Combining polyhedral approaches and stochastic dual dynamic integer programming for solving the uncapacitated lot-sizing problem under uncertainty

Franco Quezada, Céline Gicquel, Safia Kedad-Sidhoum

▶ To cite this version:

Franco Quezada, Céline Gicquel, Safia Kedad-Sidhoum. Combining polyhedral approaches and stochastic dual dynamic integer programming for solving the uncapacitated lot-sizing problem under uncertainty. INFORMS Journal on Computing, 2022, 34 (2), pp.1024-1041. 10.1287/ijoc.2021.1118 . hal-03606367

HAL Id: hal-03606367

https://hal.science/hal-03606367

Submitted on 11 Mar 2022

Combining polyhedral approaches and stochastic dual dynamic integer programming for solving the uncapacitated lot-sizing problem under uncertainty

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

[1] Sorbonne Université, LIP6, Paris, France
[2] Universidad de Santiago de Chile, LDSPS, Santiago, Chile
[3] Université Paris Saclay, LISN, Orsay, France
[4] CNAM, CEDRIC,Paris, France

Abstract: We study the uncapacitated lot-sizing problem with uncertain demand and costs. The problem is modeled as a multi-stage stochastic mixed-integer linear program in which the evolution of the uncertain parameters is represented by a scenario tree. To solve this problem, we propose a new extension of the stochastic dual dynamic integer programming algorithm (SDDiP). This extension aims at being more computationally efficient in the management of the expected cost-to-go functions involved in the model, in particular by reducing their number and by exploiting the current knowledge on the polyhedral structure of the stochastic uncapacitated lot-sizing problem. The algorithm is based on a partial decomposition of the problem into a set of stochastic sub-problems, each one involving a subset of nodes forming a sub-tree of the initial scenario tree. We then introduce a cutting-plane generation procedure that iteratively strengthens the linear relaxation of these sub-problems and enables the generation of additional strengthened Benders' cut, which improves the convergence of the method. We carry out extensive computational experiments on randomly generated large-size instances. Our numerical results show that the proposed algorithm significantly outperforms the SDDiP algorithm at providing good-quality solutions within the computation time limit.

Keywords: Multi-stage stochastic programming, stochastic lot-sizing, stochastic dual dynamic integer programming, node aggregation, valid inequalities, partial decomposition.

1 Introduction

The single-item deterministic uncapacitated lot-sizing problem (ULS) is a production planning problem first introduced by Wagner and Whitin (1958). It considers a single type of item and aims at determining the quantity to be produced in each time period in order to meet demand over a finite discrete-time planning horizon. Producing a positive amount in a period incurs a fixed cost, called setup cost, together with a production cost per unit produced and an inventory holding cost per unit held in stock between two consecutive periods. The objective is to build a production plan such that the customers' demand is met in each time period and the total costs, i.e. the sum of setup, production, and inventory holding costs over the whole planning horizon, are minimized. This fundamental problem naturally appears as an embedded sub-problem in many practical production planning problems. Solving it efficiently is thus essential to develop algorithms capable of dealing with real-world problems.

As such, the deterministic ULS is known to be solvable in strongly polynomial time. A simple dynamic programming algorithm is proposed by Wagner and Whitin (1958). It is based on the zero-inventory-ordering property, i.e. production is undertaken in a period only if the entering inventory level drops to zero, and runs in $\mathcal{O}(T^2)$ time, where T is the number of time periods. This time complexity was later improved to $\mathcal{O}(T \log T)$ by Aggarwal and Park (1993) and Wagelmans et al. (1992). We refer the reader to Brahimi et al. (2017) for an updated and comprehensive survey on the single-item dynamic lot-sizing problem.

However, in many applications, assuming deterministic input data (demand and costs) is not realistic. Examples of real-world lot-sizing problems with uncertain input parameters can be found among others in Camargo et al. (2014) for the spinning industry, Hu and Hu (2016) for a manufacturing company producing braking equipment, Ghamari and Sahebi (2017) for a chemical-petrochemical case study, Kilic et al. (2018) for a remanufacturing system, Macedo et al. (2016) for a hybrid manufacturing/remanufacturing system and Moreno et al. (2018) for humanitarian logistics.

In the present paper, we thus investigate a stochastic extension of the ULS, denoted SULS in what follows, in which the problem parameters are subject to uncertainty. We consider a multi-stage decision process corresponding to the case where the value of the uncertain parameters unfolds gradually following a discrete-time stochastic process and the production decisions can be made progressively as more and more information on the demand and cost realizations are collected. In order to address this problem, we use a multi-stage stochastic mixed-integer linear programming approach. We assume that the underlying stochastic input process has a finite probability space so that the information on the evolution of the uncertain parameters can be represented by a discrete scenario tree. Moreover, throughout this work, we rely on the commonly used assumption that the scenario tree is stage-wise independent.

1.1 Related literature

Scarf (1959) found that, when all cost parameters are deterministic and only the demand is subject to uncertainty, the optimal solution of SULS can be obtained by following a (s, S) inventory management policy. However, Halman et al. (2009) showed that a special case of SULS in which the production and inventory holding costs are stochastic, the setup costs are set to zero and the uncertain demand can take only two possible values in each period is NP-Hard. It is thus unlikely to find polynomial algorithms in the number of time periods for the problem, unless $P = NP$. Guan and Miller (2008) developed a dynamic programming algorithm for solving the SULS, which is polynomial in the number of nodes of the scenario tree. However, this number increases exponentially fast with the number of time periods so that using their algorithm to solve problems with a medium to large size planning horizon might lead to numerical difficulties. Moreover, this algorithm is based on some specific properties of the optimal solutions of the SULS. As a consequence, its direct extension to more general lot-sizing problems whose optimal solutions do not display these properties is not straightforward. This is why other solution approaches based on mixed-integer linear programming or nested Benders' decomposition have also been explored. In what follows, we provide an overview of the related literature.

Using a scenario tree to represent the evolution of the uncertain parameters namely leads to the formulation of a Mixed-Integer Linear Program (MILP) which can be solved using mathematical programming solvers. Consequently, several works focused on the polyhedral study of this MILP in order to strengthen its linear relaxation and improve the computational efficiency of the branch-and-cut algorithms embedded in MILP solvers. Valid inequalities can be found in Guan et al. (2006b), Di Summa and Wolsey (2008) and Guan et al. (2009). In particular, Guan et al. (2009) proposed a general method for generating cutting planes for multi-stage stochastic mixed-integer linear programs based on combining valid inequalities previously known for the deterministic variant of the corresponding problem and applied it on the SULS. Their numerical results showed that a branch-and-cut algorithm based on these new inequalities is more effective at solving instances on medium-size scenarios than a stand-alone mathematical programming solver. Some extended formulations have also been studied. An extended formulation using variable disaggregation was proposed by Ahmed et al. (2003). More recently, Zhao and Guan (2014) developed an extended formulation that provides integral solutions for the general SULS. However, the size of this formulation grows exponentially fast with the number of time periods, making its use computationally unpractical for solving instances defined on large-size scenario trees.

In general, solution approaches based on strengthening MILP formulations do not scale up well with the size of the scenario tree. They namely entail solving very large-scale (mixed-integer) linear programs, with millions of variables and constraints, which leads to memory issues and/or prohibitive computation times in practice. Decomposition methods, such as the nested Benders' decomposition algorithm, are thus an attractive alternative to tackle instances with large-size scenario trees. In particular, the Stochastic Dual Dynamic Programming (SDDP) approach proposed by Pereira and Pinto (1991) has been widely used to solve large-size multi-stage stochastic linear programs. This approach relies on a dynamic programming formulation of the stochastic problem and leads to a decomposition of the overall problem into a series of small deterministic sub-problems. Each of these problems focuses on making decisions for a small subset of nodes belonging to the same scenario and the same decision stage, taking into account not only the current cost of these decisions but also their future cost which is represented by an expected cost-to-go function. In a linear setting, the expected cost-to-go functions are convex and piecewise linear and can thus be under-approximated through a set of supporting hyperplanes. The SDDP algorithm builds such an approximation by iteratively adding Benders' cuts to each sub-problem and converges to an optimal solution in a finite number of iterations.

Recently, Zou et al. (2019) proposed a new extension of the SDDP algorithm, called the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm, capable of solving multi-stage stochastic mixed-integer linear programs in which the state variables, i.e. the variables linking the nodes to one another, are restricted to be binary. One of their main contributions was to introduce a new class of cutting planes, called Lagrangian cuts, which satisfies the validity, tightness and finiteness conditions ensuring the convergence of the algorithm to optimality. For problems in which the state variables are not binary but continuous, the authors propose to introduce auxiliary binary variables in order to make a binary approximation of the state variables. However, for large size scenario trees, this approximation might be computationally inefficient and leads to large optimality gaps as shown e.g. by the numerical results presented by Quezada et al. (2019).

1.2 Contributions

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

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

- In the SDDiP algorithm proposed by Zou et al. (2019), three classes of cuts are used to under-approximate the expected cost-to-go functions, namely the *Integer optimality cuts*, the *Lagrangian cuts* and the *strengthened Benders' cuts*. In particular, the strengthened Benders' cuts generated at a given node of the scenario tree are linear inequalities for which part of the coefficients are obtained by solving the linear relaxation of the sub-problems corresponding to its children nodes and by recording the dual values of the constraints linking the sub-problems to one another. In addition to the strengthened Benders' cuts generated using the initial linear relaxation of the children sub-problems, we propose to generate additional strengthened Benders' cuts using an improved linear relaxation of these sub-problems. We thus introduce a cutting-plane generation procedure that takes advantage of previously published results on the polyhedral structure of the SULS to iteratively strengthen the relaxation of these sub-problems. Our computational results show that the joint use of the strengthened Benders' cuts obtained using the initial relaxation and the ones obtained using an improved relaxation of the sub-problems leads to a significant improvement of the computational efficiency of the SDDiP algorithm.

- Finally, as proposed e.g. by Hjelmeland et al. (2018) and Quezada et al. (2019), before actually running the SDDiP algorithm as defined by Zou et al. (2019), we introduce an initial phase in which only strengthened Benders' cuts are generated on a dynamic programming formulation using the initial continuous state variables. This strategy relies on the idea that, even if the obtained cuts do not satisfy the tightness condition necessary to theoretically ensure the global convergence of the SDDiP algorithm, they enable to build a first under-approximation of the expected cost-to-go functions with a reduced computational effort as each sub-problem involves a limited number of binary variables. This initial under-approximation is then further refined in a second phase based on a dynamic programming formulation using a binary approximation of the state variables.

Note that the use of a sub-tree decomposition (or node aggregation) has been explored by several authors in the context of stochastic dynamic programming. Cerisola and Ramos (2000) studied a multistage stochastic linear problem for hydro-power generation scheduling. They proposed to decompose the scenario tree into connected sub-trees and presented several node aggregation protocols to generate these sub-trees. Later, Cristobal et al. (2009) considered multi-stage stochastic mixed-integer programs and developed a stochastic dynamic programming approach in which the original scenario tree is decomposed into sub-trees. However, their algorithm significantly differs from the SDDiP algorithm as it builds an upper envelope of the expected cost-to-go functions and only provides a heuristic solution whereas the SDDiP algorithm under-approximates the expected cost-to-go functions and is capable of providing both a lower and an upper bound of the optimal value. Note that Aldasoro et al. (2015) and Escudero et al. (2018) recently presented an extension on the algorithm proposed by Cristobal et al. (2009) aiming at exploiting parallel computing to further reduce the computation time.

