \in \mathcal{P}$, $y_p^n \in \{0,1\}$ the setup variable for process $p \in \mathcal{P}$, s_i^n the inventory level of part $i \in \mathcal{I}$, w_i^n the quantity of part $i \in \mathcal{I}_r \cup \{0\}$ discarded and ℓ^n the lost sales of remanufactured products. This leads to the following MILP model.

$$\min \sum_{n \in \mathcal{V}} \rho^n F^n(x^n, y^n, s^n, w^n, l^n) \tag{1}$$

$$x_p^n \leq M_p^n y_p^n \qquad \forall p \in \mathcal{P}, \forall n \in \mathcal{V} \tag{2}$$

$$s_0^n = s_0^{a^n} + r^n - x_0^n - w_0^n \qquad \forall n \in \mathcal{V} \tag{3}$$

$$s_i^n = s_i^{a^n} + \pi_i^n \alpha_i x_0^n - x_i^n - w_i^n \qquad \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} \tag{4}$$

$$s_i^n = s_i^{a^n} + x_{i-P}^n - \alpha_i x_{P+1}^n \qquad \forall i \in I_s, \forall n \in \mathcal{V} \tag{5}$$

$$s_{2I+1}^n = s_{2I+1}^{a^n} + x_{P+1}^n - d^n + \ell^n \qquad \forall n \in \mathcal{V} \tag{6}$$

$$s_i^{a^1} = 0 \qquad \forall i \in \mathcal{I}, \forall n \in \mathcal{V} \tag{7}$$

$$\ell^n, s_i^n \geq 0 \qquad \forall i \in \mathcal{I}, \forall n \in \mathcal{V} \tag{8}$$

$$x_p^n \geq 0, y_p^n \in \{0,1\} \qquad \forall p \in \mathcal{P}, \forall n \in \mathcal{V} \tag{9}$$

The objective function (1) aims at minimizing the expected total cost, over all nodes of the scenario tree. The total cost at node n, $F^n(x^n, y^n, s^n, w^n, l^n) = \sum_{p \in \mathcal{P}} f_p^n y_p^n + \sum_{i \in \mathcal{I}} h_i^n s_i^n + l^n \ell^n + \sum_{i \in \mathcal{I}_r \cup \{0\}} q_i^n w_i^n + g^n x_0^n$, is the sum of the setup, inventory holding, lost sales and disposal costs. Constraints (2) link the production quantity variables to the setup variables. Constraints (3)-(6) are the inventory balance constraints. Without loss of generality, we assume that the initial inventories are all set to 0, i.e., $s_i^{a^1} = 0$ for each $i \in \mathcal{I}$. Finally, Constraints (8)-(9) provide the domain of the decision variables.

In what follows, we denote by X^n the subset of constraints (2)-(9) related to node n.

3 Partial nested decomposition approach

In order to solve large-size instances of Problem (1)-(9), we propose a new solution approach based on a partial nested decomposition of the original stochastic problem into a series of smaller stochastic sub-problems.

Partial decomposition The proposed approach relies on a partial decomposition of the scenario tree $\mathcal{V}$ into a series of smaller sub-trees. This decomposition is obtained by first partitioning 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 by $\mathcal{S}(\gamma)$. We let $t(\gamma)$ (resp. $t'(\gamma)$) represent the first (resp. the last) time period belonging to macro-stage γ. 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 $\mathcal{W}^\eta = \cup_{t=t(\gamma),\ldots,t'(\gamma)} \mathcal{V}^t \cap \mathcal{V}(\eta)$. Here, $\mathcal{V}(\eta)$ denotes 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 by $\mho = \cup_{\gamma \in \mathcal{G}} \mathcal{V}^{t(\gamma)}$ the set of sub-tree root nodes induced by $\mathcal{G}$.

