Stochastic Dual Dynamic Programming for Multiechelon Lot Sizing with Component Substitution

Simon Thevenin, Yossiri Adulyasak, Jean-François Cordeau

Stochastic Dual Dynamic Programming for Multiechelon Lot Sizing with Component Substitution

Simon Thevenin
IMT Atlantique, LS2N-CNRS, 44307 Nantes, France; simon.thevenin@imt-atlantique.fr

Yossiri Adulyasak and Jean-François Cordeau
GERAD and Department of Logistics and Operations Management, HEC Montréal, Montréal, QC, H3T 2A7, Canada;
{yossiri.adulyasak, jean-francois.cordeau}@hec.ca

This work investigates lot sizing with component substitution under demand uncertainty. The integration of component substitution with lot sizing in an uncertain demand context is important because the consolidation of the demand for components naturally allows risk pooling and reduces operating costs. The considered problem is relevant not only in a production context but also in the context of distribution planning. We propose a stochastic programming formulation for the static-dynamic type of uncertainty, where the setup decisions are frozen, but the production and consumption quantities are decided dynamically. To tackle the scalability issues commonly encountered in multistage stochastic optimization, this paper investigates the use of stochastic dual dynamic programming (SDDP). In addition, we consider various improvements of SDDP, including the use of strong cuts, the fast generation of cuts by solving the linear relaxation of the problem, and retaining the average demand scenarios. Finally, we propose two heuristics, namely, a hybrid of progressive hedging with SDDP and a heuristic version of SDDP. Computational experiments conducted on well-known instances from the literature show that the heuristic version of SDDP outperforms other methods. The proposed method can plan with up to 10 decision stages and 20 scenarios per stage, which results in 20^{10} scenario paths in total. Moreover, as the heuristic version of SDDP can replan to account for new information in less than a second, it is convenient in a dynamic context.

Key words: Lot sizing, stochastic dual dynamic programming, stochastic optimization, stochastic demand
History:

1. Introduction

The manufacturing industry is undergoing a paradigm shift from mass production to mass customization. As a result, companies tend to increase their assortment and launch new products more frequently. In this context, the product demand can be highly volatile and unpredictable, and thus the manufacturers must operate in an increasingly uncertain and complex environment (Sreedevi and Saranga 2017). To cope with this uncertainty, companies tend to operate with more flexibility (Chan et al. 2017). Among other agility means,

various companies rely on component substitution to mitigate supply chain risk (Freeman et al. 2020). Indeed, unlike what is commonly assumed in production planning, the bills of materials of various manufacturing products found in practice are usually not fixed. In fact, component substitution is common in various industries such as computers (Begnaud et al. 2009) and semiconductors (Rao et al. 2004). Lot sizing with substitution also occurs in distribution planning, when a retailer can be supplied by different distribution centers with different transportation costs (Firoozi et al. 2020). To plan the production in this situation, the available planning tools must properly account for the stochastic demand and component substitution (Cecere 2015). This paper investigates the multiechelon lot-sizing problem with component substitution under uncertain demand.

The integration of component selection with lot sizing in an uncertain demand context is important because the consolidation (centralization) of the demand for components reduces the fixed setup costs and allows risk pooling. In such circumstances, the decision maker can benefit from the flexibility through an adaptive decision process. That is, the manufacturer decides the amount of end-item to produce and the consumed components dynamically, after observing the demand in the previous period. Consequently, we consider the case of stochastic demand with static-dynamic types of uncertainty.

In the static-dynamic type of uncertainty (Bookbinder and Tan 1988), the setups are decided in the initial period (period 0) for the entire horizon, whereas the lot sizes are chosen at each period after having observed the demand. This decision process leads to a multistage stochastic optimization problem, and such problems are commonly modeled with a scenario tree. However, scenario trees have two major drawbacks: (1) the size of the tree grows exponentially with the number of decision stages (leading to N^T scenario paths, where N is the number of scenarios in each stage, and T is the number of decision stages), and this is impractical for a long planning horizon; (2) the solution computed based on a sampled scenario tree gives only the first stage decisions, and the problem must be solved again from scratch when new information (not in the considered sample) is available.

To overcome these difficulties, the present paper explores the use of the stochastic dual dynamic programming (SDDP) algorithm framework. This framework was proposed by Pereira and Pinto (1991) to solve multistage stochastic optimization problems. SDDP decomposes the problem per decision stage, and it iterates successive forward and backward passes. In the forward pass, the decisions in each stage t are computed with the current

policy based on the state of the system, on the cost of stage t, and on the approximated cost-to-go (from stage $t+1$ to the end of the horizon). The backward pass improves the cost-to-go approximations with information on the dual subproblems. Note that SDDP does not suffer from the drawbacks of the scenario tree approach for the following reasons. First, the stagewise decomposition in SDDP prevents the exponential growth of the number of scenario paths (as a result, SDDP considers only N scenarios per stage), provided that the probability distributions of the different decision stages are independent. Second, the cost-to-go approximations remain valid for any realization of the stochastic parameters. Therefore, when new information unfolds, SDDP updates the quantities in a fraction of a second because it only requires solving the subproblem of the corresponding stage.

The contribution of this work lies in the study of SDDP for the multiechelon lot-sizing problem with component substitution under static-dynamic uncertain demand. To the best of our knowledge, we are the first to develop a highly scalable framework for this challenging problem. Our results show that a straightforward adaptation of SDDP to the considered problem is not efficient because computing accurate approximations of the future costs in each stage requires significant computing effort, and it is a very time-consuming process. Therefore, to develop an efficient algorithm, we first employ various enhancements, including those from the literature on Benders decomposition such as the use of strong cuts (Magnanti and Wong 1981) with the implementation of Papadakos (2008) and the multicut version of SDDP. While these improvements are often used in Benders decomposition, only a few papers investigate their effectiveness and applicability in SDDP. In addition to these algorithmic improvements, we propose a lower bound lifting method based on the average demand scenario, and scenario sampling with Randomized Quasi-Monte Carlo (RQMC) (Cranley and Patterson 1976). To tackle instances of practical size, we propose two SDDP heuristics. First, we develop a hybrid approach of progressive hedging (PH) and SDDP, where the setup values are computed with PH and fixed before running SDDP. In addition, we propose an efficient heuristic version of SDDP. The heuristic SDDP starts with initial setup values computed with a two-stage approximation of the problem, and it iteratively improves this solution. At each iteration, the first-stage problem is solved to redetermine the setup values prior to running SDDP on the subsequent stage until convergence. To investigate the performance of these proposed methods, extensive computational

experiments were conducted using the well-known instances from Tempelmeier and Derstroff (1996). The results show that the heuristic SDDP outperforms the hybrid approach of PH and SDDP as well as the multistage model originally proposed in Thevenin et al. (2021).

The rest of this paper is organized as follows. Section 2 reviews the literature on component substitution and stochastic optimization in lot sizing. Section 3 describes the multiechelon capacitated lot-sizing problem under static-dynamic uncertain demands. Section 4 presents the SDDP algorithms, and the methods to compute the setups are presented in Section 5. Finally, Section 6 reports the experimental results, followed by a conclusion in Section 7.

2. Literature Review

Since the seminal paper of Harris (1990) (reprint of the 1913 article), the lot-sizing problem has received considerable attention from the operations research community. Readers interested in the latest development on this topic are referred to recent surveys (e.g., Karimi et al. 2003, Jans and Degraeve 2008, Buschkühland et al. 2010, Brahimi et al. 2017). The rest of this section reviews the specific characteristic considered in this work, namely, component substitution and stochastic demand.

2.1. Literature Review on Lot Sizing with Component Substitution

In various manufacturing contexts, producers have the flexibility to substitute a given component with a component of better quality or with more functionalities. Despite the higher cost of these alternative components, the substitution is relevant because it reduces the ordering costs. Similarly, in a transportation context, a local distribution center can be substituted by a farther one at the expense of additional transportation costs. Lot sizing with component substitution was introduced by Balakrishnan and Geunes (2000). The authors propose a dynamic programming approach, and they show that component substitution can lead to significant cost savings, especially in a dynamic demand context.

Component selection is usually considered in a single-echelon bill of material (BOM). Geunes (2003) proposes a facility location reformulation to solve the problem with the dual ascent procedure proposed by Erlenkotter (1978) as a solution method. Yaman (2009) considers the specific case with only two items and one-way substitution. The author provides a polyhedral analysis and gives some valid inequalities. Furthermore, Lang and

Domschke (2010) introduce a case where multiple components are required to substitute a component. The authors propose a facility location reformulation, as well as some valid inequalities. A different stream of research (e.g., Wu et al. 2017) considers the joint lot-sizing and cutting stock problem, where components are cut from sheets of different sizes.

For the multiechelon case, Begnaud et al. (2009) consider the case of a flexible BOM, where each item can be produced with a different recipe. Each recipe corresponds to a different task in the BOM, and the model selects one task among all possible ones. A variant of the problem is called multiechelon lot sizing with BOM substitution (Wei et al. 2019). In this variant, the entire BOM can be modified in each period, and all items use the same BOM in a period (but different BOMs can be used in different periods).

To the best of our knowledge, lot sizing with component substitution has never been considered in a stochastic demand situation. However, there exists a vast literature (e.g., Gallego et al. 2006, Han et al. 2014) on inventory systems with product substitution (Shin et al. 2015). This stream of research differs from the current study because the substitution concerns the end-items and not the components. Product substitution can be customer-driven if the customer selects a different product when the one the customer wants is not available, or supplier-driven if the supplier decides to downgrade a product to meet the demand (Huang et al. 2011). There exist several studies considering stochastic optimization for inventory systems with product substitution. For instance, Hsu and Bassok (1999) propose a scenario-based stochastic optimization approach for a production system under random yield and random demand with product substitution. To speed up the solution process, the authors rely on Benders decomposition, and they show that a greedy algorithm can solve the subproblems as they have a special network flow structure. Also, Rao et al. (2004) propose a stochastic optimization method and heuristics for the single-period inventory system with setup in a static-dynamic framework. The authors show that substitution results in substantial savings, especially when the setup costs or the demand variance is large.

2.2. Literature Review on Lot Sizing Under Stochastic Demand

Compared to the vast literature on deterministic lot sizing, only a limited number of works consider the stochastic version of this problem. There exist various sources of uncertainty in lot sizing, such as demand, yield, lead times, or capacity, and the present work focuses on demand uncertainty. As meeting the demand in any situation is expensive, stochastic

optimization models seek to minimize the expected backlog (e.g., Tarim and Kingsman 2006, Tunc et al. 2018) or to reach a given service level (e.g., Bookbinder and Tan 1988) with chance constraint approaches. In addition, three decision frameworks exist, namely, static-static, static-dynamic, and dynamic-dynamic. In static-static, the lot sizes are determined in period 0 for the entire horizon, and they are frozen. On the contrary, in dynamic-dynamic, the decisions are made in each period after having observed the realization of the stochastic parameters. In static-dynamic, the setups are static (selected in the first period for the entire horizon and frozen), whereas the quantities are computed in each period after the unknown parameters unfold. For more information on stochastic lot sizing, the interested reader is referred to the recent surveys of Tempelmeier (2013) and of Aloulou et al. (2014).

A few studies consider the capacitated lot-sizing problem with stochastic demands (e.g., Tempelmeier and Hilger 2015, Brandimarte 2006). However, most studies on stochastic lot sizing consider a single-echelon production system. To the best of our knowledge, the only papers on stochastic optimization for multiechelon systems are Grubbström and Wang (2003), Quezada et al. (2020), and Thevenin et al. (2021). Grubbström and Wang (2003) propose a dynamic programming approach to minimize the net present value in the capacitated multiechelon lot-sizing problem with stochastic demand, but they ignore lead times. Quezada et al. (2020) propose a branch-and-cut approach for the uncapacitated lot-sizing problem in a re-manufacturing system with three echelons under various types of uncertainty (demand, cost, supply, yield). Thevenin et al. (2021) propose a scenario tree approach for the multiechelon capacitated lot-sizing problem under stochastic demand. Thevenin et al. (2021) show that stochastic optimization significantly reduces the costs in material requirements planning when compared to traditional approaches used in lot-size and safety stock determination. In the present work, we propose an SDDP approach to solve requirements planning with component substitution over a long planning horizon. To alleviate the scalability issues, we develop a heuristic version of SDDP, and this method can handle a much larger set of scenario paths than the mixed-integer linear program (MILP) proposed in Thevenin et al. (2021).

Optimization methods commonly used for stochastic lot sizing include PH (Haugen et al. 2001), scenario-based mathematical programming (Brandimarte 2006), dynamic programming (Sox 1997), heuristics (Piperagkas et al. 2012), branch-and-bound (Tarim et al.