The contributions of this work are threefold following these three main ideas. First, we propose a new extension of the SDDiP algorithm in which a partial decomposition of the scenario tree is used to generate sub-problems. To the best of our knowledge, this is the first time such an extension is studied in the context of the SDDiP algorithm. Second, we propose to take advantage of the tree structure of the sub-problems to exploit results on the polyhedral structure of the SULS and generate additional strengthened Benders' cuts to approximate the expected cost-to-go functions. Third, we carry out extensive computational experiments to assess the performance of the proposed algorithm at solving the SULS. We thus compare its performance with the one of a stand-alone mathematical solver and the one of the SDDiP algorithm proposed by Zou et al. (2019). The results show that this new algorithm outperforms ILOG-CPLEX and the SDDiP algorithm at solving large-size instances of the SULS.

The remaining part of this paper is organized as follows. Section 2 introduces a deterministic equivalent mixed-integer linear programming formulation and a stochastic dynamic programming formulation of the SULS. Section 3 presents the extension of the SDDiP algorithm in which a partial decomposition of the scenario tree is used. Section 4 then describes two further enhancements of the algorithm. Finally, the results of our computational experiments are reported in Section 5. Conclusions and directions for further works are discussed in Section 6.

2 Mathematical formulations

We aim at planning the production of a single type of item on a single resource over a planning horizon $\mathcal{T} = \{1, ..., T\}$ of T periods under uncertain demand and costs. We consider a decision process involving Σ decision stages and denote by $\mathcal{S} = \{1, ..., \Sigma\}$ the set of stages. A stage may correspond to one or several consecutive planning periods. This is of particular interest in the context of lot-sizing problems as the time discretization used by the decision-makers to plan production activities is indeed usually finer than the one used to update the demand and cost forecasts and readjust the production plan. A planning period may thus typically correspond to an 8-hours shift or a day whereas a stage may correspond to a week or a month. Let $\mathcal{T}^\sigma$ be the set of time periods belonging to stage $\sigma \in \mathcal{S}$. Note that the sets $\{\mathcal{T}^\sigma, \sigma \in \mathcal{S}\}$ form a partition of $\mathcal{T}$.

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

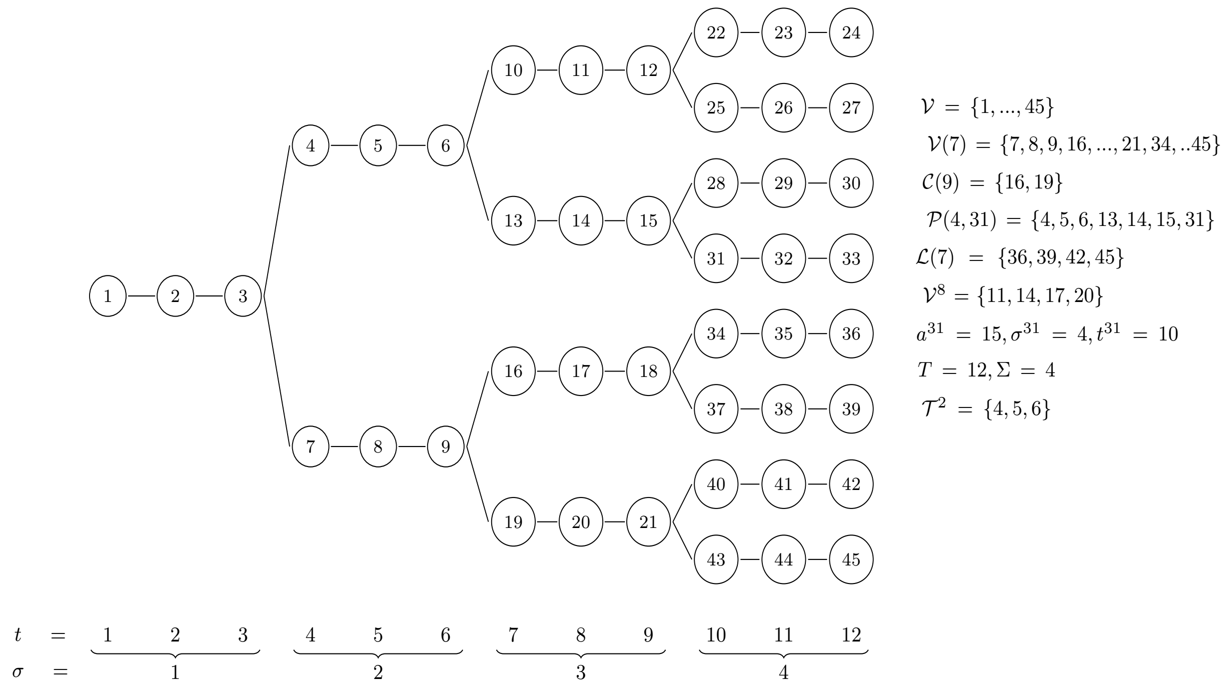
$\mathcal{V} = \{1, ..., 45\}$
$\mathcal{V}(7) = \{7, 8, 9, 16, ..., 21, 34, ..45\}$
$\mathcal{C}(9) = \{16, 19\}$
$\mathcal{P}(4, 31) = \{4, 5, 6, 13, 14, 15, 31\}$
$\mathcal{L}(7) = \{36, 39, 42, 45\}$
$\mathcal{V}^8 = \{11, 14, 17, 20\}$
$a^{31} = 15, \sigma^{31} = 4, t^{31} = 10$
$T = 12, \Sigma = 4$
$\mathcal{T}^2 = \{4, 5, 6\}$
t = 1 2 3 4 5 6 7 8 9 10 11 12
σ = 1 2 3 4

Figure 1: Scenario tree structure

The stochastic input parameters are defined as follows:

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

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

2.1 Extensive MILP formulation

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

- x^n: quantity produced at node $n \in \mathcal{V}$,
- $y^n = 1$ if a setup for production is carried out at node $n \in \mathcal{V}$, $y^n = 0$ otherwise,
- s^n: inventory level at node $n \in \mathcal{V}$,

This leads to the following MILP formulation:

$$\min \sum_{n \in \mathcal{V}} \rho^n (f^n y^n + h^n s^n + g^n x^n) \tag{1}$$

$$x^n \leq M^n y^n \qquad \forall n \in \mathcal{V} \tag{2}$$

$$s^n + d^n = x^n + s^{a^n} \qquad \forall n \in \mathcal{V} \tag{3}$$

$$x^n, s^n \geq 0, y^n \in \{0,1\} \qquad \forall n \in \mathcal{V} \tag{4}$$

The objective function (1) aims at minimizing the expected total setup, inventory holding and production costs over all nodes of the scenario tree. Constraints (2) link the production quantity variables to the setup variables. Note that the value of constant M^n can be set by using an upper bound on the quantity to be processed at node n, usually defined as the maximum future demand as seen from node n, i.e. $M^n = \max_{\ell \in \mathcal{L}(n)} d^{n\ell}$, where $d^{n\ell} = \sum_{m \in \mathcal{P}(n,\ell)} d^m$. Constraints (3) are the inventory balance constraints. Constraints (4) provide the decision variables domain.

Several MILP formulation strengthening techniques have been investigated for this problem: see Appendix A in the online supplement of this work for a detailed presentation of the valid inequalities proposed by Guan et al. (2009).

Problem (1)-(4) can thus be solved using MILP solvers. Nonetheless, the size of the formulation grows exponentially fast with the number of nodes $|\mathcal{V}|$ in the scenario tree, leading to prohibitive computation times in practice. We thus investigate in what follows a dynamic programming formulation which serves as a basis to develop a decomposition algorithm to solve the problem.

2.2 Dynamic programming formulation

An alternative to the extensive formulation of the SULS discussed above is a dynamic programming formulation involving nested expected cost-to-go functions. This approach decomposes the original problem into a series of smaller sub-problems linked together by dynamic programming equations. When applying the SDDiP algorithm proposed by Zou et al. (2019) on the SULS, a full decomposition of the problem is carried out, resulting in a large number of small sub-problems. Each of these sub-problems is a small deterministic lot-sizing problem aiming at planning production on a subset of nodes corresponding to a single scenario and a single decision stage. In what follows, we propose to consider a partial decomposition of the problem resulting in a smaller number of larger sub-problems, each one being a stochastic lot-sizing problem aiming at planning production on a sub-tree of the original scenario tree.

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

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

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

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

$$Q^\eta(s^{a^\eta}) = \min \sum_{n \in \mathcal{W}^\eta} \rho^n (f^n y^n + h^n s^n + g^n x^n) + \sum_{\ell \in \mathfrak{L}(\eta)} \sum_{m \in \mathcal{C}(\ell)} Q^m(s^\ell) \tag{5}$$

$$x^n \le M^n y^n \qquad \forall n \in \mathcal{W}^\eta \tag{6}$$

$$s^n + d^n = s^{a^n} + x^n \qquad \forall n \in \mathcal{W}^\eta \tag{7}$$

$$x^n, s^n \ge 0, y^n \in \{0,1\} \qquad \forall n \in \mathcal{W}^\eta \tag{8}$$

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

In (5), $Q^\eta(s^{a^\eta})$ denotes the optimal value of sub-problem P^η as a function of the entering stock level s^{a^η} and $Q^m(s^\ell)$ the optimal value of sub-problem P^m as a function of the entering stock level s^ℓ. The expected cost-to-go function at node $\ell \in \mathfrak{L}(\eta)$ is defined as the expected value of $Q^m(\cdot)$ over all the children of ℓ in the initial scenario tree $\mathcal{V}$, i.e. over all $m \in \mathcal{C}(\ell)$, which gives $\mathcal{Q}^\ell(\cdot) = \sum_{m \in \mathcal{C}(\ell)} Q^m(\cdot)$. The expected future costs of the decisions made in $\mathcal{W}^\eta$ are thus computed as the sum, over all nodes $\ell \in \mathfrak{L}(\eta)$, of $\mathcal{Q}^\ell(s^\ell)$.

We note that in case of $\mathcal{G} \equiv \mathcal{S}$, i.e. in case each macro-stage corresponds to a single initial decision stage, each sub-tree $\mathcal{W}^\eta$ reduces to a set of nodes belonging to a single deterministic scenario involving T^{σ^η} periods and we obtain a decomposition similar to the one used by Zou et al. (2019).

3 Sub-tree-based SDDiP algorithm

We now present the proposed extension of the SDDiP algorithm applied to the SULS. This extension relies on the dynamic programming formulation (5)-(8) and corresponds to a partial decomposition of the original problem into a set of smaller problems, each one expressed on a sub-tree of the scenario tree. As described in the SDDiP proposed by Zou et al. (2019), the main idea is to solve a sequence of sub-problems in which the expected cost-to-go functions $\mathcal{Q}^\ell(\cdot), \ell \in \mathfrak{L}(\eta)$, of each sub-problem $P^\eta, \eta \in \mho$, are iteratively approximated by a piece-wise linear function. However, whereas the original SDDiP considers a large number of small deterministic sub-problems, we use a smaller number of medium-size stochastic sub-problems.