We then define the following sub-problem, denoted $\mathscr{P}^\eta(s^{a^\eta})$, for each sub-tree $\mathcal{W}^\eta, \eta \in \mho$. $\mathscr{P}^\eta(s^{a^\eta})$ focuses on optimizing the production plan for the nodes belonging to sub-tree $\mathcal{W}^\eta$ given the entering stock level of each part $i \in \mathcal{I}$, $s_i^{a^\eta}$, imposed by the parent node a^η.

$$Q^\eta(s^{a^\eta}) = \min \sum_{n \in \mathcal{W}^\eta} \rho^n F^n(x^n, y^n, s^n, w^n, l^n) + \sum_{\ell \in \mathfrak{L}(\eta)} \sum_{m \in \mathcal{C}(\ell)} Q^m(s^\ell) \qquad (10)$$

$$(x^n, y^n, s^n, w^n, l^n) \in X^n \qquad \forall n \in \mathcal{W}^\eta \qquad (11)$$

The objective value $Q^\eta(s^{a^\eta})$ denotes the optimal value of $\mathscr{P}^\eta(s^{a^\eta})$ as a function of the entering stock level s^{a^η}. It comprises two terms. The first term is related to the total expected production cost over all nodes $n \in \mathcal{W}^\eta$. The second term called the expected cost-to-go function represents the expected future costs, i.e. the costs which will have to be paid at the forthcoming decision stages, incurred by the production decisions made within sub-tree $\mathcal{W}^\eta$.

The expected cost-to-go function at leaf node $\ell \in \mathfrak{L}(\eta)$ is defined as the expected value of $Q^m(\cdot)$ over all the children m of leaf node ℓ in the initial scenario tree $\mathcal{V}$, i.e. over all $m \in \mathcal{C}(\ell)$. This 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 $\mathcal{Q}^\ell(s^\ell)$. Note that for all nodes belonging to the last period of the planning horizon, i.e. for all $\ell \in \mathcal{V}^T$, $\mathcal{Q}^\ell(\cdot) \equiv 0$.

Extended Stochastic Dual integer Programming algorithm The reformulation of Problem (1)-(9) using the dynamic programming recursion described by Equations (10)-(11) enables to develop a solution approach based on the Stochastic Dual integer Programming (SDDiP) algorithm recently presented by [4]. Basically, this algorithm will solve Problem (1)-(9) by solving a sequence

of sub-problems (10)-(11) in which each expected cost-to-go function $\mathcal{Q}^{\ell}(\cdot)$ is iteratively approximated by a piece-wise linear function.

Note that a key assumption for developing such algorithm is that the scenario tree satisfies 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 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, in this case, the stochastic process can be represented at macro-stage γ by a set $\mathcal{R}^{\gamma} = \{1, \ldots, R^{\gamma}\}$ 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 $t(\gamma), \ldots, t'(\gamma)$. Let $\mathfrak{L}(\gamma, \zeta)$ 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 $\mathscr{P}^{\gamma}$ per macro-stage and each sub-problem $\mathscr{P}^{\eta}, \eta \in \mho$, will be described as $\mathscr{P}^{\gamma^{\eta}}(s^{a^{\eta}}, \mathcal{X}^{\gamma^{\eta},\zeta})$ where $\mathcal{X}^{\gamma^{\eta},\zeta}$ is the realization corresponding to $\mathcal{W}^{\eta}$.

Each iteration υ of the extended SDDiP algorithm comprises a sampling step, a forward step and a backward step. In the sampling step, a subset of W scenarios is sampled from the scenario tree. 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.

In the forward step, the algorithm proceeds stage-wise from macro-stage $\gamma = 1$ to Γ by solving, for each sampled scenario ω^w and each macro-stage γ, the problem $\hat{\mathscr{P}}^{\gamma}(s^m, \mathcal{X}^{\gamma,\zeta^{w,\gamma}})$, which uses 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. At the end of this step, a statistical upper-bound of the problem is computed as the weighted average over all sampled scenarios.

In the backward step, the algorithm proceeds stage-wise from macro-stage $\gamma = G$ to macro-stage 1. Thus, for each scenario $w = 1, \ldots, W$, each node $m \in \omega_{\upsilon}^w \cap \mathcal{V}^{t'(\gamma)}$ and each realization $\zeta \in \mathcal{R}^{\gamma+1}$, it solves a suitable relaxation of Problem $\hat{\mathscr{P}}^{\gamma+1}(s^m, \mathcal{X}^{\gamma+1,\zeta})$. This relaxation is then used to improve the representation of the approximate cost-to-go function $\mathcal{Q}^{\gamma}(\cdot)$ through the generation of a new cut. Finally, the sub-problem solved at macro-stage $\gamma = 1$ provides a lower bound for the overall problem. The algorithm stops when the upper and lower bounds are close enough, according to a convergence criteria.

As a synthesis, the main steps of the proposed extended SDDiP algorithm are summarized in Algorithm 1.

Original vs extended SDDiP algorithm The reader is referred to [4] for a detailed description of the original version of SDDiP algorithm. The proposed extended version differs from the original one with respect to two key features.