2011), among others. To the best of our knowledge, the only papers on the application of the SDDP algorithm to a lot-sizing problem are (Quezada et al. 2019) and (Quezada et al. 2021). Nevertheless, in that work, the authors consider the uncapacitated single-item lot-sizing problem. Thus, the technique applied in that work cannot be applied in a straightforward way to the capacitated multiechelon multi-item lot-sizing problem with component substitution studied in the present work. The SDDP algorithm has been mostly applied to hydrothermal energy operations planning (e.g., Soares et al. 2017a, Lohmann et al. 2016), and other applications include the control of microgrids (Pacaud et al. 2018), pastoral dairy farms (Dowson et al. 2019), or portfolio optimization (Valladão et al. 2019). The multiechelon lot-sizing problem under demand uncertainty requires a large number of scenarios to approximate the costs (Thevenin et al. 2021). Consequently, the classical scenario tree approach leads to a huge MILP that consumes too much memory. In this context, decomposition approaches such as PH (decomposition per scenario) and SDDP (decomposition per stage) are suitable and could be highly efficient in addressing the tractability issue in this problem. While existing studies (e.g., Brandimarte 2006, Quezada et al. 2020) on scenario tree methods for lot sizing are limited to less than 1000 scenario paths, the SDDP algorithm proposed in the present work computes the production quantities based on 20^{10} scenario paths.

Our contributions are threefold. (1) This paper is the first to study lot sizing with component substitution under stochastic demand. Our analysis shows that component substitution can pool the risk, and it allows maintaining the same service level with less inventory. (2) We investigate the performance of SDDP for the lot-sizing problem, and we study the impact of multiple algorithmic improvements of the SDDP. (3) We propose two strategies to compute and fix the setups in the SDDP framework, namely, a hybrid of PH and SDDP and a heuristic version of SDDP. Our results show that the heuristic is competitive with the MILP on small scenario trees, but it can handle much larger scenario sets.

3. Considered Problem

This paper addresses the capacitated multiechelon lot-sizing problem with component substitution (CMLCS) under uncertain demand. Given a multiechelon BOM, the demand distribution of the end-items, and the unit capacity consumption of each item, the CMLCS

is to decide when to produce, as well as the sizes of the production lots, and to select the components to minimize the expected setup, production, substitution, inventory, and backlog costs.

The BOM gives the production recipe of each item in the set $\mathcal{I}_e$ of end-items. More precisely, the production of item j consumes R_{ij} units of component i. However, an item i' in the set $\mathcal{A}_i$ can substitute component i, and this substitution costs $a_{i'i}$. For simplicity, we assume $i \in \mathcal{A}_i$ with $a_{ii} = 0$. We denote by $\mathcal{I}_c$ the set of components and by $\mathcal{I}$ the set of all items with $\mathcal{I} = \mathcal{I}_e \cup \mathcal{I}_c$. Without loss of generality, we assume that there is no demand for components, whereas the demand's probability distribution of each end-item is given for each period in the planning horizon $\mathcal{H} = \{1, \ldots, T\}$. The uncertain demands for each period are represented by a set of discrete scenarios. We denote by Ω_t the set of demand realizations at a given stage t, where each scenario $\omega_t \in \Omega_t$ corresponds to a realization of the demand in period t. We also let Φ denote the set of *scenario paths* $\phi \in \Phi$, where each scenario path represents a vector of demand realizations over the planning horizon $\mathcal{H}$, i.e., $\phi = (\omega_1, \omega_2, ..., \omega_T)$. In addition, let $\phi_{[t]}$ denote a partial scenario path with demand realizations from period 1 to t and let $\gamma_t(\phi)$ denote the scenario ω_t in period t associated with the scenario path ϕ. Each scenario path ϕ has a probability σ_ϕ, and we also define p_{ω_t} as the conditional probability associated with scenario node ω_t, i.e., $p_{\omega_t} = \mathcal{P}rob\{\omega_t = \gamma_t(\phi) | \gamma_{t-1}(\phi)\}$. The requirement plan must account for the production capacity and lead times. Each resource r in the set of resources $\mathcal{R}$ has a given capacity C_r, and the production of one unit of item i consumes K_{ir} units of capacity of resource r. The items i produced in period t are available in period $t + L_i$. The problem accounts for the backlog cost b_i, holding cost h_i, production cost v_i, the end-of-horizon backlog cost e_i, the substitution costs $a_{i'i}$, and the setup cost s_i for each item i. The objective is to find a production plan that minimizes the expected total cost.

In the classical lot sizing with backlog formulation, a set of demand satisfaction constraints is imposed to ensure that all the demands are met within the planning horizon. In a stochastic demand context, such constraints would be too conservative as they require producing enough items to meet the demands in the worst-case scenario paths. Therefore, to avoid over-conservative solutions, the model allows a positive backlog level in the last period T of the horizon (Absi et al. 2011, Thevenin et al. 2021). In practice, this backlog may be met several periods after the end of the horizon, but the model cannot account

for these costs. Therefore, the end-of-horizon backlog cost e_i is set larger than the backlog cost b_i within the horizon.

In the static-dynamic uncertainty, the setups for each item are decided in period 0 for all the periods in the horizon, and they remain frozen. On the contrary, the inventory and backlog levels are computed after observing the demands in each scenario, and the quantities for each item ordered in period t are computed after having observed the demand in period $t-1$ in each scenario. Similarly, the quantity of components consumed to meet the internal demand of item j in period t is decided after observing the demand in period $t-1$ for each scenario. For the sake of clarity, we present below the subproblem of each decision stage before introducing the scenario tree formulation.

3.1. SDDP Formulation of CMLCS

Each decision stage corresponds to the arrival of new information on the end-items' demands. More precisely, the first stage corresponds to the initial state, where no information on the demand is available, and the decisions on the quantities to produce in period 1 must be made. Similarly, the stage $t \in \{2, \dots, T\}$ corresponds to the beginning of period t, where the demands of end-items in period $t-1$ are known, and the decisions on the quantities to produce in period t can be determined sequentially at each stage. Finally, the last stage computes the inventory and backlog costs when the last period demands are known. This section successively presents the subproblem in stage 1, the subproblem in stage $t \in \{2, \dots, T\}$, and the subproblem in the last stage $T+1$. For ease of exposition, we assume strictly positive lead times for all items, but the proposed algorithm can easily be extended to the case of zero lead times. For clarity, we denote by $Q_{\mathcal{I}\mathcal{H}}$ (resp. $Q_{\mathcal{I}[t]}$) the matrix of quantity variables Q_{it} for all $i \in \mathcal{I}$ and $t \in \mathcal{H}$ (resp. period $\tau \in \{1, \dots, t\}$), and the same notation applies to other variables. We provide below the optimization problem for each decision stage.

First-stage model. The first-stage decisions are made when the demand in periods 1 to t is unknown. These decisions include the setups Y_{it} for all items i and periods t, the production quantities Q_{i1} for all items i in period 1, the consumed quantities W_{ij1} for all arcs (i, j) in the BOM in period 1, and the inventory levels I_{i1} of components in period 1. Contrarily to the components, the end-item inventory levels in period 1 depend on the realization of the demands in period 1, and they are thus computed in stage 2. Without loss of generality, we assume that the initial inventory levels (I_{i0}) and initial backlog levels

(B_{i0}) are equal to zero for all items i (these initial values can also be set to the actual values in practice). The first-stage problem can be formulated as follows:

$$\text{Min} \quad \sum_{i\in\mathcal{I}}\left(\sum_{t\in\mathcal{H}} s_i Y_{it} + v_i Q_{i1} + \sum_{j\in\mathcal{I}} a_{ij} W_{ij1}\right) + \sum_{i\in\mathcal{I}_c} h_i I_{i1} \tag{1}$$
$$+ \mathcal{F}_2(Y_{\mathcal{I}\mathcal{H}}, Q_{\mathcal{I}1}, I_{\mathcal{I}_e 0}, I_{\mathcal{I}_c 1}, B_{\mathcal{I}_e 0})$$

$$\text{s.t.} \quad I_{i0} - \sum_{j\in\mathcal{I}} W_{ij1} - I_{i1} = 0 \qquad i\in\mathcal{I}_c \tag{2}$$

$$\sum_{i\in\mathcal{A}_k} W_{ik1} = \sum_{j\in\mathcal{I}} R_{kj} Q_{j1} \qquad k\in\mathcal{I}_c, \quad j\in\mathcal{I} \tag{3}$$

$$Q_{i1} \leq Y_{i1} M_i \qquad i\in\mathcal{I} \tag{4}$$

$$\sum_{i\in\mathcal{I}} Q_{i1} K_{ir} \leq C_r \qquad r\in\mathcal{R} \tag{5}$$

$$Q_{i1} \geq 0 \qquad i\in\mathcal{I} \tag{6}$$

$$W_{ij1} \geq 0 \qquad i\in\mathcal{I}, \quad j\in\mathcal{I} \tag{7}$$

$$I_{i1} \geq 0 \qquad i\in\mathcal{I}_c \tag{8}$$

$$Y_{it} \in \{0,1\} \qquad i\in\mathcal{I}, \quad t\in\mathcal{H}. \tag{9}$$

The objective function (1) includes the setup costs for all products and all periods, the production and consumption costs for all items in period 1, the inventory costs of components in period 1, and the future costs $\mathcal{F}_2(Y_{\mathcal{I}\mathcal{H}}, Q_{\mathcal{I}1}, I_{\mathcal{I}_e 0}, I_{\mathcal{I}_c 1}, B_{\mathcal{I}_e 0})$ depending on the decisions made in the first stage. The definition of the future cost function $\mathcal{F}_t(.)$ for a given period t is given in equation (21). Constraints (2) compute the inventory levels of components, and constraints (3) determine the consumed quantities. Constraints (4) set the quantity to 0 if there is no setup, where the value of M_i is the minimum between the lower bound M_i^1 on the production quantities computed from the demands (10) and the lower bound M_i^2 computed from the resource capacities (11). The capacity constraints are given in (5). The bounds M_i^1 and M_i^2 are computed as follows:

$$M_i^1 = \begin{cases} \max_{\phi\in\Phi} \sum_{t\in\mathcal{H}} D_{it}^{\phi} & \text{if } i\in\mathcal{I}_e \\ \sum_{j\in\mathcal{I}} \left(\sum_{k|i\in\mathcal{A}_k} R_{kj}\right) M_j^1 & \text{if } i\in\mathcal{I}_c \end{cases} \tag{10}$$

$$M_i^2 = \min_{r\in\mathcal{R}|K_{ir}>0} \frac{C_r}{K_{ir}}. \tag{11}$$

Subproblem of stage $t \in \{2,\ldots,T\}$. In stage $t \in \{2,\ldots,T\}$, the state of the system is given by the decisions $(\widehat{Y}_{\mathcal{IH}}, \widehat{Q}_{\mathcal{I}[t-1]}, \widehat{I}_{\mathcal{I}_e[t-2]}, \widehat{I}_{\mathcal{I}_c[t-1]}, \widehat{B}_{\mathcal{I}_e[t-2]})$ made in the previous stages (the notation $\widehat{X}$ is used to represent the set of values of the corresponding decision variables X determined in the previous stages prior to t). The end-item demand of period $t-1$ unfolds, and this is captured in the scenario $\omega_{t-1} \in \Omega_{t-1}$. With this information, the inventory level (denoted by $I_{it-1}^{\omega_{t-1}}$) and the backlog level (denoted by $B_{it-1}^{\omega_{t-1}}$) for end-item i at the end of period $t-1$ for scenario ω_{t-1} can be computed directly from the decisions made and demand realizations prior to period t. Based on this information, the subproblem of stage t computes the quantities $Q_{it}^{\omega_{t-1}}$ and $W_{ijt}^{\omega_{t-1}}$. Similarly to the stage 1 subproblem, the inventory level $I_{it}^{\omega_{t-1}}$ in period t for components i in scenario ω_{t-1} depends on the internal demand, and it can be computed in stage t. The decisions are chosen to minimize the costs given the state of the system at the beginning of stage t. Note that we use the notation $[a\ b]$ to denote the augmented matrix of matrices a and b. The formulation of the stage $t \in \{2,\ldots,T\}$ subproblem for scenario ω_{t-1} is defined as follows:

$\mathcal{G}_t(\widehat{Y}_{\mathcal{IH}}, \widehat{Q}_{\mathcal{I}[t-1]}, \widehat{I}_{\mathcal{I}_e[t-2]}, \widehat{I}_{\mathcal{I}_c[t-1]}, \widehat{B}_{\mathcal{I}_e[t-2]}, D_{\mathcal{I}_e t-1}^{\omega_{t-1}}) =$

$$\text{Min} \quad \sum_{i\in\mathcal{I}} v_i Q_{it}^{\omega_{t-1}} + \sum_{i\in\mathcal{I}_c} h_i I_{it}^{\omega_{t-1}} + \sum_{i\in\mathcal{I}_e} \left(b_i B_{it-1}^{\omega_{t-1}} + I_{it-1}^{\omega_{t-1}}\right) + \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}} a_{ij} W_{ijt}^{\omega_{t-1}} \tag{12}$$

$$+ \mathcal{F}_{t+1}\big(\widehat{Y}_{\mathcal{IH}}, \left[\widehat{Q}_{\mathcal{I}[t-1]}\ Q_{\mathcal{I}t}^{\omega_{t-1}}\right], \left[\widehat{I}_{\mathcal{I}_e[t-2]}\ I_{\mathcal{I}_e t-1}^{\omega_{t-1}}\right], \left[\widehat{I}_{\mathcal{I}_c\ [t-1]} I_{\mathcal{I}_c t}^{\omega_{t-1}}\right], \left[\widehat{B}_{\mathcal{I}_e[t-2]}\ B_{\mathcal{I}_e t-1}^{\omega_{t-1}}\right]\big)$$

$$\text{s.t.} \quad \widehat{Q}_{it-L_i-1} + \widehat{I}_{it-2} - \widehat{B}_{it-2} - D_{it-1}^{\omega_{t-1}} - I_{it-1}^{\omega_{t-1}} + B_{it-1}^{\omega_{t-1}} = 0 \qquad i \in \mathcal{I}_e \tag{13}$$

$$\widehat{Q}_{it-L_i} + \widehat{I}_{it-1} - \sum_{j\in\mathcal{I}} W_{ijt} - I_{it}^{\omega_{t-1}} = 0 \qquad i \in \mathcal{I}_c \tag{14}$$

$$\sum_{i\in\mathcal{A}_k} W_{ikt}^{\omega_{t-1}} = \sum_{j\in\mathcal{I}} R_{kj} Q_{jt}^{\omega_{t-1}} \qquad k \in \mathcal{I}_c, \quad j \in \mathcal{I} \tag{15}$$

$$Q_{it}^{\omega_{t-1}} \le M_i \widehat{Y}_{it} \qquad i \in \mathcal{I} \tag{16}$$

$$\sum_{i\in\mathcal{I}} Q_{it}^{\omega_{t-1}} K_{ir} \le C_r \qquad r \in \mathcal{R} \tag{17}$$

$$B_{it-1}^{\omega_{t-1}}, \quad I_{it-1}^{\omega_{t-1}}, \quad Q_{it}^{\omega_{t-1}} \ge 0 \qquad i \in \mathcal{I}_e \tag{18}$$

$$W_{ijt}^{\omega_{t-1}} \ge 0 \qquad i \in \mathcal{I}, \quad j \in \mathcal{I} \tag{19}$$

$$I_{it}^{\omega_{t-1}}, \quad Q_{it}^{\omega_{t-1}} \ge 0 \qquad i \in \mathcal{I}_c. \tag{20}$$

The objective function (12) includes the production costs, the inventory costs of the components in period t, the backlog and inventory costs of end-items in period $t-1$, the component substitution costs, and the future costs $\mathcal{F}_{t+1}(\cdot)$.

This future cost corresponds to the average expected cost from period $t+1$ to T, and it is computed as:

$$\mathcal{F}_{t+1}(\widehat{Y}_{\mathcal{IH}}, \widehat{Q}_{\mathcal{I}[t]}, \widehat{I}_{\mathcal{I}_e[t-1]}, \widehat{I}_{\mathcal{I}_c[t]}, \widehat{B}_{\mathcal{I}_e[t-1]}) = \sum_{\omega_t \in \Omega_t} p_{\omega_t} \; \mathcal{G}_{t+1}(\widehat{Y}_{\mathcal{IH}}, \widehat{Q}_{\mathcal{I}[t]}, \widehat{I}_{\mathcal{I}_e[t-1]}, \widehat{I}_{\mathcal{I}_c[t]}, \widehat{B}_{\mathcal{I}_e[t-1]}, D^{\omega_t}_{\mathcal{I}_e t}). \tag{21}$$

Constraints (13) and (14) compute the inventory and backlog levels for end-items and components, respectively. Also, constraints (15), (16), and (17) are the consumption, setup, and capacity constraints, respectively.

Last-stage subproblem. The last stage $(T+1)$ computes the level of inventory and end-of-horizon backlog based on the decisions made in previous stages, where the variables include the end-item inventory $I^{\omega_T}_{iT}$ and the end-of-horizon backlog $B^{\omega_T}_{iT}$ for end-item i in period T for scenario $\omega_T \in \Omega_T$:

$\mathcal{G}_{T+1}(\widehat{Q}_{\mathcal{I}_e[T]}, \widehat{I}_{\mathcal{I}_e[T-1]}, \widehat{I}_{\mathcal{I}_c[T]}, \widehat{B}_{\mathcal{I}_e[T-1]}, D^{\omega_T}_{\mathcal{I}_e T}) =$

$$\text{Min} \quad \sum_{i \in \mathcal{I}_e} (h_i I^{\omega_T}_{iT} + e_i B^{\omega_T}_{iT}) \tag{22}$$

$$\text{s.t.} \quad \widehat{Q}_{iT-L_i} + \widehat{I}_{iT-1} - \widehat{B}_{iT-1} - D^{\omega_T}_{iT} - I^{\omega_T}_{iT} + B^{\omega_T}_{iT} = 0 \qquad i \in \mathcal{I}_e \tag{23}$$

$$I^{\omega_T}_{iT}, \quad B^{\omega_T}_{iT} \quad \geq \quad 0 \qquad i \in \mathcal{I}_e. \tag{24}$$

The objective (22) includes the end-of-horizon backlog and inventory costs of end-items, and their values are computed with constraints (23).

Finally, note that the considered problem is NP-hard because it extends the deterministic uncapacitated multiechelon lot-sizing problem with a general BOM that is NP-hard (Arkin et al. 1989).

3.2. Scenario Tree Formulation of CMLCS

This section provides the scenario tree formulation of the CMLCS. This alternative formulation is an MILP that can be implemented in a commercial solver, and we use it to benchmark the proposed SDDP approach. In addition, the PH approach proposed in Section 5.1 relies on the scenario tree formulation. The scenario tree formulation accounts for the realization of the demands in all periods in a single model by considering the scenario path ϕ rather than the scenario node ω_t. As a result, the variables are indexed by scenario path rather than scenario node. More precisely, Q^{ϕ}_{it}, B^{ϕ}_{it}, and I^{ϕ}_{it} denote, respectively, the

production quantity, backlog level, and inventory level for item i in period t in scenario path ϕ. In addition, W^{ϕ}_{ijt} denotes the quantity of item i consumed to produce j in period t in scenario path ϕ. Let $D^{\phi}_{\mathcal{I}_e[t]}$ denote the demand matrix of a scenario path ϕ that contains the values of the demand realizations $D^{\phi}_{i\tau}$ for each item $i \in \mathcal{I}_e$ for each period $\tau \in \{1, \ldots, t\}$. Note that the size of this model grows exponentially when the number of periods increases. The CMLCS can be formulated as follows:

$$\min \sum_{\phi \in \Phi} \sigma_\phi \left(\sum_{t \in \mathcal{H}} \sum_{i \in \mathcal{I}} \left(h_i I^{\phi}_{it} + s_i Y_{it} + v_i Q^{\phi}_{it} + \sum_{j \in \mathcal{I}} a_{ij} W^{\phi}_{ijt} \right) + \sum_{i \in \mathcal{I}_e} \left(\sum_{t=1}^{t=T-1} b_i B^{\phi}_{it} + e_i B^{\phi}_{iT} \right) \right) \quad (25)$$

$$\text{s.t.} \quad I^{\phi}_{it-1} - B^{\phi}_{it-1} + Q^{\phi}_{it-L_i} - D^{\phi}_{it} - I^{\phi}_{it} + B^{\phi}_{it} = 0 \qquad i \in \mathcal{I}_e,\ t \in \mathcal{H},\ \phi \in \Phi \quad (26)$$

$$I^{\phi}_{it-1} + Q^{\phi}_{it-L_i} - \sum_{j \in \mathcal{I}} W^{\phi}_{ijt} - I^{\phi}_{it} = 0 \qquad i \in \mathcal{I}_c,\ t \in \mathcal{H},\ \phi \in \Phi \quad (27)$$

$$\sum_{i \in \mathcal{A}_k} W^{\phi}_{ikt} = \sum_{j \in \mathcal{I}} R_{kj} Q^{\phi}_{jt} \qquad k \in \mathcal{I}_c,\ t \in \mathcal{H},\ \phi \in \Phi \quad (28)$$

$$Q_{it} \leq M_i Y_{it} \qquad i \in \mathcal{I},\ t \in \mathcal{H} \quad (29)$$

$$\sum_{i \in \mathcal{I}} K_{ir} Q^{\phi}_{it} \leq C_r \qquad t \in \mathcal{H},\ r \in \mathcal{R},\ \phi \in \Phi \quad (30)$$

$$W^{\phi}_{ijt+1} = W^{\phi'}_{ijt+1}, \quad Q^{\phi}_{it+1} = Q^{\phi'}_{it+1} \qquad i \in \mathcal{I},\ t \in \mathcal{H},\ \phi, \phi' | D^{\phi}_{\mathcal{I}_e[t]} = D^{\phi'}_{\mathcal{I}_e[t]} \quad (31)$$

$$B^{\phi}_{it} \geq 0 \qquad i \in \mathcal{I}_e,\ t \in \mathcal{H},\ \phi \in \Phi \quad (32)$$

$$I^{\phi}_{it} \geq 0 \qquad i \in \mathcal{I},\ t \in \mathcal{H},\ \phi \in \Phi \quad (33)$$

$$W^{\phi}_{ijt} \geq 0 \qquad i, j \in \mathcal{I},\ t \in \mathcal{H},\ \phi \in \Phi \quad (34)$$

$$Q^{\phi}_{it} \geq 0 \qquad i \in \mathcal{I},\ t \in \mathcal{H},\ \phi \in \Phi \quad (35)$$

$$Y_{it} \in \{0, 1\} \qquad i \in \mathcal{I},\ t \in \mathcal{H}. \quad (36)$$

The objective function (25) is the expected sum of inventory costs, setup costs, unit production costs, backlog costs, end-of-horizon backlog costs, and component substitution costs. The backlog and inventory quantities are computed with constraints (26) for end-items and constraints (27) for components. Constraints (28) compute the component consumption associated with the production quantities. Constraints (29) set the production quantities to zero in periods without setups. Finally, constraints (30) and (31) are the capacity and non-anticipativity constraints, respectively. More specifically, constraints (31) enforce equal production quantities in period $t+1$ for all scenario paths ϕ with identical demand

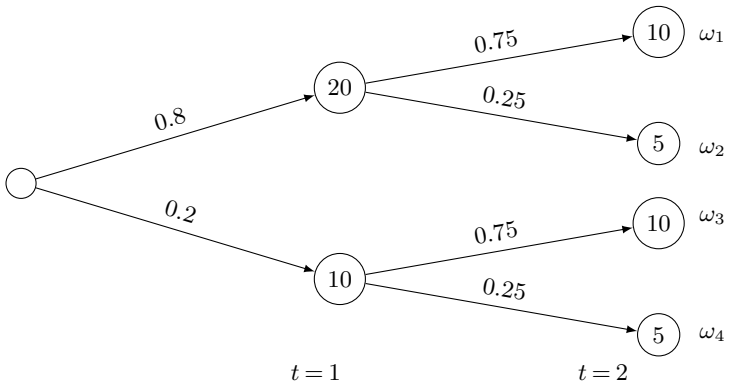

Figure 1 **Example of symmetric scenario tree** [2, 2]

realizations $D^{\phi}_{\mathcal{I}_e[t]}$ from period 1 to t. Note that the inventory and backlog levels can be directly computed from the quantities, consumptions, and demands up to period $t-1$. Consequently, the non-anticipativity constraints for the backlog and inventory variables are implicit.

4. Stochastic Dual Dynamic Programming for the CMLCS

This section describes the proposed SDDP algorithm for the CMLCS. Section 4.1 gives a generic description of SDDP, and Section 4.2 describes its adaptation to the CMLCS. Finally, Section 4.3 presents the enhancements for SDDP.