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

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

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

3.1 Sub-problem reformulation

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

Namely, in the SULS, the state variables are the continuous inventory variables s^n. As the SDDiP developed by Zou et al. (2019) requires the state variables to be binary, we first carry out a binary approximation of the state variables before applying the algorithm to our problem. This binary approximation is obtained by replacing the continuous variable s^n by a set of binary variables $u^{n,\beta}$ such that $s^n = \sum_{\beta\in\mathcal{B}} 2^\beta u^{n,\beta}$, where $\mathcal{B} = \{1, \ldots, B\}$. We have $u^{n,\beta} = 1$ if coefficient 2^β is used to compute the value of s^n and $u^{n,\beta} = 0$ otherwise. We note however that this binary approximation is not needed for all inventory variables, but only for those coupling the sub-problems $P^\gamma(\cdot,\cdot)$, to one another. Thus, in sub-problem $P^\gamma(s^m, \mathcal{X}^{\gamma,r})$, we use a binary approximation for the entering stock s^m at root node $\xi^{\gamma,r}$ and for the leaving stock s^ℓ at each leaf node $\ell \in \mathfrak{L}(\gamma, r)$. Note that for the instances considered in our numerical experiments, introducing this binary approximation of the inventory variables will not lead to a sub-optimal solution. Namely, the randomly generated demand vectors only comprise integer components and the optimal leaving inventory at each node is known to take an integer value in this case: see Guan and Miller (2008).

Then, as indicated by Zou et al. (2019), we introduce local copies of the binary state variables relative to root node $\xi^{\gamma,r}$. More precisely, $\hat{u}^{\xi^{\gamma,r},\beta}$ is an auxiliary continuous decision variable representing the value of the state variable $u^{m,\beta}$ at the parent node m. It is thus a local copy in problem $P^\gamma(s^m, \mathcal{X}^{\gamma,r})$ of the state variable $u^{m,\beta}$, the value of which is considered as a given input parameter for this problem.

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

$$Q^{\gamma,r}(u^m) = \min \sum_{n \in \mathcal{X}^{\gamma,r}} \rho^n (f^n y^n + h^n s^n + g^n x^n) + \sum_{\ell \in \mathfrak{L}(\gamma,r)} \mathcal{Q}^{\gamma}(u^{\ell}) \tag{9}$$

$$\begin{aligned}
& x^n \leq M^n y^n && \forall n \in \mathcal{X}^{\gamma,r} && (10)\\
& s^{\xi^{\gamma,r}} + d^{\xi^{\gamma,r}} = \sum_{\beta \in \mathcal{B}} 2^{\beta} \hat{u}^{\xi^{\gamma,r},\beta} + x^{\xi^{\gamma,r}} && && (11)\\
& \hat{u}^{\xi^{\gamma,r},\beta} = u^{m,\beta} && \forall \beta \in \mathcal{B} && (12)\\
& s^n + d^n = s^{a^n} + x^n && \forall n \in \mathcal{X}^{\gamma,r} \setminus \{\xi^{\gamma,r}\} && (13)\\
& s^{\ell} = \textstyle\sum_{\beta \in \mathcal{B}} 2^{\beta} u^{\ell,\beta} && \forall \ell \in \mathfrak{L}(\gamma, r) && (14)\\
& \hat{u}^{\xi^{\gamma,r},\beta} \in [0,1] && \forall \beta \in \mathcal{B} && (15)\\
& u^{\ell,\beta} \in \{0,1\} && \forall \ell \in \mathfrak{L}(\gamma,r), \forall \beta \in \mathcal{B} && (16)\\
& x^n, s^n \geq 0, y^n \in \{0,1\} && \forall n \in \mathcal{X}^{\gamma,r} && (17)
\end{aligned}$$

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

In this reformulation, Constraint (11) corresponds to the inventory balance at node $\xi^{\gamma,r}$ in which the entering stock level s^m is computed using the auxiliary variables $\hat{u}^{\xi^{\gamma,r}\beta}$. Equalities (12) are copy constraints ensuring that the value of each auxiliary variable $\hat{u}^{\xi^{\gamma,r},\beta}$ is equal to the value of the corresponding state variable $u^{m,\beta}$ imposed by the parent node m. Constraints (13) ensure the inventory balance at each node of sub-tree $\mathcal{X}^{\gamma,r}$ except the root node $\xi^{\gamma,r}$. Constraints (14) define, for each leaf node $\ell \in \mathfrak{L}(\gamma, r)$, the value of the binary variables $u^{\ell,\beta}$, which will be used to compute the future expected costs as $\mathcal{Q}^{\gamma}(u^{\ell}) = \sum_{r' \in \mathcal{R}^{\gamma+1}} Q^{\gamma+1,r'}(u^{\ell})$. Note that, although variables $\hat{u}^{\xi^{\gamma,r},\beta}$ and constraints (12) are redundant for sub-problem $P^{\gamma}(u^m, \mathcal{X}^{\gamma,r})$, they will play a key role in the generation of the Lagrangian and strengthened Benders' cuts used in the SDDiP algorithm to approximate the expected cost-to-functions.

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

3.2 Sampling step

In the sampling step, a subset of K scenarios, i.e. a set of paths going from the root node to a leaf node, are randomly selected. Let $\Omega_i = \{\omega_i^1, \ldots, \omega_i^k, \ldots, \omega_i^K\}$ be the set of sampled scenarios, ω_i^k be the set of nodes belonging to scenario k at iteration i and $r_i^{k,\gamma}$ be the index of the realization in $\mathcal{R}^{\gamma}$ containing the values of the uncertain parameters in scenario ω_i^k at macro-stage γ.

3.3 Forward step

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

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

$$\psi_i^\gamma(u^\ell) = \min\{\theta^{\gamma\ell} : \theta^{\gamma\ell} \geq \sum_{r \in \mathcal{R}^{\gamma+1}} \nu_j^{\gamma+1,r} + \pi_j^{\gamma+1,r} u^\ell \quad \forall j \in \{1, \dots, i-1\}\} \quad (18)$$

where $\nu_j^{\gamma+1,r}$ and $\pi_j^{\gamma+1,r}$ are the coefficients of the cut generated at iteration $j < i$ by considering realization $r \in \mathcal{R}^{\gamma+1}$. This leads to the following sub-problem $\hat{P}_i^\gamma(u_i^m, \psi_i^\gamma, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$:

$$\hat{Q}_i^{\gamma, r_i^{k,\gamma}}(u_i^m) = \min \sum_{n \in \mathcal{X}^{\gamma, r_i^{k,\gamma}}} \rho^n(f^n y^n + h^n s^n + g^n x^n) + \sum_{\ell \in \mathfrak{L}(\gamma, r_i^{k,\gamma})} \theta^{\gamma\ell} \quad (19)$$

$$\theta^{\gamma\ell} \geq \sum_{r \in \mathcal{R}^{\gamma+1}} \nu_j^{\gamma+1,r} + \pi_j^{\gamma+1,r} u^\ell \qquad \forall j = 1, \dots, i-1, \quad \forall \ell \in \mathfrak{L}(\gamma, r_i^{k,\gamma}) \quad (20)$$

Constraints (10) − (17) for $r = r_i^{k,\gamma}$

The forward step at iteration i ends when sub-problem $\hat{P}_i^\gamma(u_i^m, \psi_i^\gamma, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$ has been solved for all sampled scenarios and all macro-stages. Its output is a feasible production plan for all nodes belonging to a sampled scenario. In particular, it provides a value u_i^m for all state variables u^m such that $m \in \omega_i^k \cap \mathcal{V}^{t'(\gamma)}, \gamma \in \mathcal{G}$, $k = 1, \dots, K$. These values will be used in the backward step to generate additional cuts and improve the approximation of the expected cost-to-go functions $\mathcal{Q}^\gamma(\cdot), \gamma \in \mathcal{G}$.

3.4 Backward step

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

This step starts from macro-stage Γ and goes back to macro-stage 1. Note that the sub-problems relative to macro-stage Γ do not have any expected future costs, therefore $\psi_i^\Gamma \equiv 0$, for all i. At each macro-stage $\gamma = \Gamma - 1, \dots, 1$, the updating of the approximation of $\mathcal{Q}^\gamma(\cdot)$ is carried out as follows. For each scenario $k = 1, \dots, K$, each node $m \in \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$ and each realization $r \in \mathcal{R}^{\gamma+1}$, we solve a suitable relaxation of $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ and collect the cut coefficients $\{\nu_i^{\gamma+1,r}, \pi_i^{\gamma+1,r}\}$. These coefficients are then used to generate a new linear inequality of type (18) to be added to the current approximation of $\mathcal{Q}^\gamma(\cdot)$. The backward step continues iteratively until the approximation of the expected cost-to-go function at macro-stage $\gamma = 1$ is updated. Since ψ_{i+1}^1 is an under-approximation of the expected cost-to-go function $\mathcal{Q}^1(\cdot)$, the optimal value $\hat{Q}_{i+1}^{1,1}(0)$ of problem $\hat{P}_i^1(0, \psi_{i+1}^1, \mathcal{X}^{1,1})$, provides a lower bound of the optimal value of the stochastic problem.

3.4.1 Cut families

We now briefly recall the three types of cutting planes used in Zou et al. (2019) to improve the approximation of the expected cost-to-go functions during the backward step. Let us consider a macro-stage γ, a scenario index k and the node $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$. Let u_i^m be the value of the state variables u^m in the solution of problem $\hat{P}_i^\gamma(u_i^m, \psi_i^\gamma, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$ solved in the forward step of iteration i. The three following cuts can be added to compute the approximation ψ_i^γ of $\mathcal{Q}^\gamma(\cdot)$.

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

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

Lagrangian cut: We consider, for each $r \in \mathcal{R}^{\gamma+1}$, the Lagrangian relaxation of problem $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ in which the copy constraints (12) are dualized. Each corresponding Lagrangian dual problem is solved to optimality. The generated Lagrangian cut takes the form of inequality (18), where $\nu_i^{\gamma+1,r}$ corresponds to the optimal value of the Lagrangian dual problem and coefficient $\pi_i^{\gamma+1,r,\beta}$ of variable $u^{m,\beta}$ to the optimal value of the Lagrangian multiplier relative to copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$.

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

3.5 Theoretical convergence and stopping criteria

The following property shows the theoretical convergence of the sub-tree-based SDDiP algorithm.

Proposition 1 *The forward step of the sub-tree-based SDDiP algorithm provides, with probability one, an optimal solution of Problem $P^1(0, \mathcal{X}^{1,1})$ after a finite number of iterations.*

The proof relies on Theorem 2 of Zou et al. (2019) and is straightforward. The sub-tree-based SDDiP algorithm namely complies with the three conditions which, according to this theorem, are sufficient to guarantee the finite convergence of a stochastic nested decomposition algorithm with probability one. More precisely, we have:

1. The sampling procedure in the forward step is done with replacement.

2. All the state variables involved in Problem $P^1(0, \mathcal{X}^{1,1})$ are binary. This ensures that the Lagrangian and integer optimality cuts discussed in Subsection 3.4 have the required validity, tightness and finiteness properties defined by Zou et al. (2019).

3. Each sub-problem $\hat{P}_i^{\gamma}(u_i^m, \psi_i^{\gamma}, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$ is solved with a deterministic mixed-integer linear programming solver. Hence, we can rely on the practical assumption that, given the same parent u_i^m, the same realization $\mathcal{X}^{\gamma, r_i^{k,\gamma}}$ and the same approximate cost-to-go function ψ_i^{γ}, $\hat{P}_i^{\gamma}(u_i^m, \psi_i^{\gamma}, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$ will be solved to the same optimal solution.

However, even if the finite convergence of the algorithm is guaranteed in theory, the number of iterations needed to obtain an optimal solution in practice may be prohibitively large. Hence, we introduce two stopping criteria commonly used for the SDDiP in the literature. The first one is based on a maximum number of consecutive iterations without any improvement of the lower bound, the second one on a maximum total number of iterations.