Algorithm 1: Extended 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 $\hat{\mathscr{P}}^\gamma(s^m, \mathcal{X}^{\gamma,\zeta})$ for $m = \omega_\upsilon^w \cap \mathcal{V}^{t'(\gamma-1)}$
      Record S_υ^ℓ for $\ell = \omega_\upsilon^w \cap \mathfrak{L}(\gamma, \zeta_\upsilon^{w,\gamma})$
    end
    $\upsilon^w \leftarrow \sum_{n \in \omega_\upsilon^w} F^n(x^n, y^n, s^n, w^n, l^n)$
  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 $\hat{\mathscr{P}}^\gamma(s^m, \mathcal{X}^{\gamma,\zeta})$ and collect the coefficients of the strengthened Benders' cut
        Solve the Lagrangian relaxation of $\hat{\mathscr{P}}^\gamma(s^m, \mathcal{X}^{\gamma,\zeta})$ and collect the constant value of the strengthened Benders' cut
      end
    end
    Add the generated cut to the current approximation of $\mathcal{Q}^{\gamma+1}$
  end
  $LB \leftarrow \hat{Q}_{\upsilon+1}^1(0)$
  $\upsilon \leftarrow \upsilon + 1$
end

First, the original version of the SDDiP algorithm uses a full decomposition of the stochastic problem into small deterministic sub-problems, i.e. a decomposition in which each macro-stage γ corresponds to a single decision stage σ, whereas we propose to use a partial decomposition. Using a partial nested decomposition instead of a full one may positively impact the computational efficiency of the SDDiP algorithm. Namely, it reduces the number of expected cost-to-go functions for which a piece-wise linear approximation must be built. Furthermore, each solved sub-problem covers a larger portion of the planning horizon so that the solution obtained at a given iteration of the algorithm will tend to be less myopic and thus of better quality. All this may contribute in accelerating the global convergence of the algorithm. Yet, the sub-problems to be solved at each iteration will be MILPs of larger size. The computational effort needed to

solve them will thus also be larger. It is however possible to decrease it to some extent by using polyhedral approaches such as the one presented in [3].

Second, Zou et al. [4] showed that the finite convergence of the SDDiP algorithm is guaranteed when the state variables, i.e. the variables linking the decision stages to one another, are restricted to be binary. In Problem (1)-(9), the state variables are the continuous inventory level variables s^n. In this case, Zou et al. [4] suggest to use a binary approximation of the continous state variables, which requires the introduction of a large number of additional binary variables in the problem formulation. In the proposed extended version of this algorithm, as done e.g. by Hjelmeland et al. [1] and Quezada et al. [2], we keep continuous state variables. 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.

4 Computational results

In this section, we focus on assessing the performance of the proposed extended SDDiP algorithm by comparing it with the one of a stand-alone mathematical programming solver using formulation (1)-(9) and the one of the original SDDiP algorithm proposed by Zou et al. [4].

We randomly generated instances following the same procedure as the one used in [3]. We used various scenario tree structures and various values for the return over demand 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 instances involving between 1000 and 3.2 millions scenarios.

The partial decomposition of the problem was obtained by using 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. As for the stopping criteria, the algorithm stops when the lower bound does not improve after 30 iterations, or when 1000 iterations have been carried out. All the algorithms were implemented in C++ using the Concert Technology environment. The MILP and LP sub-problems embedded into the SDDiP algorithm 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 imposed a time limit of 1800 seconds.

Table 1 displays the numerical results. Each line corresponds to a given combination of Σ, R, b, and r/d and provides the average results for the related 20 instances. For each combination, Table 1 displays the gap between the lower bound and the upper bound ($|UB - LB|/UB$) found before some stopping criterion is reached and the average computation time. The label "$*$" represents the

case when CPLEX in default mode could not report any gap within the imposed time limit.

Results from Table 1 show that the proposed extended SDDiP algorithm provides solutions of a significantly improved quality within the alloted computation time. Namely, the average gap over all tested instances is decreased from 54.50% (resp. 37.21%) when directly solving Problem (1)-(9) with CPLEX solver (resp. when solving it with the original SDDiP algorithm) to 4.98% when using the proposed extended version with $G = 2$ stages per macro-stage.