4.1. The SDDP Framework

To avoid the exponential growth in the number of states, SDDP assumes that the probability distributions are stagewise independent. Under this assumption, the complete set of scenario paths Φ corresponds to a symmetric scenario tree, where each node represents a realization of the stochastic parameters, and each path is a scenario. Figure 1 shows such a tree for the single end-item CMLCS, with the realization of the demand on each node and the probability associated with the realization on the edge leading to each node. As the tree is symmetric, all the nodes of stage t have branches toward the same scenario set Ω_{t+1} for the parameters revealed in stage $t+1$, and the branches' probability toward the same sample is identical. Consequently, the stochastic process can be represented with a sample Ω_t for each period $t \in T$.

SDDP decomposes the stochastic optimization problem per decision stage. Given the decisions $x_{[t-1]}$ made in stages 1 to $t-1$ and the realization $\omega_{[t]}$ of the stochastic parameters in stages 1 to t, the subproblem of stage t is to find the decisions x_t optimizing the costs

of stage t plus the cost-to-go. This cost-to-go function represents the expected costs from stage $t+1$ to T as a function of the decisions $x_{[t]}$ made from stage 1 to t and the realization $\omega_{[t]}$ of the stochastic parameters in stages 1 to t. As the cost-to-go functions are unknown at the start, SDDP iteratively builds an outer approximation of these functions. The reader is referred to Pereira and Pinto (1991) and Shapiro (2011) for more information about SDDP.

4.2. SDDP for CMLCS

At each iteration of SDDP, a forward pass generates a feasible solution, and a backward pass adds information about the cost-to-go functions. The forward and backward steps are described below, and they are followed by a discussion on the stopping criterion.

4.2.1. Forward Pass The forward pass simulates the use of the current policy for a set Ξ of scenario paths re-sampled from the scenario path set Φ at each iteration. For each sampled scenario path $\xi \in \Xi$, the subproblems are solved starting from stage $t = 1$ until stage $t = T$. The state of the system in stage t comes from the solution of the forward pass for sampled scenario path ξ in the previous stages. This state can be described by the initial inventory $\widehat{I}^{\xi}_{it-2}$, the initial backlog $\widehat{B}^{\xi}_{it-2}$, and the delivered quantity $\widehat{Q}^{\xi}_{it-L_i-1}$ for end-items and by $\widehat{I}^{\xi}_{it-1}$ and $\widehat{Q}^{\xi}_{it-L_i}$ for components.

In SDDP, the cost-to-go function in each stage t is described with a set of hyperplanes $\mathcal{F}^h_2(.) \in \mathcal{L}_t$ (built during the backward pass). The objective function in stage 1 becomes

$$\text{Min} \sum_{i \in \mathcal{I}} \left(\sum_{t \in \mathcal{H}} s_i Y_{it} + v_i Q_{i1} + \sum_{j \in \mathcal{I}} a_{ij} W_{ij1} \right) + \sum_{i \in \mathcal{I}_c} h_i I_{i1} + F_2, \tag{37}$$

where F_2 is a variable representing the future costs, and these costs are computed with the constraints:

$$F_2 \geq \mathcal{F}^h_2(Y_{\mathcal{I}\mathcal{H}}, Q_{\mathcal{I}1}, I_{\mathcal{I}_e 0}, I_{\mathcal{I}_c 1}, B_{\mathcal{I}_e 0}) \quad h \in \mathcal{L}_2. \tag{38}$$

As SDDP builds the set of hyperplanes iteratively, $\mathcal{L}_2$ is empty in the first iteration and thus we impose the lower bound on the objective function by adding the constraint

$$F_2 \geq 0. \tag{39}$$

Similarly, the cost-to-go function $\mathcal{F}_{t+1}(.)$ is approximated with a set of hyperplanes $\mathcal{L}_{t+1}$. The objective function becomes

$$\text{Min} \sum_{i \in \mathcal{I}} v_i Q^{\omega_{t-1}}_{it} + \sum_{i \in \mathcal{I}_c} h_i I^{\omega_{t-1}}_{it} + \sum_{i \in \mathcal{I}_e} \left(b_i B^{\omega_{t-1}}_{it-1} + I^{\omega_{t-1}}_{it-1} \right) + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} a_{ij} W^{\omega_{t-1}}_{ijt} + F_{t+1}, \tag{40}$$

and F_{t+1} is computed from the following constraints:

$$F_{t+1} \geq \mathcal{F}^h_{t+1}\left(\widehat{Y}_{\mathcal{I}\mathcal{H}}, \left[\widehat{Q}^{\xi}_{\mathcal{I}[t-1]}, Q^{\omega_{t-1}}_{\mathcal{I}t}\right], \left[\widehat{I}^{\xi}_{\mathcal{I}_e[t-2]}, I^{\omega_{t-1}}_{\mathcal{I}_e t-1}\right], \right. \tag{41}$$

$$\left. \left[\widehat{I}^{\xi}_{\mathcal{I}_c[t-1]}, I^{\omega_{t-1}}_{\mathcal{I}_c t}\right], \left[\widehat{B}^{\xi}_{\mathcal{I}_e[t-2]}, B^{\omega_{t-1}}_{\mathcal{I}_e t-1}\right] \right) \qquad h \in \mathcal{L}_{t+1}, \tag{42}$$

and

$$F_{t+1} \quad \geq \quad 0. \tag{43}$$

4.2.2. Backward Pass To compute the cost-to-go associated with the scenarios in Ω_t given the decisions made in the previous stages, the backward pass solves the subproblem of each stage t (from $t = T$ to $t = 2$) for all scenarios ω_t in Ω_t. To improve the approximation of the cost-to-go in stage $t-1$, SDDP adds a cut inferred from the dual of the stage t subproblems (i.e., Benders optimality cuts) to the subproblem of stage $t-1$.

Backward pass in stage $T+1$. The dual of subproblem (22)–(24) is

$$\text{Max} \qquad \sum_{\omega_T \in \Omega_T} p_{\omega_T} \sum_{i \in \mathcal{I}_e} \lambda_i^{k\omega_T} \left(-\widehat{Q}^{\xi}_{iT-L_i} - \widehat{I}^{\xi}_{iT-1} + \widehat{B}^{\xi}_{iT-1} + D^{\omega_T}_{iT}\right) \tag{44}$$

$$\text{s.t.} \qquad -\lambda_i^{k\omega_T} \leq h_i \qquad \forall i \in \mathcal{I}_e \tag{45}$$

$$\lambda_i^{k\omega_T} \leq e_i \qquad \forall i \in \mathcal{I}_e. \tag{46}$$

Consequently, the hyperplane

$$\sum_{\omega_T \in \Omega_T} p_{\omega_T} \sum_{i \in \mathcal{I}_e} \lambda_i^{k\omega_T} \left(-\widehat{Q}^{\xi}_{iT-L_i} - I^{\omega_{T-1}}_{iT-1} + B^{\omega_{T-1}}_{iT-1} + D^{\omega_T}_{iT}\right) \tag{47}$$

is added to $\mathcal{L}_T$, where $\lambda_i^{k\omega_T}$ is the dual variable of constraint (23) for product i in scenario ω_T at iteration k. Note that $I^{\omega_{T-1}}_{iT-1}$ and $B^{\omega_{T-1}}_{iT-1}$ are the variables of stage T associated with scenario $\omega_{T-1} \in \Omega_{T-1}$, whereas the variables $\widehat{Q}^{\xi}_{iT-L_i}$ are fixed during the forward pass at stage $T - L_i$. As the hyperplane is added to $\mathcal{L}_T$, it will be considered in the subproblem of stage T.

Backward pass at stage $t \in \{2, \ldots, T\}$. The cuts generated from the subproblems in stage $t \in \{2, \ldots, T\}$ are also based on the objective function of the dual subproblem. In the first iteration of the algorithm, the dual of the subproblem in stage t is

$$\sum_{\omega_{t-1}\in\Omega_{t-1}} p_{\omega_{t-1}}\Bigg(\sum_{i\in\mathcal{I}_e}\kappa_i^{k\omega_{t-1}}\Big(-\widehat{Q}^{\xi}_{it-L_i-1}-I^{\omega_{t-2}}_{it-2}+B^{\omega_{t-2}}_{it-2}+D^{\omega_{t-1}}_{it-1}\Big)$$
$$+\sum_{i\in\mathcal{I}_c}\eta_i^{k\omega_{t-1}}\Big(-\widehat{Q}^{\xi_{[t-L_i-1]}}_{it-L_i}-I^{\omega_{t-1}}_{it-1}\Big)$$
$$+\sum_{i\in\mathcal{I}}\theta_i^{k\omega_{t-1}}M\widehat{Y}_{it}$$
$$+\sum_{r\in\mathcal{K}}\chi_i^{k\omega_{t-1}}C_r\Bigg),$$

where $\kappa_i^{k\omega_{t-1}}$, $\eta_i^{k\omega_{t-1}}$, $\theta_i^{k\omega_{t-1}}$, and $\chi_i^{k\omega_{t-1}}$ denote, respectively, the dual variables of constraints (13), (14), (16), and (17) for item i in scenario $\omega_{t-1}\in\Omega_{t-1}$ in iteration k, and $\omega_{t-2}\in\Omega_{t-2}$ is the scenario of stage $t-1$.

However, SDDP iteratively generates a set of cuts in the subproblems, and these cuts modify the dual's objective value. For instance, at stage T, the cut (47) is added in each iteration, and the following element is added to the dual's objective:

$$\sum_{h\in\mathcal{L}_T}\pi_h^{\omega_{T-1}}\sum_{\omega_T\in\Omega_T}p_{\omega_T}\sum_{i\in\mathcal{I}_e}\lambda_i^{h\omega_T}(-\widehat{Q}^{\xi}_{iT-L_i}+D^{\omega_T}_{iT}),$$

where $\mathcal{L}_T$ denotes the set of cuts in stage T and π_h denotes the dual variables of cut h (added in iteration h). Thus, after solving the subproblem of stage T, the following cut is added to the subproblem of stage $T-1$:

$$F_T\geq\sum_{\omega_{T-1}\in\Omega_{T-1}}p_{\omega_{T-1}}\Big(\sum_{i\in\mathcal{I}_e}\kappa_i^{k\omega_{T-1}}(-\widehat{Q}^{\xi}_{iT-L_i-1}-I^{\omega_{T-2}}_{iT-2}+B^{\omega_{T-2}}_{iT-2}+D^{\omega_{T-1}}_{iT-1})$$
$$+\sum_{i\in\mathcal{I}_c}\eta_i^{k\omega_{T-1}}(-\widehat{Q}^{\xi}_{iT-L_i}-I^{\omega_{T-2}}_{iT-2})$$
$$+\sum_{i\in\mathcal{I}}\theta_i^{k\omega_{T-1}}M\widehat{Y}_{iT-1}$$
$$+\sum_{r\in\mathcal{K}}\chi_i^{k\omega_{T-1}}C_r$$
$$+\sum_{h\in\mathcal{G}_T}\pi_h^{\omega_{T-1}}\sum_{i\in\mathcal{I}_e}\lambda_i^{h\omega_T}(-\widehat{Q}^{\xi}_{iT-L_i}+D^{\omega_T}_{iT})\Big).$$

Note that if $L_i=1$, $\widehat{Q}^{\xi}_{iT-L_i}$ is a variable of the stage $T-1$ subproblem, whereas for $L_i>1$, $\widehat{Q}^{\xi}_{iT-L_i}$ is fixed to the value found in the previous stage.