3.6 Summary

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

Note that depending on the partition of $\mathcal{S}$, the number of macro-stages Γ can take any value between 1 and Σ. For $\Gamma = 1$, the forward step corresponds to solving the original problem (1)-(4) defined on the whole scenario tree $\mathcal{V}$ and no backward step is needed: Algorithm 1 thus directly solves the stochastic problem as a MILP, without any decomposition. For $\Gamma = \Sigma$, Algorithm 1 corresponds to the SDDiP algorithm of Zou et al. (2019). In general, we will have $\Gamma < \Sigma$, which means that the number of expected cost-to-go functions $\mathcal{Q}^{\gamma}(\cdot)$ to be approximated will be smaller in Algorithm 1 than the one to be handled in the SDDiP algorithm. This may have a positive impact on the global convergence of the algorithm. Namely, with $\Gamma < \Sigma$, each sub-problem $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r})$ covers a larger portion of the planning horizon and uses an approximation of its expected future costs which will be globally better as it will rely on a smaller number of approximate expected cost-to-go functions. As a consequence, the feasible

Algorithm 1: SDDiP algorithm

```
1  Initialize $LB \leftarrow -\infty, UB \leftarrow +\infty, i \leftarrow 1$
2  while no stopping criterion is satisfied do
3      Sampling step
4      Randomly select $K$ scenarios $\Omega_i = \{\omega_i^1, ..., \omega_i^K\}$
5      Forward step
6      for $k = 1, ..., K$ do
7          for $\gamma = 1, ..., \Gamma$ do
8              Solve $\hat{P}_i^\gamma(u_i^m, \psi_i^\gamma, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$ for $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma-1)}$
9              Record $u_i^\ell$ for $\ell = \omega_i^k \cap \mathfrak{L}(\gamma, r_i^{k,\gamma})$
10         end
11         $\upsilon^k \leftarrow \sum_{n \in \omega_i^k} (f^n y_i^n + h^n s_i^n + g^n x_i^n)$
12     end
13     $\hat{\mu} \leftarrow \sum_{k=1}^K \upsilon^k$ and $\hat{\chi}^2 \leftarrow \frac{1}{K-1} \sum_{k=1}^K (\upsilon^k - \hat{\mu})^2$
14     $UB \leftarrow \hat{\mu} + z_{\alpha/2} \frac{\hat{\chi}}{\sqrt{K}}$
15     Backward step
16     for $\gamma = \Gamma - 1, ..., 1$ do
17         for $k = 1, ..., K$ do
18             Let $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$
19             for $r \in \mathcal{R}^{\gamma+1}$ do
20                 Solve the linear relaxation of $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ and
                   collect the coefficients of the strengthened Benders' cut
21                 Solve the Lagrangian relaxation of $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$
                   and collect the constant value of the strengthened Benders' cut
22                 Solve $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ and collect the coefficients of
                   the Integer optimality cut
23                 Solve the Lagrangian dual problem and collect the coefficients
                   of the Lagrangian cut
24             end
25         end
26         Add the three generated cuts to $\psi_i^\gamma$ to get $\psi_{i+1}^\gamma$
27     end
28     $LB \leftarrow \hat{Q}_{i+1}^{1,1}(0)$
29     $i \leftarrow i + 1$
30 end
```

solution obtained by solving $\hat{P}_i^\gamma(\cdot, \psi_i^\gamma, \mathcal{X}^{\gamma,r})$ at a given iteration of the algorithm will tend to be less myopic and thus to provide lower and upper bounds LB and UB of better quality. However, each sub-problem $\hat{P}_i^\gamma(\cdot, \psi_i^\gamma, \mathcal{X}^{\gamma,r})$ is now an MILP expressed on a small sub-tree. In particular, the large number of binary variables $u^{\ell,\beta}$ needed to carry out the binary approximation of the leaving inventory at each leaf node $\ell \in \mathfrak{L}(\gamma, r)$ makes its resolution computationally more expensive than the one of a sub-problem expressed on a deterministic scenario involving a single leaf node. In what follows, we thus discuss two algorithmic enhancements aiming at further improving the numerical efficiency of Algorithm 1.

4 Algorithmic Enhancements

In this section, we aim at enhancing the numerical efficiency of Algorithm 1, mostly through a more efficient building of the approximation of the expected cost-to-go functions. In what follows, we detail the two proposed algorithmic enhancements.

4.1 Approximate sub-tree-based SDDiP

Hjelmeland et al. (2018) and Quezada et al. (2019) both proposed to use an approximate variant of the SDDiP algorithm in which the state variables may be continuous. In this case, the finite convergence of the algorithm is not theoretically guaranteed but, as this approximation leads to a significant reduction of the computational effort required at each iteration of the algorithm, it may positively impact the solution quality in practice. We thus explain in what follows how this approximate SDDiP algorithm can be adapted to the case where a partial sub-tree-based decomposition of the scenario tree is used.

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

$$Q^{\gamma,r}(s^m) = \min \sum_{n \in \mathcal{X}^{\gamma,r}} \rho^n (f^n y^n + h^n s^n + g^n x^n) + \sum_{\ell \in \mathfrak{L}(\gamma,r)} \mathcal{Q}^\gamma(s^\ell) \tag{21}$$

$$s^{\xi^{\gamma,r}} + d^{\xi^{\gamma,r}} = \tilde{s}^{\xi^{\gamma,r}} + x^{\tilde{s}^{\xi^{\gamma,r}}} \tag{22}$$

$$\tilde{s}^{\xi^{\gamma,r}} = s^m \tag{23}$$

$$s^n + d^n = s^{a^n} + x^n \qquad \forall n \in \mathcal{X}^{\gamma,r} \setminus \{\xi^{\gamma,r}\} \tag{24}$$

$$\text{Constraints } (6), (8) \tag{25}$$

In this reformulation, the expected cost-to-go function $\mathcal{Q}^\gamma(s^\ell) = \sum_{r' \in \mathcal{R}^{\gamma+1}} Q^{\gamma+1,r'}(s^\ell)$ is a function of the continuous state variable s^ℓ. We thus build an under-approximation of $\mathcal{Q}^\gamma(\cdot)$ through a set of linear cuts involving continuous variables s^ℓ instead of binary variables $u^{\ell,\beta}$. Let $\tilde{\psi}_i^\gamma(\cdot)$ be the approximation of the expected cost-to-go function $\mathcal{Q}^{\gamma}(\cdot)$ available at iteration i for macro-stage γ in the approximate SDDiP algorithm. We have:

$$\tilde{\psi}_i^{\gamma}(s^{\ell}) = \min\{\theta^{\gamma\ell} : \theta^{\gamma\ell} \geq \sum_{r'\in\mathcal{R}^{\gamma+1}} (\tilde{\nu}_j^{\gamma+1,r'} + \tilde{\pi}_j^{\gamma+1,r'} s^{\ell}) \quad \forall j \in \{1, ..., i-1\}\} \tag{26}$$

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

$$\tilde{Q}_i^{\gamma,r}(s^m) = \min \sum_{n\in\mathcal{X}^{\gamma,r}} \rho^n(f^n y^n + h^n s^n + g^n x^n) + \sum_{\ell\in\mathfrak{L}(\gamma,r)} \theta^{\gamma\ell} \tag{27}$$

$$\theta^{\gamma\ell} \geq \sum_{r'\in\mathcal{R}^{\gamma+1}} (\tilde{\nu}_j^{\gamma+1,r'} + \tilde{\pi}_j^{\gamma+1,r'} s^{\ell}) \qquad \forall j = 1, ..., i-1, \quad \forall \ell \in \mathfrak{L}(\gamma, r) \tag{28}$$

$$\text{Constraints } \ (22) - (25) \tag{29}$$

In the backward step of the approximate SDDiP algorithm, only strengthened Benders' cuts are generated. More precisely, for each macro-stage $\gamma = \Gamma - 1, \ldots, 1$, the updating of the approximation of $\mathcal{Q}^{\gamma}(\cdot)$ is carried out as follows. For each node $m \in \Omega_i \cap \mathcal{V}^{t'(\gamma)}$ and each realization $r' \in \mathcal{R}^{\gamma+1}$, we first solve the linear relaxation of $\tilde{P}_i^{\gamma+1}(s_i^m, \tilde{\psi}_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r'})$, get the dual value of constraint (23) in its optimal solution and set $\tilde{\pi}_j^{\gamma+1,r'}$ to this value. We then solve a Lagrangian relaxation of $\tilde{P}_i^{\gamma+1}(s_i^m, \tilde{\psi}_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r'})$ in which constraint $\tilde{s}^{\xi^{\gamma,r}} = s^m$ has been dualized with a Lagrangian multiplier set to $\tilde{\pi}_j^{\gamma+1,r'}$. We record the optimal value of this Lagrangian relaxation and set $\tilde{\nu}_j^{\gamma+1,r'}$ to this value. Once this procedure is carried out for all realizations in $\mathcal{R}^{\gamma+1}$, we get the cut $\theta^{\gamma,m} \geq \sum_{r'\in\mathcal{R}^{\gamma+1}} (\tilde{\nu}_j^{\gamma+1,r'} + \tilde{\pi}_j^{\gamma+1,r'} s^m)$ to be added to $\tilde{\psi}_i^{\gamma}$ to get $\tilde{\psi}_{i+1}^{\gamma}$.

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

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

The second enhancement of the algorithm seeks to exploit the alternative MILP formulations currently known for SULS to generate additional strengthened Benders' cuts (and improve the approximation of the expected cost-to-go functions at a relatively limited computational effort) and speed up the resolution of the numerous MILPs to be solved over the course of Algorithm 1.

4.2.1 Generation of additional strengthened Benders' cuts

Recall that to, generate a strengthened Benders' cut to be added at a given iteration i to the approximation of $\mathcal{Q}^{\gamma-1}(\cdot)$, the algorithm first solves, for each $r \in \mathcal{R}^{\gamma}$, the linear relaxation of problem $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$, where m is a node belonging to $\mathcal{V}^{t'(\gamma-1)}$. This linear relaxation can be computed using different formulations of $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$. A first option is to use the initial MILP formulation defined in Subsection 3.3. Other options consist in using the initial MILP formulation strengthened by path inequalities (31) or the initial MILP formulation strengthened by tree inequalities (32). Then, the algorithm collects the dual value of the copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$ in the linear relaxation and set coefficient $\pi_i^{\gamma,r,\beta}$ to it. The Lagrangian relaxation of problem $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$ in which each copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$ is dualized with a Lagrangian multiplier set to $\pi_i^{\gamma,r,\beta}$ is then solved. Its optimal value provides coefficient $\nu_i^{\gamma,r}$.

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

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

Before presenting this extension, we first note that valid inequalities generated for sub-problem $\hat{P}_i^{\gamma}(u_i^m, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r})$ at a given iteration i are valid for any sub-problem $\hat{P}_j^{\gamma}(\cdot, \psi_j^{\gamma}, \mathcal{X}^{\gamma,r})$ to be solved at iteration $j \leq i$. Namely, let us consider reformulation (9)-(17) of sub-problem $P^{\gamma}(u^m, \mathcal{X}^{\gamma,r})$. Valid inequalities of type (31) and (32) can be expressed as follows:

$$\sum_{\beta\in\mathcal{B}} 2^{\beta}\hat{u}^{\xi^{\gamma,r}\beta} + \sum_{n\in S_{\mathcal{O}}} x^n + \sum_{n\in \bar{S}_{\mathcal{O}}} \Delta_n(\mathcal{O})y^n \geq d^{1n_{\mathcal{O}}} \tag{30}$$

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

Note that inequalities (30) only use local copy variables $\hat{u}$ and thus remain valid for any value of the entering stock level described by the state variable u^m. Moreover, sub-problem $P^{\gamma}(\cdot,\mathcal{X}^{\gamma,r})$ and $\hat{P}_i^{\gamma}(\cdot,\psi_i^{\gamma},\mathcal{X}^{\gamma,r})$ only differ with respect to the objective function evaluation and have the same feasible space. Thus, any inequality valid for $P^{\gamma}(\cdot,\mathcal{X}^{\gamma,r})$ is also valid for $\hat{P}_i^{\gamma}(\cdot,\psi_i^{\gamma},\mathcal{X}^{\gamma,r})$.