We note that a large part of this improvement can be explained by the use of continuous (rather than binary) state variables. This can be seen by comparing the average gap obtained with the original SDDiP algorithm, 37.21%, with the one obtained when using the proposed extension with $G = 1$, i.e. with a full decomposition of the problem, 6.85%. The use of a partial decomposition based on $G = 2$ stages per macro-stage then further improves the solution quality by decreasing the average gap from 6.85% to 4.98%.

Table 1. Numerical results

Instance				CPLEX		SDDiP		Extended SDDiP $G = 1$		Extended SDDiP $G = 2$	
r/d	Σ	b	R	Gap	Time	Gap	Time	Gap	Time	Gap	Time
1	4	1	10	0.25	1,800.56	11.63	1,418.64	3.45	754.98	0.97	599.16
			20	6.57	1,801.32	11.75	1,367.78	4.16	1,041.48	3.33	1,009.13
	6	1	10	76.89	1,818.76	13.97	1,803.22	3.76	1,246.89	3.02	1,498.62
			20	*	*	16.26	1,804.41	4.22	1,300.51	3.78	1,826.17
	8	2	5	92.02	1,837.44	24.70	1,814.66	2.77	1,547.97	2.23	1,791.41
		5	5	*	*	17.58	1,840.89	2.37	1,683.59	2.70	1,832.97
	12	1	3	86.29	1,865.56	27.24	1,809.93	3.09	1,546.61	2.46	1,723.32
		3	3	*	*	34.49	1,829.69	2.49	1,785.61	1.87	1,726.96
Average				52.40	1,824.73	19.70	1,711.15	3.29	1,363.46	2.54	1,500.97
3	4	1	10	0.31	1,000.40	33.81	1,274.02	7.83	762.06	1.54	826.66
			20	7.96	1,801.40	28.91	1,276.01	6.64	1,039.12	3.84	942.09
	6	1	10	81.22	1818.23	44.30	1,804.47	8.87	1,202.04	5.94	1,135.28
			20	*	*	53.67	1,796.29	14.27	1,206.11	11.84	1,269.60
	8	2	5	96.13	1,837.22	68.13	1,818.03	10.17	1,525.91	6.71	1,682.07
		5	5	*	*	65.15	1,830.02	11.79	1,800.69	12.60	1,824.09
	12	1	3	95.09	1,869.61	70.48	1,805.16	11.96	1,650.80	8.19	1,683.99
		3	3	*	*	73.23	1,848.63	13.32	1,801.59	8.68	1805.90
Average				56.63	1,825.60	54.94	1,683.37	10.61	1,373.54	7.42	1,393.46

5 Conclusion and perspectives

We studied production planning for a remanufacturing system under uncertain input data and investigated a multi-stage stochastic integer programming approach. We proposed to use a new extension of the SDDiP algorithm recently

introduced by Zou et al. [4] to solve the problem. Computational experiments carried out on large-size randomly generated instances suggested that the proposed extended algorithm significantly outperforms both the original SDDiP algorithm and the mathematical programming solver CPLEX using an extensive MILP formulation.

Note that, in the proposed extended SDDiP algorithm, we generate strengthened Benders' cuts to under approximate the expected cost-to-go functions. These cuts are linear inequalities for which part of the coefficients are obtained by solving the linear relaxation of the corresponding sub-problems and by recording the dual values of the constraints linking the sub-problems to one another. Nonetheless, by noticing that these dual values vary according to the linear relaxation formulation used for each sub-problem, it is possible to generate different strengthened Benders' cuts by using different formulations. Thus, an interesting research direction will be to exploit the current knowledge about the polyhedral structure of the sub-problem to iteratively strengthen the linear relaxation formulation of these sub-problems. For example, valid inequalities introduced in [3] can be used to strengthen the linear relaxation of each sub-problem in order to generate additional strengthened Benders' cuts, which might positively impact the performance of the algorithm.

Finally, note that we assumed in our problem modeling uncapacitated production processes. Extending the present work in order to account for production resources with limited capacity could thus be an interesting direction for further research.

References

1. Hjelmeland, M.N., Zou, J., Helseth, A., Ahmed, S.: Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming. IEEE Transactions on Sustainable Energy (2018)
2. Quezada, F., Gicquel, C., Kedad-Sidhoum, S.: 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). pp. 1254–1259. IEEE (2019)
3. Quezada, F., Gicquel, C., Kedad-Sidhoum, S., Vu, D.Q.: 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)
4. Zou, J., Ahmed, S., Sun, X.A.: Stochastic dual dynamic integer programming. Mathematical Programming **175**(1), 461–502 (2019)