We explain below the construction of the cut for any stage $t > T-1$. Each cut c in the set of cuts $\mathcal{C}_t$ of stage t can be expressed by

$$F_{t+1} \geq A^c_{t-1}\widehat{X}_{t-1} + A^c_t X_t + A^c_e \widehat{X}_e + A^c,$$

where X_t denotes the variables of stage t, $\widehat{X}_{t-1}$ is the set of variables of stage $t-1$, $\widehat{X}_e$ is the set of variables decided earlier than stage $t-1$, A^c_{t-1}, A^c_t, and A^c_e are the matrix of coefficients, and A^c denotes the constant on the right-hand side of cut c. These cuts increase the objective function value of the dual with

$$\sum_{h \in \mathcal{L}_t} \pi_c \left(A^c_{t-1} X_{t-1} + A^c_e \widehat{X}_e + A^c \right),$$

where π_c is the dual variable associated with cut c. Consequently, after resolving the subproblem at stage t, the following cut is added to the subproblem of stage $t-1$:

$$\begin{aligned} F_t \geq \sum_{\omega_{t-1} \in \Omega_{t-1}} p_{\omega_{t-1}} \Big(& \sum_{i \in \mathcal{I}_e} \kappa_i^{k\omega_{t-1}} (-\widehat{Q}^{\xi}_{it-L_i-1} - I^{\omega_{t-2}}_{it-2} + B^{\omega_{t-2}}_{it-2} + D^{\omega_{t-1}}_{it-1}) \\ & + \sum_{i \in \mathcal{I}_c} \eta_i^{k\omega_{t-1}} (-\widehat{Q}^{\xi}_{it-L_i} - I^{\omega_{t-2}}_{it-1}) \\ & + \sum_{i \in \mathcal{I}} \theta_i^{k\omega_{t-1}} M \widehat{Y}_{it} \\ & + \sum_{r \in \mathcal{K}} \chi_i^{k\omega_{t-1}} C_r \\ & + \sum_{h \in \mathcal{L}_t} \pi_c (A^c_{t-1} X_{t-1} + A^c_e \widehat{X}_e + A^c) \Big). \end{aligned} \tag{48}$$

Algorithms 1, 2, and 3 give the steps of SDDP, the forward pass, and the backward pass, respectively.

Algorithm 1 SDDP

Generate the first-stage MILP and the linear programs of the subsequent stages.

while the stopping criterion is not met **do**

Sample a set of scenario paths Ξ.

for each sampled scenario path $\xi \in \Xi$ **do**

Perform the forward pass (*Algorithm 2*) to find the solution for sampled scenario path ξ with the current policy.

Perform the backward pass (*Algorithm 3*) to improve the approximation of the cost-to-go.

end for

end while

Algorithm 2 Forward pass on a scenario path ξ

Solve the first-stage MILP (37), (2) - (9), (38), and (39) to get the values of $\widehat{Y}_{\mathcal{I}t}$ and $\widehat{Q}_{\mathcal{I}1}$

for $t = 2$ to $t = T$ **do**

Compute $\mathcal{F}_t(\widehat{Y}_{\mathcal{I}t}, [\widehat{Q}^{\xi}_{\mathcal{I}_e t-L_i-1}, \widehat{Q}^{\xi}_{\mathcal{I}_c t-L_i}], \widehat{I}^{\xi}_{\mathcal{I}_e t-2}, \widehat{I}^{\xi}_{\mathcal{I}_c t-1}, \widehat{B}^{\xi}_{\mathcal{I}_e t-2})$ and record the resulting values of $\widehat{Q}_{\mathcal{I}t},\ \widehat{I}_{\mathcal{I}_c t},\ \widehat{B}_{\mathcal{I}_c t},\ \widehat{I}_{\mathcal{I}_e t-1},\ \widehat{B}_{\mathcal{I}_e t-1}$.

end for

Algorithm 3 Backward pass

for $t = T+1$ to $t = 2$ **do**

for each scenario $\omega_t \in \Omega_t$ **do**

Modify the LP (40), (13)–(20), (42), and (43) of stage t according to the demand of scenario ω_t and the decisions made in previous stages 1 to $t-1$ of the forward pass, and solve this LP.

end for

Create a new cut for stage $t-1$ as shown in Equation (48).

end for

4.2.3. Stopping Condition On the one hand, as F_2 represents a partial outer-approximation of the cost-to-go, the solution of the first-stage subproblem gives a lower bound LB of the optimal solution. On the other hand, the forward pass gives an approximate upper bound UB since the forward pass gives the expected cost of the current policy

over the subset Ξ of scenario paths (the resulting solutions are feasible but the true cost is not calculated unless all the scenario paths are evaluated). However, the convergence of SDDP cannot be detected by a simple comparison of UB and LB because UB is only an approximate upper bound. Statistical proofs of convergence exist in the literature (Shapiro 2011), but they are seldom used in practice as a stopping condition. As these proofs are statistical, there is either a risk of stopping the algorithm too soon (and it will not provide a good solution) or the gap may be too large to be relevant. In practice, stopping the algorithm after a predefined number of iterations or a time limit is more reliable (De Matos et al. 2015, Soares et al. 2017b). Therefore, in this work, we stop the algorithm when a predetermined time limit is reached. The time limit criterion is also practical since a solution must be provided in a reasonable amount of time, but the algorithm remains valid if the user decides to use a different stopping criterion.

4.3. Enhancements to SDDP

To speed up the convergence of SDDP, several enhancements are proposed, namely, the generation of strong cuts, retaining the average scenario, generating multiple cuts in the backward pass, and advanced scenario sampling. These strategies are described below.

4.3.1. Generation of Strong Cuts Magnanti and Wong (1981) observed that the dual of a degenerate subproblem may have multiple optimal solutions and that each solution leads to the generation of a different Benders optimality cut. Therefore, the authors proposed to accelerate Benders decomposition by generating the strongest cuts. Later, Papadakos (2008) proposed a practical enhancement called the interior point method.

To solve the subproblem of stage t in the backward pass, the previous stage variables are not fixed to the forward pass decisions, but to the values of an approximate core point. In the first iteration, the approximate core point Y_{it}^C, $Q_{it-L_i}^C$, I_{it-1}^C, and B_{it-1}^C is set to the forward pass solution. Next, in each iteration k, the approximate core point is updated with the forward pass solution (Y_{it}^k, Q_{it}^k, I_{it}^k) of iteration k as follows:

$$Y_{it}^C = pY_{it}^C + (1-p)Y_{it}^k$$

$$Q_{it-L_i}^C = pQ_{it-L_i}^C + (1-p)Q_{it-L_i}^k$$

$$I_{it-1}^C = pI_{it-1}^C + (1-p)I_{it-1}^k$$

$$B^C_{it-1} = pB^C_{it-1} + (1-p)B^k_{it-1},$$

where $p < 1$ is a parameter (set to 0.5 in our implementation). Note that this implementation assumes $|\Xi| = 1$. The strong cuts are common in the literature on Benders decomposition but are not often used in SDDP. Nevertheless, they remain valid, and they are easy to implement.

4.3.2. Lower Bound Inequality from the Expected Demand Problem As SDDP builds iteratively the approximation of the cost-to-go, this approximation is not precise in the first iterations. However, the computation of the cost-to-go based on the expected demand scenario can help drive SDDP toward good solutions despite this inaccuracy.

Given a stochastic optimization program P such that the coefficients of the variables in the objective function and in the constraints are constant, Birge and Louveaux (2011) show that the cost of problem P solved with the scenario path of the expected demand is lower than or equal to the optimal solution of P. Consequently, the expected cost-to-go $\mathcal{F}_{t+1}(.)$ from period $t+1$ to T is larger than or equal to the cost-to-go computed with the average demand scenario.

To drive SDDP toward good solutions during the first few iterations, this lower bound is computed in each subproblem. As the lower bound is computed by retaining the average scenario path in the first stage problem, this approach is similar to the partial Benders decomposition proposed by Crainic et al. (2016). More precisely, the subproblems include the variables $\bar{Q}_{i\tau}$, $\bar{W}_{ij\tau}$, $\bar{I}_{i\tau}$, and $\bar{B}_{i\tau}$ for all item i and period τ in $\{t, ..., T\}$ to represent, respectively, the quantities, consumption, inventory, and backlog in the average scenario path, and the following constraints are added to each subproblem:

$$F_{t+1} \geq \sum_{\tau \in t...T} \left(\sum_{i \in \mathcal{I}} v_i \bar{Q}_{i\tau} + \sum_{i \in \mathcal{I}_c} h_i \bar{I}_{i\tau} + \sum_{i \in \mathcal{I}_e} \left(b_i \bar{B}_{i\tau} + \bar{I}_{i\tau} \right) + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} a_{ij} \bar{W}_{ij\tau} \right) \tag{49}$$

$$\widehat{Q}^{\xi}_{i\tau-L_i} + \widehat{I}^{\xi}_{i\tau-1} - \widehat{B}^{\xi}_{i\tau-1} - \bar{D}_{i\tau} - \bar{I}_{i\tau} + \bar{B}_{i\tau} = 0 \quad i \in \mathcal{I}_e, \quad \tau \in t+1 \dots T \tag{50}$$

$$\widehat{Q}^{\xi}_{i\tau-L_i} + \widehat{I}^{\xi}_{i\tau-1} - \sum_{j \in \mathcal{I}} \bar{W}_{ij\tau} - \bar{I}_{i\tau} = 0 \quad i \in \mathcal{I}_c, \quad \tau \in t+2 \dots T \tag{51}$$

$$\sum_{i \in \mathcal{A}_k} \bar{W}_{ij\tau} = R_{kj} \bar{Q}_{j\tau} \quad k \in \mathcal{I}_c, \quad j \in \mathcal{I}, \quad \tau \in t+1 \dots T \tag{52}$$

$$\bar{Q}_{i\tau} \leq M \widehat{Y}_{i\tau} \quad i \in \mathcal{I}, \quad \tau \in t+1 \dots T \tag{53}$$

$$\sum_{i\in\mathcal{I}} \bar{Q}_{i\tau} K_{ir} \quad \leq \quad C_r \qquad\qquad r\in\mathcal{R}, \quad \tau\in t+1\ldots T \tag{54}$$

$$\bar{B}_{i\tau}, \quad \bar{I}_{i\tau}, \quad \bar{Q}_{i\tau} \quad \geq \quad 0 \qquad\qquad i\in\mathcal{I}_e, \quad \tau\in t+1\ldots T \tag{55}$$

$$\bar{I}_{i\tau}, \quad \bar{Q}_{i\tau} \quad \geq \quad 0 \qquad\qquad i\in\mathcal{I}_c, \quad \tau\in t+2\ldots T. \tag{56}$$

Constraints (49) link the lower bound with the expected cost-to-go. Constraints (50) and (51) compute the inventory and backlog for each item, where $\widehat{Q}^{\xi}_{i\tau-L_i}$, $\widehat{I}^{\xi}_{i\tau-1}$, and $\widehat{B}^{\xi}_{i\tau-1}$ correspond (depending on the value of τ) either to the values found at an earlier stage, the decision variables of stage t, or the decisions of the average scenario. Finally, constraints (52), (53), (54), (55), and (56) correspond to consumption, production, capacity, and domain constraints.

4.3.3. Multi-cut SDDP In the multi-cut version of SDDP, the cost-to-go $F_{t+1\omega_t}$ is computed independently for each scenario ω_t of the next stage, and the expected cost-to-go is computed as

$$F_{t+1} = \sum_{\omega_t\in\Omega_t} p_{\omega_t} F_{t+1\omega_t}.$$

In other words, the cuts are added separately for each scenario $\omega_t \in \Omega_t$. For instance, the cut added for scenario ω_T in stage $T+1$ is

$$F_{T+1\omega_T} \geq \sum_{i\in\mathcal{I}_e} \lambda_i^{l\omega_T} \left(-\widehat{Q}^{\xi}_{iT-L_i} - I^{\omega_{T-1}}_{iT-1} + B^{\omega_{T-1}}_{iT-1} + D^{\omega_T}_{iT} \right).$$

On the one hand, the multi-cut version leads to a better approximation of the cost-to-go functions. On the other hand, as the number of constraints increases, the multi-cut version can potentially slow down the solution process.

4.3.4. Advanced Scenario Sampling The set of all possible demand scenarios is usually too large to solve the stochastic optimization problem. In this context, SDDP relies on a sample of scenarios, and the sampling method has a critical impact on the performance of SDDP (Parpas et al. 2015). For instance, the simple and efficient Crude Monte Carlo (CMC) (Metropolis and Ulam 1949) approach samples independently n demand scenarios, and each scenario has a probability $1/n$. While the CMC approximation is on average equal to the expected cost, the variance of the approximation is σ^2/n, where σ^2 is the variance of the expected optimal cost. Advanced scenario sampling methods such as Randomized Quasi-Monte Carlo (RQMC) (Cranley and Patterson 1976) generate samples resulting in

approximations with lower theoretical variances than CMC. In other words, RQMC leads to good approximations with fewer scenarios. As in Thevenin et al. (2021), to sample the demands with RQMC, we use the Lattice Builder tool (software that implements multiple algorithms to build good rank-1 lattice rules) (L'Ecuyer and Munger 2016) to generate n vectors $\boldsymbol{\alpha}$ (i.e., a lattice) evenly distributed in the cube $|\mathcal{I}_e|$. The demand vectors are then obtained by $\Omega_t = \{i \cdot \boldsymbol{\alpha}/n + \boldsymbol{\delta} \bmod 1 \quad \forall\, i \in 1 \dots n\}$, where $\boldsymbol{\delta}$ is a random value used to shift the lattice. Thus, a different value of $\boldsymbol{\delta}$ is generated for each sample we draw.