Let $\phi_i^{\gamma,r} = \{\sum_{\beta\in\mathcal{B}} 2^{\beta}\hat{u}^{\xi^{\gamma,r}\beta} + \sum_{n\in S_{\mathcal{O}_c}} x^n + \sum_{n\in \bar{S}_{\mathcal{O}_c}} \Delta_n(\mathcal{O}_c)y^n \geq d^{1n_{\mathcal{O}_c}}, \mathcal{O}_c \subset \mathcal{X}^{\gamma,r}, c = 1,\dots,C_i^{\gamma,r}\}$ denote the set of $C_i^{\gamma,r}$ inequalities (31) and/or (32) related to sub-problem $P^{\gamma}(\cdot,\mathcal{X}^{\gamma,r})$ generated until the beginning of iteration i. Let $\hat{P}_i^{\gamma}(\cdot,\psi_i^{\gamma},\mathcal{X}^{\gamma,r},F)$ denote problem $\hat{P}_i^{\gamma}(\cdot,\psi_i^{\gamma},\mathcal{X}^{\gamma,r})$ expressed using formulation F where $F = IF(\emptyset)$ denote the initial formulation without any strengthening and $F = IF(\phi_i^{\gamma,r})$ the initial formulation strengthened by a set of inequalities $\phi_i^{\gamma,r}$.

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

- Level $\lambda = 0$: Algorithm 1 is run as described in Section 3, i.e. we solve the linear relaxation of sub-problems $\hat{P}_i^{\gamma}(\cdot,\psi_i^{\gamma},\mathcal{X}^{\gamma,r},IF(\emptyset))$ to obtain the cut coefficients. The algorithm moves to the next level after a predefined number of consecutive iterations.

- Level $\lambda = 1$: Algorithm 1 is run using the linear relaxation of sub-problems $\hat{P}_i^{\gamma}(\cdot,\psi_i^{\gamma},\mathcal{X}^{\gamma,r},IF(\phi_i^{\gamma,r}))$ to obtain the cut coefficients. In this level, at each iteration, only path inequalities (31) are added to $\phi_i^{\gamma,r}$ using a single run of a cutting plane generation procedure based on the separation algorithm presented in Barany et al. (1984). The algorithm moves to the next level after no violated path inequalities have been found during a predefined number of consecutive iterations.

- Level $\lambda = 2$: Algorithm 1 is run using the linear relaxation of sub-problems $\hat{P}_i^{\gamma}(\cdot,\psi_i^{\gamma},\mathcal{X}^{\gamma,r},IF(\phi_i^{\gamma,r}))$ to obtain the cut coefficients. In this level, at each iteration, tree inequalities (32) are added to $\phi_i^{\gamma,r}$ using a single run of the cutting plane generation procedure presented in Guan et al. (2009).

As will be shown by the numerical results to be presented in Section 5, the joint use of strengthened Benders' cuts generated using the three alternative MILP formulations available for the sub-problems allows to significantly improve the quality of the solution provided by the algorithm. Note that an alternative strategy could be to simultaneously add strengthened Benders' cuts based on these three MILPs formulations at each iteration of the algorithm. However, this would significantly increase the computational effort carried out at each iteration as it would require the resolution of many additional linear and Lagrangian relaxations of problem $\hat{P}_i^\gamma(\cdot, \psi_i^\gamma, \mathcal{X}^{\gamma,r}, \cdot)$, namely one for each formulation $F = \{IF(\emptyset), IF(\phi_i^{\gamma,r})\}$, each set of inequalities $\phi_i^{\gamma,r}$, each realization $r \in \mathcal{R}^\gamma$ and each macro-stage $\gamma \in \mathcal{G}$. This would lead to a strong decrease in the number of iterations carried out by the algorithm within the allotted computation time, which could negatively impact its performance. We thus chose to implement a strategy in which the strengthened Benders' cuts based on the alternative MILPs formulations are added sequentially to the expected cost-to-go functions approximations.

4.2.2 Sub-problem resolution with strengthened MILP formulations

Over the course of Algorithm 1, a rather large number of MILPs have to be solved. More precisely, at each iteration of the algorithm, there are one MILP to be solved for each scenario and each macro-stage in the forward step and two MILPS (the sub-problem itself and its Lagrangian relaxation) to be solved for each scenario, each macro-stage and each realization per macro-stage in the backward step. Thus, even if each of these MILPs is of moderate size, their resolution might be computationally expensive.

Valid inequalities (31) and (32) can be used to speed up the resolution of the various MILPs involved in Algorithm 1 through a strengthening of their linear relaxation and an improvement of the lower bounds used at each node of the Branch & Bound search tree. In order to save the computational effort needed to run the related cutting plane generation procedures, we propose to strengthen the MILP formulation of each sub-problem $\hat{P}_i^\gamma(\cdot, \psi_i^\gamma, \mathcal{X}^{\gamma,r})$ using only the valid inequalities already added to the current set of inequalities $\phi_i^{\gamma,r}$ and to solve each sub-problem using formulation $IF(\phi_i^{\gamma,r})$.

For the sake of clarity, in Subsection 4.2, we focused on explaining how this second algorithmic enhancement is carried out in Algorithm 1. It can, however, be straightforwardly adapted for the approximate version of Algorithm 1 described in Subsection 4.1.

In what follows, we will refer to the version of Algorithm 1 in which the two proposed enhancements discussed in this section have been implemented as extSDDiP. A detailed description of this algorithm is provided in Appendix C in the online supplement of this work.

5 Computational Experiments

In this section, we focus on assessing the performance of the extSDDiP algorithm proposed in Sections 3 and 4. This is done by comparing it with the performance of a stand-alone mathematical programming solver ILOG-CPLEX using the extensive MILP formulation (1)-(4) and the one of the SDDiP algorithm using the dynamic programming reformulation (5)-(8) with $\mathcal{G} \equiv \mathcal{S}$.

In what follows, we first describe the scheme used to randomly generate instances of the SULS and the experimental setup. We then discuss the results of our computational experiments. All the instances and results can be found at https://github.com/FrancoQuezada/SULS-IJOC2021.

5.1 Instance Generation

We randomly generated instances following the same procedure as the one used by Guan et al. (2009). This procedure considers various scenario tree structures, several ratios of the production cost to inventory holding cost and several ratios of the setup cost to the inventory holding cost.

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

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

As for the costs, we used the same numerical values as Guan et al. (2009), i.e. a production to holding cost ratio $g/h \in \{2, 4\}$ and a setup to holding cost ratio $f/h \in \{200, 400\}$. More precisely, the holding cost h^n at node n of the tree was generated from the uniform distribution $U[0, 10]$. The production cost g^n was randomly generated from $U[0.8(g/h)\bar{h}, 1.2(g/h)\bar{h}]$, where $\bar{h} = \sum_{n \in \mathcal{V}} h^n / |\mathcal{V}|$ is the average holding cost. The setup cost f^n was randomly generated from $U[0.8(f/h)\bar{h}, 1.2(f/h)\bar{h}]$. The demand d^n was generated from the discrete uniform distribution $DU[0, 100]$. The probability ρ^n of node n is given by $\rho^n = (\frac{1}{R})^{\sigma^n - 1}$. Finally, for each set and each considered combination of Σ, b, R, g/h and f/h, five random instances were generated, resulting in a total of 140 instances.

5.2 Experimental setup

Each instance is first solved with the mathematical programming solver CPLEX 12.8 using the extensive MILP formulation (1)-(4). For the Set 1 instances which involve less than 8000 scenarios, we use the cutting-plane generation strategy proposed by Guan et al. (2009) to strengthen this formulation by adding violated path inequalities (31) and tree inequalities (32) at each node of the branch-and-cut search tree. For the other instances, we use the standard branch-and-cut algorithm of solver CPLEX 12.8 using the initial formulation (1)-(4). Our preliminary experiments namely showed that this was more efficient than using the customized branch-and-cut algorithm based on valid inequalities (31) and (32) for the large instances. This solution method is denoted by CPX in what follows.

Each instance is then solved by the SDDiP algorithm proposed by Zou et al. (2019) and by the extSDDiP algorithm. For both algorithms, the number of scenarios sampled at each iteration is set to $K = 1$. The computational results presented by Zou et al. (2019) namely showed that, on average, the SDDiP algorihm is more efficient when sampling a small number of scenarios at each iteration. Moreover, the binary approximation of the continuous state variables s^n is carried out as follows. For each instance, we compute an upper bound of the inventory level at node n as $s_{max} = max_{\ell \in \mathcal{L}(1)} d^{1,\ell}$. The number B of binary variables $u^{n,\beta}$ is set to $B = \lceil log_2(s_{max}) \rceil$.

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

Furthermore, in order to better assess the impact of the two enhancements presented in Section 4, several variants of the extSDDiP algorithm are implemented.

First, to evaluate the usefulness of the approximate subtree-based SDDiP algorithm discussed in Subsection 4.1, we consider the following three implementations of the extSDDiP algorithm:

- extSDDiP-I corresponds to the case where only the approximate subtree-based SDDiP algorithm based on formulation (5)-(8) with continuous state variables s^n is run.
- extSDDiP-II corresponds to the case where only Algorithm 1 is run.
- extSDDiP-I/II corresponds to a 2-phase algorithm in which phase I first runs the approximate subtree-based SDDiP algorithm to build an initial approximation of the expected cost-to-go functions and phase II runs Algorithm 1 to further improve these approximations.

Second, we also seek to assess the impact on the algorithmic performance of exploiting alternative MILP formulations of the SULS, as presented in Subsection 4.2. Each considered setting is described by the maximum level of formulation strengthening λ_{max} used to strengthen the sub-problem formulation. Thus, extSDDiP-I/II-λ_{max} denotes for instance a 2-phase implementation of the extSDDiP in which the sub-problem formulation strengthening levels $0, \dots, \lambda_{max}$ are sequentially used following the strategy described at the end of Subsection 4.2.1.

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

Each algorithm was implemented in C++ using the Concert Technology environment. All (mixed-integer) linear programs were solved using CPLEX 12.8 with the default settings and the Lagrangian dual problems were solved by a sub-gradient algorithm. All tests were run on the computing infrastructure of the Laboratoire d'Informatique de Paris VI (LIP6), which consists of a cluster of Intel Xeon Processors X5690. We set the cluster to use two 3.46GHz cores and 12GB RAM to solve each instance. Nevertheless, when additional RAM was required for solving large MILP models with method CPX, we allowed memory overflow to obtain consistent results for all the experiments. We impose a time limit of 1800 seconds to method CPX to solve each instance. For the SDDiP and extSDDiP algorithms, we impose a time limit of 900 seconds to compute a lower bound and 900 seconds to compute the true or statistical upper bound.