In this work, the scenarios Ω_t of the backward pass are sampled and fixed, whereas the forward pass scenario paths Ξ are resampled in each iteration from the original distribution.

5. Hybrid Heuristics with SDDP

Despite the proposed improvements, the experiments (see Section 6) show that SDDP takes too much time to converge. However, the method spends most of the computation time on solving the first-stage MILP. This imbalance between the computational complexity of the first stage and the other ones is related to the inherent structure of the problem. More specifically, the first-stage model contains all the binary variables, whereas the other stages are linear programs that can be efficiently solved. This imbalance in computational effort is common in stochastic programming when the first-stage problem comprises important design or planning decisions (Adulyasak et al. 2015, Santoso et al. 2005). To alleviate this issue, we present two heuristics usable in other applications of SDDP where the first stage is a combinatorial optimization problem. In the considered problem, if the setup variables are fixed, the resulting MILP of the first-stage subproblem becomes a linear program (LP), which allows the first-stage model to be solved more efficiently. This section presents two heuristic strategies to compute the setups, namely a PH approach (Section 5.1), and a heuristic variant of SDDP (Section 5.2).

5.1. Progressive Hedging

This section presents a method that determines the setups with a scenario tree approach. However, since planning over a long time horizon leads to large scenario trees, we consider PH to speed up the solution process.

PH (Rockafellar and Wets 1991) decomposes the original stochastic programming formulation into a set of deterministic problems, one per scenario path. Similar to Lagrangian relaxation, PH is applied to solve the model with scenario path index (25)–(36) by relaxing the non-anticipativity constraints (31), but it penalizes their violation in the objective

function. The non-anticipativity constraints (31) force the variables Q_{it}^{ϕ} (resp. W_{ijt}^{ϕ}) to be equal for all scenario paths with identical demand realizations up to period $t-1$. We denote this set of scenario paths by $NA(D_{\mathcal{I}[t-1]}^{\phi_{[t-1]}})$, and it contains the demand scenario path with demands equal to $D_{\mathcal{I}[t-1]}^{\phi_{[t-1]}}$ in the first $t-1$ periods. Also, the model (25)–(36) can be expressed with scenario-dependent setup variables (Y_{it}^{ϕ}) if additional non-anticipaticity constraints are included to force these variables to be equal for all scenario paths. Once the non-anticipativity constraints are relaxed, the mathematical model (25)–(36) can be decomposed per scenario path, but the resulting solution might not be implementable. To retrieve an implementable solution, the quantity $\widetilde{Q}_{it}^{\phi}$ and consumption $\widetilde{W}_{ijt}^{\phi}$ are averaged over the scenario paths in $NA(D_{\mathcal{I}[t-1]}^{\phi})$, whereas the setups are averaged over all scenario paths:

$$\widetilde{Q}_{it}^{\phi_{[t-1]}} = \frac{\sum_{\phi \in NA(D_{\mathcal{I}[t-1]}^{\phi_{[t-1]}})} \sigma_{\phi} Q_{it}^{\phi}}{\sum_{\phi \in NA(D_{\mathcal{I}[t-1]}^{\phi_{[t-1]}})} \sigma_{\phi}}$$

$$\widetilde{W}_{ijt}^{\phi_{[t-1]}} = \frac{\sum_{\phi \in NA(D_{\mathcal{I}[t-1]}^{\phi_{[t-1]}})} \sigma_{\phi} W_{ijt}^{\phi}}{\sum_{\phi \in NA(D_{\mathcal{I}[t-1]}^{\phi_{[t-1]}})} \sigma_{\phi}}$$

$$\widetilde{Y}_{it} = \sum_{\phi \in \Phi} \sigma_{\phi} Y_{it}^{\phi}.$$

The following term is included in the objective function to penalize the deviation from the implementable solution:

$$\sum_{x \in \mathcal{V}} \Lambda_x^k (x - \widetilde{x}) + \frac{\rho}{2}(x - \widetilde{x})^2,$$

where $\mathcal{V} = \{\ W_{ijt}^{\phi},\ Q_{it}^{\phi},\ Y_{it}^{\phi}\ \forall\ i,j \in \mathcal{I},\ t \in \mathcal{H},\ \phi \in \Phi\}$ is the set of decision variables subject to non-anticipativity constraints, $\widetilde{x}$ is the implementable decision associated with variable x, ρ is a parameter, and Λ_x^k is the Lagrangian multiplier in iteration k. This Lagrangian multiplier is updated for each variable $x \in \mathcal{V}$ at the end of each iteration as follows:

$$\Lambda_x^k = \Lambda_x^{k-1} + \rho(x^k - \widetilde{x}^k),$$

where x^k is the value of variable x at iteration k. Note that ρ is a sensitive parameter since it impacts the quality of the solution and the computation time required for convergence.

A large value of ρ leads to fast convergence to a suboptimal solution, whereas a small value leads to slow convergence. In our implementation, we use $\rho = 0.1$. Note that this value is determined using a preliminary experiment using different values of ρ and the results are included in the Online Supplement.

Finally, PH stops when the setup variables have converged, that is,

$$\sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{H}} \left(\widetilde{Y}_{it} - Y_{it}^{\phi} \right) \leq \epsilon,$$

where ϵ is set to 0.01 in this paper.

As explained earlier, we use PH to get the setup values, and then SDDP optimizes the production quantities. Even though the hybrid PH SDDP relies on a scenario tree, it is convenient in a reactive context because PH is only used to compute the setups (first-stage variables) and SDDP is run afterward. Consequently, the problem can be resolved dynamically in short computation time thanks to the approximation of the cost-to-go.

5.2. Heuristic SDDP

Heuristic SDDP (HSDDP) starts with initial setup values and iteratively performs the following steps: (1) run SDDP with fixed setups until the lower bound stabilizes (i.e., 10 iterations without improvement of LB) and (2) solve the first-stage problem (where the setups are not fixed) to find potentially better setup values.

The initial setups are computed with the two-stage approximation of model (25)–(36). In this two-stage approximation, the quantity and consumption variables for the entire horizon are determined in the first stage. More precisely, the model is similar to (25)–(36), but variables W_{ijt}^{ϕ} and Q_{it}^{ϕ} become W_{ijt} and Q_{it}, respectively, and the non-anticipativity constraints (31) are removed. In other words, the resulting model ignores the dynamic decision framework. However, the setup values found with the two-stage approximation are feasible (but suboptimal) with respect to the original model (25)–(36). Since the two-stage model does not need a scenario tree, the two-stage approximation model can be efficiently solved in a reasonable amount of time.

6. Experiments

Section 6.1 describes the considered instances, and Sections 6.2 to 6.6 present computational experiments to evaluate the performance of SDDP and the proposed heuristic methods.

The algorithms are implemented in Python with CPLEX 12.9, and the tests are performed with an Intel Xeon Brodwell EP E5-2630v4 2.20GHz processor. We use CPLEX with the default settings. For all methods, the stopping criterion is a time limit of $900|T|$ seconds.

The evaluation of the proposed SDDP policies is based on a simulation over a set of 5000 scenario paths (which are drawn independently from the scenario sample Ω_t used for optimization). Each scenario path represents a possible realization of the demand for the entire planning horizon, and these scenario paths are sampled independently with Monte Carlo following the probability distributions of the demands. Note that for a given instance, all methods are evaluated with the same set of scenario paths. As mentioned earlier, SDDP reacts quickly to new information, and this simulation is not time-consuming. In fact, the evaluation corresponds to a forward pass of SDDP on the given set of scenario paths. This simulation returns an unbiased cost estimate of using each method m, which we call approximated upper bound and denote $\widehat{UB}(m)$. In the analysis of the results, we report the percentage gap (GAP) between the approximated upper bound of the considered methods:

$$GAP = 100 \ \frac{\widehat{UB}(m) - \widehat{UB}^{\star}}{\widehat{UB}^{\star}}, \tag{57}$$

where $\widehat{UB}^{\star}$ is the best approximated upper bound among all methods tested in the considered experiment.

6.1. Generation of Instances

The experiments are performed with the well-known multiechelon lot-sizing instances from Tempelmeier and Derstroff (1996). As these instances do not include lead times and demand distributions, these elements are generated similarly to Thevenin et al. (2021) as explained in the Online Supplement.

To study the performance of the methods under various conditions while maintaining a reasonable number of instances, we select a subset of instances from Tempelmeier and Derstroff (1996) with a TBO of 1 or 3, a capacity utilization of 90% or 50%, and the two BOM given in Figure 2.

We generate instances with different values for the planning horizon ($T \in \{2, 4, 6, 8, 10\}$), the number of alternates ($a \in \{0, 2, 4, 6\}$), and the substitution costs ($a_{ij} \in \{0, 0.1, 1\}$). We set the default values for these parameters to $T = 4$, $a_{ij} = 0.1$, and $a = 4$, and we vary one factor at a time. This leads to a total of 80 instances.

Assembly BOM General BOM

Figure 2 Considered BOM

6.2. Performance of SDDP

This section evaluates the performance of SDDP with different scenario sample sizes $|\Omega_t| \in \{5, 10, 20, 50, 100\}$. To avoid convergence and memory problems, the experiments are performed with the setups fixed to the solution of the two-stage approximation of model (25)–(36) and with the single-cut version of SDDP.

Table 1 reports the average GAP obtained with each sample size, and the value in bold is the smallest GAP for each value of the planning horizon. Table 1 shows that the sample size is a sensitive parameter. A too small number of scenarios leads to a bad approximation of the stochastic process, whereas a too large number prevents SDDP from converging. In fact, the sample size should depend on the planning horizon length. For 2 periods, 50 scenarios per stage lead to the best solution, whereas for 8 periods, 5 scenarios per stage lead to the best results.

In the rest of the experiments, SDDP is run with 20 scenarios per decision stage since this scenario number gives the smallest average GAP.

	$\|\Omega_t\|$				
$\|T\|$	5	10	20	50	100
2	0.24	0.09	0.03	**0.00**	**0.00**
4	1.18	0.19	**0.01**	0.21	0.23
6	0.85	0.20	**0.00**	0.60	0.72
8	**0.08**	0.11	0.19	0.81	2.05
10	**0.05**	0.06	0.25	1.25	1.89
Average	0.82	0.16	**0.05**	0.38	0.59

Table 1 Average GAP obtained with SDDP for different number of scenarios per stage

6.3. SDDP Enhancements

To evaluate the impact of the proposed SDDP enhancements, this section reports the performance of multiple variants of SDDP. We compare SDDP with all enhancements and the setup fixed with the two-stage heuristic, denoted by *Default*. We compare this Default version with Default without the multi-cut, Default without the generation of strong cuts, Default without the average scenario lower bound, Default without RQMC sampling, Default without fixed setups, and the Standard SDDP where no enhancement is used.

Table 2 reports the average percentage gap GAP^{UB} (resp., GAP^{LB}) between the upper bound (resp. lower) obtained with each variant of SDDP and the upper (resp. lower) bound obtained with the Default SDDP algorithm. In addition, Table 2 gives the estimated optimality gap for each method GAP^{Opt}. We detail below the computation of GAP^{UB}, GAP^{LB}, and GAP^{Opt} for each method m.

$$\begin{aligned} GAP^{UB}(m) &= 100\ \frac{\widehat{UB}(m) - \widehat{UB}(\text{Default})}{\widehat{UB}(\text{Default})} \\ GAP^{LB}(m) &= 100\ \frac{LB(m) - LB(\text{Default})}{LB(\text{Default})} \\ GAP^{Opt} &= 100\ \frac{\widehat{UB}(m) - LB(m)}{LB(m)}. \end{aligned} \tag{58}$$

Table 2 shows that a simple adaptation of SDDP is not efficient for the CMLCS with an optimality gap of 42.74%. The binary setup variables in the first-stage subproblem significantly slow down the convergence of SDDP since the optimality gap is 40.07% when the setups are not fixed. In addition, RQMC sampling has a significant impact since removing this component increases the upper bound by 0.62%. Similarly, the multi-cut version improves the convergence since removing this enhancement increases the upper bound by 0.19%. On the contrary, the strong cuts and the average scenario lower bound have a limited impact, and they decrease the upper bound by 0.02% and 0.10%, respectively.

Figure 3 reports the lower bound value in each iteration for the different variants of SDDP on the instance with general BOM, $|\mathcal{H}| = 6$, $TBO = 1$, a utilization of 90%, four alternates, and an alternate cost of 0.1. Contrary to Table 2, we report the value of the lower bound itself rather than the gap. Nevertheless, the gap is computed from the lower bound as explained in Equation (58).

	Default	Without multi-cut	Without strong cuts	Without avg scenario	Without RQMC	Without fixed setup	Standard SDDP
GAP^{UB}	-	0.19	0.02	0.10	0.62	23.73	21.4
GAP^{LB}	-	-0.09	0.01	0.01	-2.02	-6.52	-12.29
GAP^{Opt}	0.07	0.36	0.08	0.16	2.84	40.07	42.74

Table 2 Performance of each improvement of SDDP

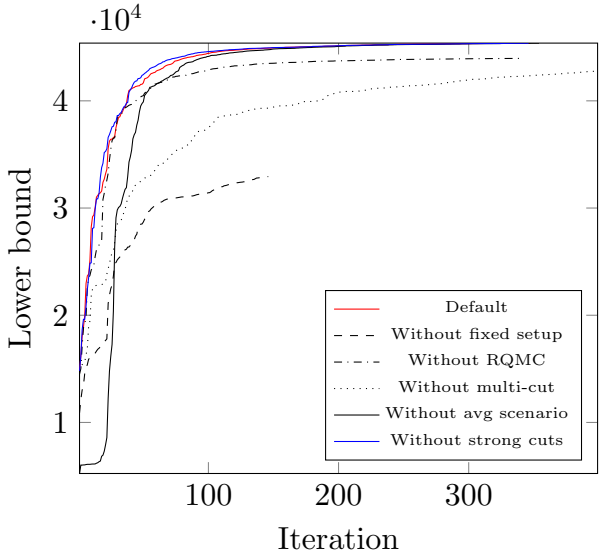

Figure 3 Evolution of the lower bound for different versions of SDDP

Note that $Default$ without multi-cut performs more than 2000 iterations within the time limit, but the last iterations are not shown for the sake of presentation. Figure 3 confirms that the fixed setups strategy and the multi-cut version of SDDP speed up the convergence of SDDP, and the versions of SDDP with and without strong cuts behave similarly. In addition, SDDP with RQMC converges to a better lower bound than SDDP with MC. Finally, retaining the average scenario path improves significantly the lower bound, but after a sufficient number of iterations, the variant of SDDP without this improvement reaches the same lower bound as the default version. In the rest of the experiments, SDDP includes all the improvements.

6.4. Comparison of the Heuristic SDDP

This section compares the performance of different heuristics proposed in Section 5.2. We decompose the HSDDP method into three methods (2S, 2S-SDDP, and HSDDP) to show the benefit of each component of HSDDP. The two-stage approximation originally presented in Thevenin et al. (2021) (denoted by 2S) is used to compute the initial setup

values in HSDDP. The version of SDDP with the setups fixed to the value of the two-stage approximation (denoted by 2S-SDDP) computes the production quantity associated with this initial setup decision. HSDDP extends 2S-SDDP by iteratively solving the first-stage model to generate a new setup vector before computing its associated production quantity. Finally, we also compare HSDDP with the version of SDDP where the setups are fixed to the value of PH (denoted by PH).

Table 3 reports the average GAP of each method based on different instance characteristics, where the GAP on each instance is computed by equation (57). The detailed results are available in the Online Supplement. The results show that HSDDP outperforms 2S-SDDP, 2S and PH with average GAPs of 0.30% versus 0.74%, 2.04%, and 20.22%, respectively. Table 3 also reports the average CPU time. The reported CPU time shows that solving the two-stage model is faster (i.e., 20 seconds on average), whereas the methods based on SDDP stop at the time limit (set to $900|T|$ seconds). Note that 2S refers to the two-stage approximation of model (25)–(36), and this method provides the initial setup values used in HSDDP. Since the two-stage approximation does not require a scenario tree, the resulting model can be solved optimally in a few seconds. Solving the multistage problem is much more complex than solving the two-stage model. This computational advantage of the two-stage heuristic is leveraged to determine the initial setup decisions in the HSDDP algorithm. Nevertheless, contrarily to the heuristic 2S that provides only an upper bound, the HSDDP procedure provides both lower and upper bounds to the multistage problem in a systematic fashion. Thus, the practitioner can benefit from this procedure, which leverages the fast heuristic (2S) and produces solution bounds while the solution quality can be iteratively improved. This allows the practitioner to terminate the algorithm when needed based on the computing time allowed to solve the instance.

6.5. Comparison of the Heuristic SDDP with Optimal Solutions

Thevenin et al. (2021) show the benefit of using multistage stochastic optimization for material requirements planning. More precisely, the best method in terms of solution quality in Thevenin et al. (2021) is the multistage stochastic program (denoted by M-stage in Thevenin et al. (2021)) based on a scenario tree. While this approach performs well, it does not scale for large planning horizons. This multistage model based on a scenario tree corresponds to the MILP model (25)–(36) of the present paper. This section compares the results of the proposed SSDP approach with the optimal costs on instances with a few

		2S	2S-SDDP	PH	HSDDP
$\|T\|$	2	0.01	0.00	8.43	0.13
	4	2.05	0.95	13.88	0.08
	6	2.23	0.96	21.16	0.45
	8	3.75	1.74	26.23	1.24
	10	3.99	1.73	27.32	0.39
Structure	General	2.33	1.02	18.63	0.23
	Assembly	2.22	1.07	25.30	0.22
Utilization	90%	3.85	1.89	5.94	0.24
	50%	0.70	0.20	37.99	0.21
TBO	1	1.83	0.67	22.30	0.31
	4	2.72	1.42	21.63	0.15
Average		2.04	0.74	20.22	0.30
CPU (sec.)		19.9	4486.4	4515.4	4583.2

Table 3 GAPs of 2S, 2S-SDDP, PH, and HSDDP

scenario paths. The optimal costs are computed with the model (25)–(36) and a symmetric scenario tree. Note that HSSDP is solved with the same set of scenarios as the MILP.

Table 4 provides the upper bound $\widehat{UB}$ obtained with HSDDP approximated based on a sample of 5000 scenario paths, the percentage gap between the exact HSDDP upper bound (computed on all optimization scenarios) and the optimal solution, and the time at which the best solution was found. Note that some combinations of planning horizon and number of scenarios per stage lead to a huge number of scenario paths. In such a case, it is impossible to compute the exact upper bound, and we indicate such instances using "-" in Table 4. For the MILP, Table 4 gives the actual CPU time spent to solve the instance, the optimality gap, and the number of variables and constraints after CPLEX presolve. For the instances that were too large to build the mathematical model, we provide an approximation of the raw number of variables and constraints. For some instances, the resulting mathematical model cannot be built since the size of the model was too large, and we indicate such instances using "-" in Table 4.

The results show that HSDDP performs well since the exact upper bound is only 1.2% (average of the values in the column "Gap exact UB" on the instances where the optimality gap can be obtained) larger on average than the optimal cost. The performance of the heuristic depends mainly on the number of periods in the horizon and not on the number of scenarios. Indeed, the gap remains 0 when the number of scenarios increases from 2 to 10 if

$\|\Omega_t\|$	$\|T\|$	$\|\Omega\|$	HSDDP $\widehat{UB}$	HSDDP Gap exact UB (%)	HSDDP Time best sol. (s)	MILP CPU time (s)	MILP Opt. gap (%)	MILP # variables	MILP # constraints
2	2	2^2	22,410.55	0.00	4.64	0.02	0.00	93.00	143.13
	4	2^4	39,617.15	0.09	497.54	0.35	0.00	375.67	538.50
	6	2^6	58,298.64	2.45	3860.15	14.91	0.00	1510.00	2,127.00
	8	2^8	77,003.57	3.39	6212.35	703.92	0.01	6278.00	8,633.00
	10	2^{10}	96,383.99	3.40	9023.68	9041.68	3.24	25053.00	34,333.00
5	2	5^2	22,202.27	0.00	41.75	0.07	0.00	149.00	249.00
	4	5^4	40,568.14	0.57	1370.17	6.84	0.00	2560.67	4,007.50
	6	5^6	59,773.26	1.09	4604.08	5400.02	6.60	64343.00	101,852.00
	8	5^8	79,088.43	-	7522.59	-	-	$\approx 10^7$	$\approx 10^7$
	10	5^{10}	93,844.87	-	9631.96	-	-	$\approx 10^8$	$\approx 10^9$
10	2	10^2	22,657.23	0.00	103.95	0.11	0.00	222.00	395.00
	4	10^4	40,797.96	0.97	1792.03	212.05	0.00	17944.33	31,704.83
	6	10^6	59,153.93	-	5553.89	-	-	$\approx 10^6$	$\approx 10^6$
	8	10^8	76,976.11	-	7851.30	-	-	$\approx 10^8$	$\approx 10^9$
	10	10^{10}	95,905.33	-	10123.11	-	-	$\approx 10^{10}$	$\approx 10^{11}$

"-" indicates the instances that are too large to build the mathematical model since it consumes too much memory.

Table 4 Comparison between HSDDP and the equivalent MILP.

$|T| = 2$, whereas the gap increases from 0 to 3.40 when the planning horizon increases from 2 to 10 periods with $|\Omega_t| = 2$. Finally, these experiments show that, contrarily to SDDP, the MILP approach is limited to a small number of scenarios since the MILP cannot be generated for 6 periods and 10 scenarios, and such a small number of scenarios does not give a good approximation of the stochastic demand.

6.6. The Impact of Component Substitution

To analyze the impact of component substitution on the production plan, Table 5 reports the values of different key performance indicators (KPIs), including total cost, rate of on-time delivery, inventory cost, backlog cost, end-of-horizon backlog cost, production cost, and component cost, for different numbers of alternative components (denoted by a), and substitution costs. We compare the performance of the expected demand model, the two-stage model, and the heuristic SDDP. The three approaches are simulated in a static-dynamic decision framework. That is, the expected demand and two-stage models are resolved in each iteration in a receding horizon fashion to update the production and consumption quantities based on the observed demands. Note that the cost components in percentage are provided in the Online Supplement. These results show that HSDDP tends to use more alternative components than 2S and Average. Therefore, we may conclude that the management of alternative components requires properly accounting for the uncertainties with a multistage model.

Method	a	a_{ij}	Total costs	On time (%)	# setups	Setup cost	Inventory cost	Backlog cost	End-of-horizon backlog cost	Production cost	Consumption cost
HSDDP	0	0.1	41,377.0	81.0	28.25	17,648.0	14,037.8	4,013.5	3,110.9	2,566.8	0.0
	2		40,672.2	80.9	25	17,090.7	13,828.3	4,028.7	3,114.4	2,569.0	41.1
	4		40,479.4	81.0	24	16,914.7	13,716.7	4,080.4	3,147.3	2,526.2	94.1
	6		40,718.6	80.8	20.5	17,157.0	13,841.4	3,925.2	3,024.3	2,585.4	185.2
	4	0	40,624.6	80.9	20.25	17,139.5	13,772.8	4,040.1	3,093.5	2,578.6	0.0
		0.1	40,479.4	81.0	24	16,914.7	13,716.7	4,080.4	3,147.3	2,526.2	94.1
		1	41,193.5	81.1	26.5	17,571.6	14,106.4	3,860.9	2,963.0	2,581.5	110.1
2S	0	0.1	42,000.4	75.4	27.75	17,606.7	12,347.6	4,796.1	4,759.8	2,490.1	0.0
	2		41,249.3	75.5	24	17,047.0	12,330.9	4,784.1	4,557.1	2,499.3	30.8
	4		40,804.5	75.7	23.75	16,887.0	12,224.0	4,762.3	4,390.0	2,487.3	53.8
	6		41,198.2	75.9	20	17,134.5	12,447.9	4,553.2	4,443.6	2,510.6	108.4
	4	0	41,223.0	75.8	20.25	17,152.0	12,475.7	4,572.7	4,509.6	2,512.9	0.0
		0.1	40,804.5	75.7	23.75	16,887.0	12,224.0	4,762.3	4,390.0	2,487.3	53.8
		1	41,726.5	75.8	27.5	17,586.7	12,467.5	4,562.2	4,604.5	2,489.4	16.1
Expected demand model	0	0.1	68,982.3	52.2	28	13,502.5	6,933.4	6,395.9	40,637.6	1,513.0	0.0
	2		66,650.9	52.6	24	12,787.5	6,997.0	6,304.2	39,016.6	1,525.6	20.0
	4		65,875.2	53.0	23.25	12,763.0	7,032.4	6,207.5	38,325.5	1,516.2	30.6
	6		65,046.0	53.1	19	12,730.0	7,040.8	6,170.9	37,442.7	1,549.8	111.8
	4	0	66,396.0	52.7	19.75	13,040.0	7,005.7	6,314.6	38,505.1	1,530.6	0.0
		0.1	65,875.2	53.0	23.25	12,763.0	7,032.4	6,207.5	38,325.5	1,516.2	30.6
		1	67,219.7	52.5	27.5	13,322.5	7,007.0	6,357.7	38,973.2	1,522.6	36.6

Table 5 Impact (in terms of various KPIs) of the number of substitutable components and of the substitution cost for the expected demand model, the two-stage model, and the heuristic SDDP in the static-dynamic decision framework

First, Table 5 shows that component substitution leads to lot consolidation. For instance, in the expected demand framework, the number of setups decreases from 28 to 19 when the number of possible alternates increases from 0 to 6. Second, component substitution leads to risk pooling. In the static-dynamic framework, the total inventory decreases by around 300 units when the number of possible alternates increases from 0 to 4, whereas the proportion of on-time delivery remains constant at 81%. In other words, pooling the risks allows maintaining the same service level with less inventory.