5.3 Results

Tables 1-4 display the numerical results. Columns R and b describe the structure of the scenario tree when needed. The corresponding number of nodes in the scenario tree, $|\mathcal{V}|$, and the number of scenarios, $|\mathcal{L}(1)|$, are then provided. Column G indicates the number of stages per macro-stage in the partial decomposition of the scenario tree and Column "Method" indicates the algorithm used to solve each instance. Each line in the table thus provides the average results of the indicated resolution method over the 20 instances corresponding to the given scenario tree structure with various values of the ratios f/h and g/h. Column "Gap" displays the gap between the lower bound (LB) and the upper bound (UB) found by each method, i.e. $Gap = |UB - LB|/UB$. The average total computation time in seconds is reported in Column "Time (s)", the average number of iterations in Column "# ite" and the total number of valid inequalities of type (31) and (32) generated are provided in Column "# VI".

Results from Table 1 first show that, when using the extensive formulation (1)-(4) strengthened by valid inequalities (31) and (32), method CPX outperforms the other methods for the smallest considered instances, i.e. the instances corresponding to $\Sigma = 4$, $R = 10$ and $b = 1$, providing an average gap of 1.36% within the allotted time limit. When the number of realizations per stage increases, i.e. for the instances corresponding to $\Sigma = 4$, $R = 20$ and $b = 1$, the relative performance of method CPX deteriorates but the average gap remains below 5%. However, when the number of stages, and consequently the size of the scenario tree, increases, the performance of method CPX strongly deteriorates. This can be seen from the results displayed in Tables 2-4: method CPX namely provides gaps between 19% for the instances with $\Sigma = 6$, $R = 10$ and $b = 1$ and 94% for the instances with $\Sigma = 6$ and $R = 20$ and $b = 1$.

We also observe from the results displayed in Tables 1-4 that method SDDiP compares well with method CPX. It namely consistently provides average gaps between 9% and 30% for all instances and significantly outperforms method CPX in terms of solution quality for the largest instances.

Furthermore, these results show that, on average, algorithm extSDDiP significantly outperforms methods CPX and SDDiP. Indeed, we note that, whatever the partial decomposition and the formulation strengthening setting used, i.e. whatever the value of $G \in \{2, 3, 4, 6\}$ and $\lambda_{max} \in \{0, 1, 2\}$, the gap provided by algorithm extSDDiP-I/II is significantly smaller than the one provided by algorithm SDDiP. In particular, if we consider the results obtained with algorithm extSDDiP-I/II-2 with a partial decomposition using $G = 2$ stages per macro-stage, we obtain an average gap over the 140 instances of 5.07% as compared to an average gap of 19.69% obtained with algorithm SDDiP and an average gap of 41.16% obtained with CPX. Moreover, we would like to point out that, by considering, for each set of instances, the values of G and λ_{max} providing the lowest gap, algorithm extSDDiP-I/II is able to provide a solution within an average gap of 3.76%. This clearly shows that jointly using the partial decomposition of the scenario tree into sub-trees discussed in Section 3 and the two algorithmic enhancements presented in Section 4 leads to a significant improvement of the performance of the SDDiP algorithm proposed by Zou et al. (2019).

We now discuss the individual impact of each of these three elements on the performance of algorithm extSDDiP.

The separate impact of the partial decomposition of the scenario tree into sub-trees on the algorithmic performance can be evaluated by looking at the results obtained by algorithm extSDDiP-II-0: see Tables 10-13 in Appendix D in the online supplement of this work. We thus observe that the average gap can be reduced from 19.68% with algorithm SDDiP to 11.16% with algorithm extSDDiP-II-0 using a partial decomposition involving $G = 2$ stages per macro-stage. Moreover, increasing the value of G further improves the performance of algorithm extSDDiP-II-0 as long as the size of the obtained sub-problems remains small enough to enable a MILP solver to solve them in a short computation time. Thus, in Tables 10-13, we observe that the average gap is reduced to 7.50% when using algorithm extSDDiP-II-0 with $G \in \{3, 4, 6\}$. However, for the instances corresponding to $\Sigma = 6$, $R = 20$ and $b = 1$, the increase of G from 2 to 3 leads to a strong deterioration of the algorithmic performance: see Table 11. This might be explained by the fact that, for these instances, when $G = 3$, each sub-tree involves 421 nodes, among which 400 are leaf nodes.

A binary approximation of the leaving inventory level has to be used for each of these 400 nodes so that each sub-problem comprises a rather large number of binary variables and requires a large computational effort to be solved. It is thus not possible to carry out one full iteration of algorithm extSDDiP-II-0 within the allotted computation time. This observation is confirmed by the results reported in Table 14 in Appendix E in the online supplement of this work. These results were obtained while solving an additional set of instances involving a large number ($\Sigma = 20$) of decision stages with the extSDDiP-I/II algorithm. They show that the algorithm performs better when using a value of G in $\{2, 4, 5\}$ than when using a value of G equal to 10.

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

To get an assessment of the usefulness of exploiting the alternative MILP formulations available for SULS, we first look at the results provided in Tables 10-13 in Appendix D. We thus note that when $G = 2$, the average gap is reduced from 11.16% with algorithm extSDDiP-II-0 to 7.43% with algorithm extSDDiP-II-1. This clearly shows the positive impact of generating strengthened Benders' cut using a linear relaxation of the sub-problems strengthened by path inequalities (31). However, for larger values of $G \in \{3, 4, 6\}$, we observe that the average gap is increased from 7.50% with algorithm extSDDiP-II-0 to 8.88% with algorithm extSDDiP-II-1. This deterioration might be explained by the large number of valid inequalities added to the sub-problem formulations: see e.g. the instances corresponding to $\Sigma = 6$, $R = 20$ and $b = 1$ for which more than 20,000 path inequalities are added to the various sub-problems. Thus, adding too many valid inequalities leads to an increase in the size of the MILPs to be solved at each iteration of the extSDDiP algorithm so that fewer iterations can be carried out and weaker lower bounds are provided. Moreover, results from Tables 10-13 also seem to indicate that the use of additional sub-problem formulation strengthening techniques does not enable algorithm extSDDiP-II-2 to provide better quality solutions. Finally, we note that the proposed additional strengthened Benders' cuts may positively impact the performance of the SDDiP algorithm of Zou et al. (2019), even when there is no partial decomposition. Namely, provided that the number of periods per stage is greater than 1, each sub-problem to be solved in the SDDiP algorithm is a small multi-period deterministic lot-sizing problem whose formulation can be strengthened by path inequalities (31). Line 2 of Table 12 in Appendix D in the online supplement of this work reports the results obtained with the SDDiP-1 algorithm in which additional strengthened Benders' cuts based on sub-problem formulations strengthened by path inequalities (31) are generated. These results show the positive impact of these additional cuts on the algorithmic performance, reducing the gap from over 20% (resp. 13%) with the SDDiP algorithm to less than 10% (resp. 6%) with the SDDiP-1 algorithm for the instances with $b = 2$ (resp. $b = 5$) periods per stage.

Thus, each of the two enhancements proposed for the extSDDiP algorithm, when used separately, has a moderate positive impact on the solution quality. However, their combined use, in particular the use of alternative MILP formulations of the SULS to generate additional strengthened Benders' cuts in Phase I of the extSDDiP algorithm, seems to significantly improve the algorithmic performance. Namely, when $G = 2$, the average gap is reduced from 10.37% with algorithm extSDDiP-I/II-0 to 5.89% with algorithm extSDDiP-I/II-1 and 5.06% with algorithm extSDDiP-I/II-2. Similarly, when $G \in \{3, 4, 6\}$, the average gap is reduced from 8.34% with algorithm extSDDiP-I/II-0 to 5.32% with algorithm extSDDiP-I/II-1 and 5.47% with algorithm extSDDiP-I/II-2. This improvement might be explained by the fact that, when running algorithm extSDDiP-I/II with the formulation strengthening settings 1 or 2, more iterations of phase I of the algorithm are carried out than when running algorithm extSDDiP-I/II-0. This enables algorithms extSDDiP-I/II-1 and extSDDiP-I/II-2 to generate more strengthened Benders' cut in Phase I than algorithm extSDDiP-I/II-0 and consequently to obtain a better initial lower bound at the end of phase I. All three algorithms will improve this lower bound during phase II. However, a phase II iteration is more computationally demanding than a phase I iteration so that only a limited number of phase II iterations might be carried out within the allotted time. Thus, using algorithm extSDDiP-I/II-0, does not enable to generate enough cutting planes during phase II to make up for the difference with algorithms extSDDiP-I/II-1 and extSDDiP-I/II-2 in the lower bound value obtained at the end of phase I.

Finally, we would like to point out that the approximate sub-tree based algorithm, i.e. extSDDiP-I, when combined with the use of sub-problem strengthening techniques to generate additional strengthened Benders' cuts, shows a remarkable computational performance. The results provided in Tables 6-9 (see Appendix D in the online supplement of this work) namely show that using algorithm extSDDiP-I-2 with $G = 2$, enables to provide solutions displaying an average gap of 6.27% within an average computation time of only 166.71s. This suggests that algorithm extSDDiP-I, even if it does not have any theoretical guarantee of convergence, could be used as a heuristic solution approach capable of providing in practice near-optimal solutions in reduced computation times.

6 Conclusions and perspectives

We investigated a multi-stage stochastic mixed-integer linear programming approach for the SULS problem and focused on the resolution of instances involving large-size scenario trees. We presented a new extension of the SDDiP algorithm proposed by Zou et al. (2019). This new extension is based on three main features: the partial decomposition of the stochastic problem into smaller stochastic sub-problems (rather than into deterministic sub-problems), the introduction of an initial phase in which the state variables are kept continuous and the exploitation of alternative MILP formulations of the stochastic sub-problems to generate additional strengthened Benders' cuts. Computational experiments carried out on randomly generated instances show that the proposed extended algorithm significantly outperforms the original SDDiP algorithm.

A first interesting direction for further research could be to extend this work to single-item single-echelon stochastic lot-sizing problems involving complicating features such as the possibility of backlogging the demand, a limited production capacity or upper bounds on the inventory level. These extensions of the SULS problem would comprise a limited number of continuous state variables at each node and valid inequalities that could be used to obtain alternative MILP formulations. These inequalities are known for most of them (see e.g. Pochet and Wolsey (2006)). It should thus be possible to adapt the two-phase algorithm extSDDiP-I/II for these problems. It might also be worth investigating whether the proposed extended algorithm could be used to solve multi-item and/or multi-echelon stochastic lot-sizing problems. For such problems, the number of continuous state variables for which a binary approximation would have to be built will be much larger so that the use of the one-phase algorithm extSDDiP-I might be more appropriate. Finally, in many practical settings, assuming a stage-wise independent stochastic process may not be suitable and there may be temporal correlations, in particular in the demand parameter. It would thus be interesting to study how the algorithmic enhancements proposed in the present work for the SDDiP may be exploited to improve the computational efficiency of extensions of the SDDiP dealing with stage-wise dependent stochastic processes such as the ones investigated by Shapiro et al. (2013) and Philpott and de Matos (2012).

Acknowledgment: The authors are grateful for the partial support provided by the French Gaspard Monge Program for Optimization, Operational research and their interactions with data science (PGMO) and the Fondation Mathématique Jacques Hadamard (FMJH). This work was also partially funded by the Agencia Nacional de Investigación y Desarrollo (ANID) / Becas de doctorado en el extranjero / BECAS CHILE/2018 - 72190160. The authors also would like to thank the anonymous referees who spotted errors and provided important suggestions to improve this manuscript.

References

Aggarwal A, Park JK (1993) Improved algorithms for economic lot size problems. *Operations Research* 41:549–571.