Table 5 shows that the level of inventory remains stable for 2S when the number of substitutions increases, but the percentage of on-time delivery increases (from 75.3 to 75.9 with 2S when the number of substitutable components increases from 0 to 6). The two-stage model does not plan to reassign the component's production to react to the observed demand. Nevertheless, when using the two-stage model in a static-dynamic decision frame-

work, the components get reassigned (thus, the service level is improved when the number of possible substitutions increases). However, as these reassignments are not foreseen, the model orders the large number of components required to hedge against demand uncertainty in the static framework (to ensure the production of each end-item separately), and thus the inventory level remains large. In other words, to pool the uncertainty of multiple end-items, the planning must be created with a multistage optimization model.

7. Conclusion

This paper investigates the performance of the SDDP algorithm for the capacitated multi-echelon lot-sizing problem (CMLSP) with stochastic demand and component substitution. The experiments show that SDDP performs well when the setups are fixed. Consequently, two strategies are proposed to compute the setups, namely PH, and a heuristic version of SDDP. The experiments show that the heuristic SDDP outperforms PH and the two-stage heuristic proposed in Thevenin et al. (2021). The proposed method can solve CMLSP with static-dynamic demand uncertainty for medium size planning horizons. In addition, contrarily to the scenario tree approach, once good approximations of the cost-to-go are built, SDDP can react quickly when new information is available.

As component substitution has the effect of aggregating the demand for components, it pools the risks associated with the uncertain demand and reduces the setups. Despite its practical relevance, only a few papers exist on lot sizing with component substitution under stochastic demand. Future research should investigate stochastic optimization for the dynamic-dynamic decision framework, as well as uncertainty in various parameters including lead times, yields, and production capacities.

Acknowledgments

This research was supported by the Institut de Valorisation des Données (IVADO) and by MITACS. The authors would also like to thank the CCIPL computing center and the region Pays de la Loire for the generous computing resource allocation. We are grateful to three anonymous reviewers for their valuable comments.

References

Absi N, Kedad-Sidhoum S, Dauzère-Pérès S (2011) Uncapacitated lot-sizing problem with production time windows, early productions, backlogs and lost sales. *International Journal of Production Research* 49(9):2551–2566.

Adulyasak Y, Cordeau JF, Jans R (2015) Benders decomposition for production routing under demand uncertainty. *Operations Research* 63(4):851–867.

Aloulou MA, Dolgui A, Kovalyov MY (2014) A bibliography of non-deterministic lot-sizing models. *International Journal of Production Research* 52(8):2293–2310.

Arkin E, Joneja D, Roundy R (1989) Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters* 8(2):61–66.

Balakrishnan A, Geunes J (2000) Requirements planning with substitutions: exploiting bill-of-materials flexibility in production planning. *Manufacturing & Service Operations Management* 2(2):166–185.

Begnaud J, Benjaafar S, Miller LA (2009) The multi-level lot sizing problem with flexible production sequences. *IIE Transactions* 41(8):702–715.

Birge JR, Louveaux F (2011) *Introduction to Stochastic Programming* (Springer, New York).

Bookbinder JH, Tan JY (1988) Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science* 34(9):1096–1108.

Brahimi N, Absi N, Dauzère-Pérès S, Nordli A (2017) Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research* 263(3):838–863.

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

Buschkühland L, Sahling F, Helber S, Tempelmeier H (2010) Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum* 32(2):231–261, ISSN 1436-6304.

Cecere L (2015) Does better forecasting improve inventory? Why I don't think so anymore. `https://www.forbes.com/sites/loracecere/2015/11/29/does-better-forecasting-improve-inventory-why-i-dont-think-so-anymore/amp/` (19/09/2020).

Chan AT, Ngai EW, Moon KK (2017) The effects of strategic and manufacturing flexibilities and supply chain agility on firm performance in the fashion industry. *European Journal of Operational Research* 259(2):486–499.

Crainic TG, Rei W, Hewitt M, Maggioni F (2016) Partial Benders decomposition strategies for two-stage stochastic integer programs. Tech. Rep. CIRRELT-2016-37, Université de Montréal, Canada.

Cranley R, Patterson TNL (1976) Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis* 13(6):904–914.

De Matos VL, Philpott AB, Finardi EC (2015) Improving the performance of stochastic dual dynamic programming. *Journal of Computational and Applied Mathematics* 290:196–208.

Dowson O, Philpott A, Mason A, Downward A (2019) A multi-stage stochastic optimization model of a pastoral dairy farm. *European Journal of Operational Research* 274(3):1077–1089.

Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. *Operations Research* 26(6):992–1009.

Firoozi M, Babai MZ, Klibi W, Ducq Y (2020) Distribution planning for multi-echelon networks considering multiple sourcing and lateral transshipments. *International Journal of Production Research* 58(7):1968–1986.

Freeman NK, Narayanan A, Keskin BB (2020) Optimal use of downward substitution in a manufacturing operation subject to uncertainty. *Omega* 102372.

Gallego G, Katircioglu K, Ramachandran B (2006) Semiconductor inventory management with multiple grade parts and downgrading. *Production Planning & Control* 17(7):689–700.

Geunes J (2003) Solving large-scale requirements planning problems with component substitution options. *Computers & Industrial Engineering* 44(3):475–491.

Grubbström RW, Wang Z (2003) A stochastic model of multi-level/multi-stage capacity-constrained production–inventory systems. *International Journal of Production Economics* 81:483–494.

Han G, Dong M, Liu S (2014) Yield and allocation management in a continuous make-to-stock system with demand upgrade substitution. *International Journal of Production Economics* 156:124–131.

Harris FW (1990) How many parts to make at once. *Operations Research* 38(6):947–950.

Haugen KK, Løkketangen A, Woodruff DL (2001) Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operational Research* 132(1):116–122.

Hsu A, Bassok Y (1999) Random yield and random demand in a production system with downward substitution. *Operations Research* 47(2):277–290.

Huang D, Zhou H, Zhao QH (2011) A competitive multiple-product newsboy problem with partial product substitution. *Omega* 39(3):302–312.

Jans R, Degraeve Z (2008) Modeling industrial lot sizing problems: a review. *International Journal of Production Research* 46(6):1619–1643.

Karimi B, Ghomi SF, Wilson J (2003) The capacitated lot sizing problem: a review of models and algorithms. *Omega* 31(5):365–378.

Lang JC, Domschke W (2010) Efficient reformulations for dynamic lot-sizing problems with product substitution. *OR Spectrum* 32(2):263–291.

L'Ecuyer P, Munger D (2016) Algorithm 958: Lattice builder: A general software tool for constructing rank-1 lattice rules. *ACM Transactions on Mathematical Software (TOMS)* 42(2):15.

Lohmann T, Hering AS, Rebennack S (2016) Spatio-temporal hydro forecasting of multireservoir inflows for hydro-thermal scheduling. *European Journal of Operational Research* 255(1):243–258.

Magnanti TL, Wong RT (1981) Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research* 29(3):464–484.

Metropolis N, Ulam S (1949) The Monte Carlo method. *Journal of the American Statistical Association* 44(247):335–341.

Pacaud F, Carpentier P, Chancelier JP, De Lara M (2018) Stochastic optimal control of a domestic microgrid equipped with solar panel and battery. *arXiv preprint arXiv:1801.06479* .

Papadakos N (2008) Practical enhancements to the Magnanti–Wong method. *Operations Research Letters* 36(4):444–449.

Parpas P, Ustun B, Webster M, Tran QK (2015) Importance sampling in stochastic programming: A Markov chain Monte Carlo approach. *INFORMS Journal on Computing* 27(2):358–377.

Pereira MV, Pinto LM (1991) Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming* 52(1-3):359–375.

Piperagkas GS, Konstantaras I, Skouri K, Parsopoulos KE (2012) Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics. *Computers & Operations Research* 39(7):1555–1565.

Quezada F, Gicquel C, Kedad-Sidhoum S (2019) Stochastic dual dynamic integer programming for a multi-echelon lot-sizing problem with remanufacturing and lost sales. *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 1254–1259 (IEEE).

Quezada F, Gicquel C, Kedad-Sidhoum S (2021) Combining polyhedral approaches and stochastic dual dynamic integer programming for solving the uncapacitated lot-sizing problem under uncertainty. *INFORMS Journal on Computing* (forthcoming).

Quezada F, Gicquel C, Kedad-Sidhoum S, Vu DQ (2020) A multi-stage stochastic integer programming approach for a multi-echelon lot-sizing problem with returns and lost sales. *Computers & Operations Research* 116:104865.

Rao US, Swaminathan JM, Zhang J (2004) Multi-product inventory planning with downward substitution, stochastic demand and setup costs. *IIE Transactions* 36(1):59–71.

Rockafellar RT, Wets RJB (1991) Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* 16(1):119–147.

Santoso T, Ahmed S, Goetschalckx M, Shapiro A (2005) A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research* 167(1):96–115.

Shapiro A (2011) Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research* 209(1):63–72.

Shin H, Park S, Lee E, Benton W (2015) A classification of the literature on the planning of substitutable products. *European Journal of Operational Research* 246(3):686–699.

Soares MP, Street A, Valladão DM (2017a) On the solution variability reduction of stochastic dual dynamic programming applied to energy planning. *European Journal of Operational Research* 258(2):743–760.

Soares MP, Street A, Valladão DM (2017b) On the solution variability reduction of stochastic dual dynamic programming applied to energy planning. *European Journal of Operational Research* 258(2):743–760.

Sox CR (1997) Dynamic lot sizing with random demand and non-stationary costs. *Operations Research Letters* 20(4):155–164.

Sreedevi R, Saranga H (2017) Uncertainty and supply chain risk: The moderating role of supply chain flexibility in risk mitigation. *International Journal of Production Economics* 193:332–342.

Tarim SA, Dogru MK, Özen U, Rossi R (2011) An efficient computational method for a stochastic dynamic lot-sizing problem under service-level constraints. *European Journal of Operational Research* 215(3):563–571.

Tarim SA, Kingsman BG (2006) Modelling and computing (Rn, Sn) policies for inventory systems with non-stationary stochastic demand. *European Journal of Operational Research* 174(1):581–599.

Tempelmeier H (2013) Stochastic lot sizing problems. *Handbook of Stochastic Models and Analysis of Manufacturing System Operations*, 313–344 (Springer).

Tempelmeier H, Derstroff M (1996) A Lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science* 42(5):738–757.

Tempelmeier H, Hilger T (2015) Linear programming models for a stochastic dynamic capacitated lot sizing problem. *Computers & Operations Research* 59:119–125.

Thevenin S, Adulyasak Y, Cordeau JF (2021) Material requirements planning under demand uncertainty using stochastic optimization. *Production and Operations Management* 30(2):475–493.

Tunc H, Kilic OA, Tarim SA, Rossi R (2018) An extended mixed-integer programming formulation and dynamic cut generation approach for the stochastic lot-sizing problem. *INFORMS Journal on Computing* 30(3):492–506.

Valladão D, Silva T, Poggi M (2019) Time-consistent risk-constrained dynamic portfolio optimization with transactional costs and time-dependent returns. *Annals of Operations Research* 282(1-2):379–405.

Wei M, Qi M, Wu T, Zhang C (2019) Distance and matching-induced search algorithm for the multi-level lot-sizing problem with substitutable bill of materials. *European Journal of Operational Research* 277(2):521–541.

Wu T, Akartunalı K, Jans R, Liang Z (2017) Progressive selection method for the coupled lot-sizing and cutting-stock problem. *INFORMS Journal on Computing* 29(3):523–543.

Yaman H (2009) Polyhedral analysis for the two-item uncapacitated lot-sizing problem with one-way substitution. *Discrete Applied Mathematics* 157(14):3133–3151.