Ahmed S, King AJ, Parija G (2003) A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization* 26(1):3–24.

Aldasoro U, Escudero LF, Merino M, Monge JF, Pérez G (2015) On parallelization of a stochastic dynamic programming algorithm for solving large-scale mixed 0–1 problems under uncertainty. *TOP* 23(3):703–742.

Barany I, Van Roy TJ, Wolsey LA (1984) Strong formulations for multi-item capacitated lot sizing. *Management Science* 30(10):1255–1261.

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.

Camargo VC, Toledo FM, Almada-Lobo B (2014) HOPS–Hamming-oriented partition search for production planning in the spinning industry. *European Journal of Operational Research* 234(1):266–277.

Cerisola S, Ramos A (2000) Node aggregation in stochastic nested Benders decomposition applied to hydrothermal coordination. *PMAPS2000: 6th International Conference on Probabilistic Methods Applied to Power Systems.*

Cristobal MP, Escudero LF, Monge JF (2009) On stochastic dynamic programming for solving large-scale planning problems under uncertainty. *Computers & Operations Research* 36(8):2418–2428.

Di Summa M, Wolsey LA (2008) Lot-sizing on a tree. *Operations Research Letters* 36(1):7–13.

Escudero LF, Monge JF, Morales DR (2018) On the time-consistent stochastic dominance risk averse measure for tactical supply chain planning under uncertainty. *Computers & Research* 100:270–286.

Ghamari A, Sahebi H (2017) The stochastic lot-sizing problem with lost sales: A chemical-petrochemical case study. *Journal of Manufacturing Systems* 44:53–64.

Guan Y, Ahmed S, Miller AJ, Nemhauser GL (2006a) On formulations of the stochastic uncapacitated lot-sizing problem. *Operations Research Letters* 34(3):241–250.

Guan Y, Ahmed S, Nemhauser GL (2009) Cutting planes for multistage stochastic integer programs. *Operations Research* 57(2):287–298.

Guan Y, Ahmed S, Nemhauser GL, Miller AJ (2006b) A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming* 105(1):55–84.

Guan Y, Miller AJ (2008) Polynomial-time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research* 56(5):1172–1183.

Halman N, Klabjan D, Mostagir M, Orlin J, Simchi-Levi D (2009) A fully polynomial-time approximation scheme for single-item stochastic inventory control with discrete demand. *Mathematics of Operations Research* 34(3):674–685.

Hjelmeland MN, Zou J, Helseth A, Ahmed S (2018) Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming. *IEEE Transactions on Sustainable Energy* 10(1):481–490.

Hu Z, Hu G (2016) A two-stage stochastic programming model for lot-sizing and scheduling under uncertainty. *International Journal of Production Economics* 180:198–207.

Kilic OA, Tunc H, Tarim SA (2018) Heuristic policies for the stochastic economic lot sizing problem with remanufacturing under service level constraints. *European Journal of Operational Research* 267(3):1102–1109.

Macedo PB, Alem D, Santos M, Junior ML, Moreno A (2016) Hybrid manufacturing and remanufacturing lot-sizing problem with stochastic demand, return, and setup costs. *International Journal of Advanced Manufacturing Technology* 82(5-8):1241–1257.

Moreno A, Alem D, Ferreira D, Clark A (2018) An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. *European Journal of Operational Research* 269(3):1050–1071.

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

Philpott AB, de Matos VL (2012) Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research* 218(2):470 – 483.

Pochet Y, Wolsey LA (2006) *Production planning by mixed-integer programming* (Springer).

Quezada F, Gicquel C, Kedad-Sidhoum S (2019) A stochastic dual dynamic integer programming for the uncapacitated lot-sizing problem with uncertain demand and costs. *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 353–361.

Scarf H (1959) The optimality of (s, S) policies in the dynamic inventory problem. Technical report, Stanford University.

Shapiro A, Tekaya W, Costa J, Soares M (2013) Risk neutral and risk averse stochastic dual dynamic programming method. *European Journal of Operational Research* 224:375–391.

Wagelmans A, Van Hoesel S, Kolen A (1992) Economic lot sizing: an $\mathcal{O}(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research* 40(1-supplement-1):S145–S156.

Wagner HM, Whitin TM (1958) Dynamic version of the economic lot size model. *Management Science* 5(1):89–96.

Zhao C, Guan Y (2014) Extended formulations for stochastic lot-sizing problems. *Operations Research Letters* 42(4):278–283.

Zou J, Ahmed S, Sun XA (2019) Stochastic dual dynamic integer programming. *Mathematical Programming* 175(1-2):461–502.

A Strengthening of the extensive MILP formulation

This section recalls previously published results on the polyhedral structure of the extensive formulation (1)-(4) and their use to strengthen its linear relaxation. Formulation (1)-(4) can be strengthened by applying valid inequalities known for the deterministic ULS to each path of the scenario tree.

Proposition 2 *Guan et al. (2006b).*
Given $\ell \in \mathcal{V}$ and $S \subseteq \mathcal{P}(1,\ell)$, the following (ℓ, S) inequality is valid for problem (1)-(4)*:*

$$s^0 + \sum_{n \in S} x^n + \sum_{n \in \bar{S}} d^{n\ell} y^n \geq d^{1\ell} \tag{31}$$

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

For the deterministic ULS, the (ℓ, S) inequalities provide a full description of the convex hull of the feasible space. It is however not the case for the SULS. Guan et al. (2009) thus proposed to further strengthen formulation (1)-(4) by exploiting its tree structure.

Proposition 3 *Guan et al. (2009).*
Given a subset $\mathcal{O} = \{n_1, ..., n_O\} \subseteq \mathcal{V}$ which is partially ordered such that $0 = d^{1n_0} \leq d^{1n_1} \leq ... \leq d^{1n_O}$ and a subset $S_{\mathcal{O}}$ of nodes belonging to $\cup_{o=1...O}\mathcal{P}(1, n_O)$, the following inequality is valid for problem (1)-(4)*:*

$$s^0 + \sum_{n \in S_{\mathcal{O}}} x^n + \sum_{n \in \bar{S}_{\mathcal{O}}} \Delta_n(\mathcal{O}) y^n \geq d^{1n_O} \tag{32}$$

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

Note that inequalities (31) can be seen as a special case of inequalities (32) in which $\mathcal{O}$ comprises a single node ℓ. Guan et al. (2006a) showed that a particular case of inequalities (32) suffices to fully describe the convex hull of the two-period SULS. However, this is not the case anymore when more than two planning periods are involved in the problem.

The number of valid inequalities (31) and (32) is too large to allow adding all of them a priori to formulation (1)-(4). Hence, a cutting-plane generation strategy is needed to add only a subset of these valid inequalities into the formulation. Consequently, the corresponding separation problems must be solved in order to identify which inequalities should be incorporated in the formulation. For inequalities (31), the separation problem can be solved exactly by the polynomial time algorithm proposed by Barany et al. (1984). For inequalities (32), the complexity status of the separation problem remains unknown but Guan et al. (2009) proposed a heuristic separation algorithm.

B Generation of strengthened Benders' cuts using alternative MILP formulations of the sub-problems

This section provides additional insights about the generation of additional strengthened Benders' cuts using alternative MILP formulations of the sub-problems. We present a numerical example illustrating the fact that there is no dominance between the strengthened Benders' cuts generated while using different MILP formulations of the SULS. To do this, we use formulation (21)-(25) rather than formulation (9)-(17). It namely has a single continuous state variable at each node, which facilitates the graphic representation of the approximation of the expected cost-to-go function.

Consider the following scenario tree involving $\mathcal{S} = 4$ stages and $R^{\sigma} = 3$ realizations at stages 2 to 4. Each realization is described by its stage σ and its realization index $r \in \{1, 2, 3\}$. The structure of the scenario tree is provided in Figure 2 and the corresponding values of the uncertain parameters are provided in Table 5.

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

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

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

$$\theta^1 \geq 682.18 - 8.24 s^{\ell} \tag{33}$$

- by solving the LP relaxation of sub-problems $\tilde{P}_1^2(0, 0, \mathcal{X}^{2,r}, IF(\phi_1^{2,r}))$, $r \in$

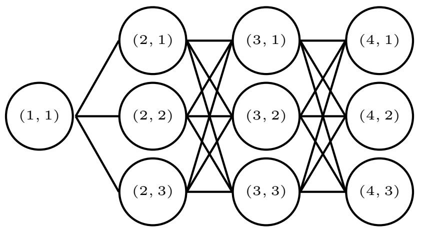

Figure 2: Scenario tree structure for Example 1

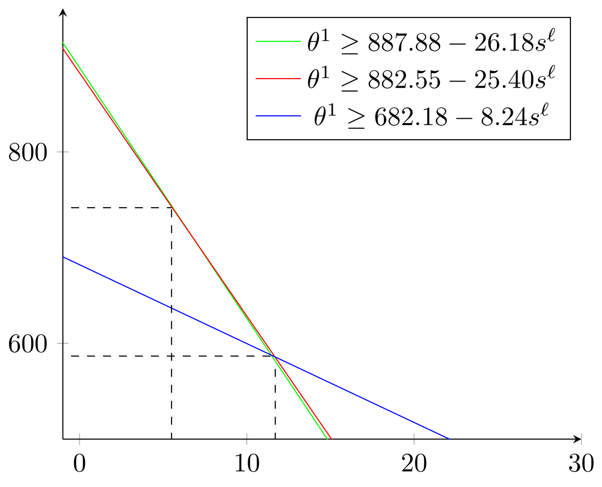

Figure 3: Illustration for Example B.

$\{1,2,3\}$, in which $\phi_1^{2,r}$ contains only path inequalities (31):

$$\theta^1 \geq 882.55 - 25.40s^\ell \tag{34}$$

- by solving the LP relaxation of sub-problems $\tilde{P}_1^2(0,0,\mathcal{X}^{2,r}, IF(\phi_1^{2,r}))$, $r \in \{1,2,3\}$, in which $\phi_1^{2,r}$ contains both path inequalities (31) and tree inequalities (32):

$$\theta^1 \geq 887.88 - 26.18s^\ell \tag{35}$$

Figure B provides a graphic representation of the three generated cuts. We observe that there is no dominance amongst them, i.e. none of the cuts seems to provide a better approximation of the expected cost-to-go function for all possible values of the leaving inventory level s^ℓ. More specifically, when s^ℓ lies in the interval $[1,7]$, the best approximation is obtained by the cut generated using formulation $IF(\phi_i^{\gamma,r})$ strengthened by inequalities (31) and (32). When s^ℓ lies in $[7,12]$, the best approximation is obtained by the cut generated using formulation $IF(\phi_i^{\gamma,r})$ strengthened only by inequalities (31). Finally, when s^ℓ is greater than 12, the best approximation is obtained by the cut generated using formulation $IF(\emptyset)$. This shows the practical interest of generating strengthened Benders'cuts based on alternative MILP formulation of the sub-problems.

C Detailed description of the proposed extSD-DiP

Algorithm 2: Strengthening sub-problems algorithm

```
1  if $\lambda = 0$ then
2      $F = IF(\emptyset))$
3  else if $\lambda = 1$ then
4      Run the cutting plane procedure to generate path inequalities (31)
5      Add the generated inequalities to $\phi_i^{\gamma,r}$ to get $\phi_{i+1}^{\gamma,r}$
6      $F = IF(\phi_i^{\gamma,r})$
7  else if $\lambda = 2$ then
8      Run the cutting plane procedure to generate tree inequalities (32)
9      Add the generated inequalities to $\phi_i^{\gamma,r}$ to get $\phi_{i+1}^{\gamma,r}$
10     $F = IF(\phi_i^{\gamma,r})$
```

Algorithm 3: Approximate extSDDiP algorithm

```
1  while no stopping criterion is satisfied do
2      Randomly select $K$ scenarios $\Omega_i = \{\omega_i^1, ..., \omega_i^K\}$
3      for $k = 1, ..., K$ do
4          for $\gamma = 1, ..., \Gamma$ do
5              Solve $\tilde{P}_i^\gamma(s_i^m, \tilde{\psi}_i^\gamma, \mathcal{X}^{\gamma, r_i^{k,\gamma}}, IF(\phi_i^{\gamma,r}))$ for $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma-1)}$
6              Record $s_i^\ell$ for $\ell = \omega_i^k \cap \mathfrak{L}(\gamma, r_i^{k,\gamma})$
7          end
8      end
9      for $\gamma = \Gamma - 1, ..., 1$ do
10         for $k = 1, ..., K$ do
11             for $r \in \mathcal{R}^{\gamma+1}$ do
12                 Let $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$
13                 Run Algorithm 2
14                 Solve the Linear relaxation of
                     $\tilde{P}_i^{\gamma+1}(s_i^m, \tilde{\psi}_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r}, F(\phi_{i+1}^{\gamma,r}))$ and collect the coefficients
                     of the strengthened Benders' cut
15                 Solve the Lagrangian relaxation of
                     $\tilde{P}_i^{\gamma+1}(s_i^m, \tilde{\psi}_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r}, F(\phi_{i+1}^{\gamma,r}))$ and collect the constant
                     value of the strengthened Benders' cut
16             end
17             Add the generated cut to $\tilde{\psi}_i^\gamma$ to get $\tilde{\psi}_{i+1}^\gamma$
18         end
19         if a criterion for Algorithm 2 is satisfied then
20             $\lambda \leftarrow \lambda + 1$
21         end
22         $LB \leftarrow \tilde{Q}_{i+1}^{1,1}(0)$
23         $i \leftarrow i + 1$
24     end
25 end
```

D Computational assessment of the various components of the extSDDiP algorithm

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

Tables 6-9 provide the results obtained while running extSDDiP-I, i.e. running only the approximate sub-tree based algorithm. Tables 10-13 provide the results obtained while running extSDDiP-II, i.e. running only Algorithm 1 without an initial phase based on the approximate sub-tree based algorithm.

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

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

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

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

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

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

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

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

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

Table 5: Numerical values for Example 1

Algorithm 4: extSDDiP algorithm

```
Initialize $LB \leftarrow -\infty, UB \leftarrow +\infty, i \leftarrow 1, \lambda \leftarrow 0$
for $\gamma = \Gamma - 1, ..., 1$ do
    for $r \in \mathcal{R}^{\gamma+1}$ do
        Initialize $\phi^{\gamma,r} \leftarrow \emptyset$
    end
end
Run Algorithm 3
while no stopping criterion is satisfied do
    Randomly select $K$ scenarios $\Omega_i = \{\omega_i^1, ..., \omega_i^K\}$
    for $k = 1, ..., K$ do
        for $\gamma = 1, ..., \Gamma$ do
            Solve $\hat{P}_i^{\gamma}(u_i^m, \psi_i^{\gamma}, \mathcal{X}^{\gamma, r_i^{k,\gamma}}, IF(\phi_i^{\gamma,r}))$ for $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma-1)}$.
            Record $u_i^{\ell}$ for $\ell = \omega_i^k \cap \mathfrak{L}(\gamma, r_i^{k,\gamma})$
        end
        $\upsilon^k \leftarrow \sum_{n \in \omega_i^k} (f^n y_i^n + h^n s_i^n + g^n x_i^n)$
    end
    $\hat{\mu} \leftarrow \sum_{k=1}^{K} \upsilon^k$ and $\hat{\chi}^2 \leftarrow \frac{1}{K-1} \sum_{k=1}^{K} (\upsilon^k - \hat{\mu})^2$
    $UB \leftarrow \hat{\mu} + z_{\alpha/2} \frac{\hat{\chi}}{\sqrt{K}}$
    for $\gamma = \Gamma - 1, ..., 1$ do
        for $k = 1, ..., K$ do
            for $r \in \mathcal{R}^{\gamma+1}$ do
                Let $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$
                Run Algorithm 2
                Solve the Linear relaxation of
                  $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r}, F(\phi_{i+1}^{\gamma,r}))$ and collect the coefficients
                  of the strengthened Benders' cut
                Solve the Lagrangian relaxation of
                  $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r}, F(\phi_{i+1}^{\gamma,r}))$ and collect the constant
                  value of the strengthened Benders' cut
                Solve $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r}, F(\phi_{i+1}^{\gamma,r}))$ and collect the
                  coefficients of the integer optimality cut
                -Solve the Lagrangian dual problem and collect the coefficients
                  of the Lagrangian cut
            end
            Add the 3 generated cuts to $\psi_i^{\gamma}$ to get $\psi_{i+1}^{\gamma}$
        end
        if a criterion for Algorithm 2 is satisfied then
            $\lambda \leftarrow \lambda + 1$
        end
        $LB \leftarrow \hat{Q}_{i+1}^{1,1}(0)$
        $i \leftarrow i + 1$
    end
end
```

Table 6: Performance of the approximate algorithm at solving instances from Set 1 ($\Sigma = 4, b = 1$) of the SULS problem

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

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

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

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

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

Table 9: Performance of the approximate algorithm at solving instances from Set 4 ($\Sigma = 12, R = 3, b = 1$) of the SULS

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

Table 10: Performance of Algorithm 1 at solving instances from Set 1 ($\Sigma = 4, b = 1$) of the SULS problem

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

Table 11: Performance of Algorithm 1 at solving instances from Set 2 ($\Sigma = 6, b = 1$) of the SULS problem

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

Table 12: Performance of Algorithm 1 at solving instances from Set 3 ($\Sigma = 8, R = 5$) of the SULS problem

b	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(1)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
2	195312	78125	1	SDDiP	20.08	1662.00	91	0
				SDDiP-1	9.72	1,617.19	115	121
			2	extSDDiP-II-0	13.42	1,653.96	33	0
				extSDDiP-II-1	6.93	1,665.12	37	1,106
				extSDDiP-II-2	7.21	1,667.20	35	1,101
			4	extSDDiP-II-0	6.91	1,475.52	13	0
				extSDDiP-II-1	9.29	1,645.23	9	10,933
				extSDDiP-II-2	8.80	1,632.74	9	11,075
			8	CPX	43.81	1,802.40	-	0
5	488280	78125	1	SDDiP	13.14	2075.09	70	0
				SDDiP-1	5.71	2,025.49	100	1,204
			2	extSDDiP-II-0	7.76	1,780.95	29	0
				extSDDiP-II-1	3.66	1,730.56	35	4,815
				extSDDiP-II-2	3.53	1,703.66	37	4,831
			4	extSDDiP-II-0	6.41	1,629.26	10	0
				extSDDiP-II-1	24.18	1,864.58	4	20,673
				extSDDiP-II-2	26.63	1,862.54	4	18,764
			8	CPX	71.04	1,803.99	-	0

Table 13: Performance of Algorithm 1 at solving instances from Set 4 ($\Sigma = 12, R = 3, b = 2$) of the SULS problem

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

E Computational results on instances involving a large number of stages

This section provides the results of additional computational experiments carried out to assess the performance of the extSDDiP algorithm when using large values of G.

We used the procedure described in Subsection 5.1 of the manuscript to generate a new set of instances involving $\Sigma = 20$ stages, $b = 1$ period per stage and $R \in \{2, 3\}$ realizations per stage. We solve these instances with the mathematical solver CPLEX using the extensive MILP formulation, with the SDDiP algorithm proposed by Zou et al. (2019) and with the extSDDiP-I/II-2 algorithm for values of G in the set $\{2, 4, 5, 10\}$. The results are reported in Table 14.

As already discussed in Subsection 5.3, we first note that increasing G from 1 to 2, 4 or 5 leads to a significant improvement in the solution quality. However, further increasing G to 10 leads to a degradation of the algorithmic performance. This can be seen in particular for the instances corresponding to $R = 3$: on average, for these instances, the extSDDiP-I/II-2 algorithm with $G = 10$ provides solutions within a gap significantly larger than the one obtained using the SDDiP algorithm.

This might be explained as follows. When G is large, the extSDDiP algorithm decomposes the initial problem into a small number of large sub-problems. Solving each one of these sub-problems is computationally demanding so that the algorithm is able to perform only a limited number of iterations within the allotted time. Note e.g. how, on average, the extSDDiP-I/II-2 algorithm with $G = 10$ is able to carry out only 11 iterations when solving instances with $R = 3$ whereas the same algorithm solving the same instances with $G = 2$ carries out an average of 286 iterations. This reduction in the number of iterations negatively impacts the quality of the approximate expected cost-to-go functions built over the course of the algorithm, and as a consequence, the quality of the lower and upper bounds obtained by the algorithm.

Note that this phenomenon is also observed in the results reported in Tables 2 and 11 for the instances of Set 2 involving $R = 20$ realizations per stage. For these instances, increasing G from 2 to 3 negatively impacts the solution gap.

Table 14: Performance of each method at solving instances with $\Sigma = 20$ and $b = 1$ of the SULS problem

R	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}(1)\rvert$	G	Method	Gap	Time (s)	# ite	# VI
2	1,048,570	524,288	1	SDDiP	20.74	2,381.83	116	0
			2	extSDDiP-I/II-0	16.21	1,171.41	151	0
				extSDDiP-I/II-1	4.96	1,114.92	219	105
				extSDDiP-I/II-2	8.37	1,188.19	216	105
			4	extSDDiP-I/II-0	15.23	1,700.43	56	0
				extSDDiP-I/II-1	5.63	1,288.44	95	639
				extSDDiP-I/II-2	5.67	1,282.92	142	737
			5	extSDDiP-I/II-0	11.98	1,820.73	42	0
				extSDDiP-I/II-1	6.53	1,565.08	72	1,102
				extSDDiP-I/II-2	6.17	1,553.27	103	1,495
			10	extSDDiP-I/II-0	6.51	1,240.04	13	0
				extSDDiP-I/II-1	7.05	1,396.39	20	3,978
				extSDDiP-I/II-2	6.45	1,430.71	25	4,390
			20	CPX	88.58	1,805.53	-	-
3	$1.74 \cdot 10^9$	$1.16 \cdot 10^9$	1	SDDiP	24.23	2,212.64	93	0
			2	extSDDiP-I/II-0	26.51	1,128.94	108	0
				extSDDiP-I/II-1	9.52	1,013.80	229	276
				extSDDiP-I/II-2	11.21	1,061.60	286	275
			4	extSDDiP-I/II-0	18.45	1,862.04	53	0
				extSDDiP-I/II-1	9.23	1,855.41	96	2,719
				extSDDiP-I/II-2	8.03	1,605.79	127	5,697
			5	extSDDiP-I/II-0	13.54	1,910.29	36	0
				extSDDiP-I/II-1	7.05	1,861.89	62	4,304
				extSDDiP-I/II-2	9.57	1,829.56	60	4,201
			10	extSDDiP-I/II-0	75.37	1,787.26	10	0
				extSDDiP-I/II-1	71.60	1,877.45	12	0
				extSDDiP-I/II-2	61.54	1,773.12	11	0
			20	CPX	-	-	-	-