European Journal of Operational Research 310 (2023) 268–285

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Stochastics and Statistics

On solving large-scale multistage stochastic optimization problems with a new specialized interior-point approach

Jordi Castro [a,*], Laureano F. Escudero [b], Juan F. Monge [c]

[a] Department of Statistics and Operations Research, Universitat Politècnica de Catalunya, Barcelona, Catalonia
[b] Area of Statistics and Operations Research, Universidad Rey Juan Carlos, URJC, Móstoles Madrid, Spain
[c] Center of Operations Research, Universidad Miguel Hernández, UMH, Elche, Alicante, Spain

ARTICLE INFO

Article history:
Received 19 September 2022
Accepted 30 March 2023
Available online 5 April 2023

Keywords:
Large-scale optimization
Interior-point methods
Stochastic programming
Strategic and operational uncertainties
Two-stage structures

ABSTRACT

A novel approach based on a specialized interior-point method (IPM) is presented for solving large-scale stochastic multistage continuous optimization problems, which represent the uncertainty in strategic multistage and operational two-stage scenario trees. This new solution approach considers a split-variable formulation of the strategic and operational structures. The specialized IPM solves the normal equations by combining Cholesky factorizations with preconditioned conjugate gradients, doing so for, respectively, the constraints of the stochastic formulation and those that equate the split-variables. We show that, for multistage stochastic problems, the preconditioner (i) is a block-diagonal matrix composed of as many shifted tridiagonal matrices as the number of nested strategic-operational two-stage trees, thus allowing the efficient solution of systems of equations; (ii) its complexity in a multistage stochastic problem is equivalent to that of a very large-scale two-stage problem. A broad computational experience is reported for large multistage stochastic supply network design (SND) and revenue management (RM) problems. Some of the most difficult instances of SND had 5 stages, 839 million linear variables, 13 million quadratic variables, 21 million constraints, and 3750 scenario tree nodes; while those of RM had 8 stages, 278 million linear variables, 100 million constraints, and 100,000 scenario tree nodes. For those problems, the proposed approach obtained the solution in 1.1 days using 174 gigabytes of memory for SND, and in 1.7 days using 83 gigabytes for RM; while CPLEX v20.1 required more than 53 days and 531 gigabytes for SND, and more than 19 days and 410 gigabytes for RM.

1. Introduction and motivation

The realization of the uncertain parameters in dynamic mathematical optimization is usually structured in a finite set of scenarios along stages in a given time horizon (Birge & Louveaux, 2011; Pflug & Pichler, 2015). The representation of the uncertain data affects the type of decision models and the decomposition methodologies for problem solving to be dealt with. Therefore, the quality of the solution for the decision making process is also affected by the type of scenario tree generated in stochastic optimization. In dealing with problems within a time horizon (such as capacity expansion planing (CEP) to name one), we undoubtedly must have two types of uncertainties and two types of variables, namely, strategic and operational ones. The strategic variables are related to the decisions on the location, capacity and timing on the infrastructure elements of a system's CEP as supply chain, production system, rapid transit network, energy transmission network and energy generation mix system, to name a few. The operational variables are related to the decisions on the operations of the available elements in the system at the stages along the time horizon. Therefore, there are two types of dynamic optimization submodels, namely, strategic and operational ones which are intrinsically interrelated in a usually large-sized global model for real-life problem solving.

The rationale behind the partition of uncertain parameters into strategic and operational ones, basically, consists of considering that the strategic decisions should not be based on *individual* operational ones at the stages; see Escudero & Monge (2018). However, strategic decisions should depend on the realizations of the strategic uncertain parameters as well as on the set of realizations of the operational parameters *as a whole* in the stage and successors. Therefore, that observation is translated into

* Corresponding author.
E-mail addresses: jordi.castro@upc.edu (J. Castro), laureano.escudero@urjc.es (L.F. Escudero), monge@umh.es (J.F. Monge).

https://doi.org/10.1016/j.ejor.2023.03.042

considering that the strategic nodes in the scenario tree should *not* be successors of individual operational nodes. (An additional reason is the gigantic stochastic model that would result in the hypothetical case where the strategic and operational features are not taken into independent consideration; note that the uncertainty would be represented in a multistage scenario tree, where the nodes represent a mixture of the strategic and operational uncertain parameters; see Escudero & Monge (2021).) Note that the operational uncertainty can be represented in a two-stage tree, where the second stage nodes have one-to-one correspondence with the operational scenarios in the stage. The root nodes of those trees are precisely the strategic nodes in the stage. The above approach has been considered in works for different industrial sectors as production energy planning (Kaut et al., 2014; Werner et al., 2013), rapid transit network design (Cadarso et al., 2018), dynamic forest stand harvesting selection planning (Alonso-Ayuso et al., 2020) and, recently, hub network expansion planning (Escudero & Monge, 2021), among few others. In a different context, see a strong multistage multiscale-based stochastic formulation in Glanzer & Pflug (2020). A scheme for obtaining lower and upper bounds on this type of stochastic problems is presented in Maggioni et al. (2020).

Given the large sizes of real-life instances of many strategic optimization problems, and the current limitations of hardware/software resources, the only alternative for their efficient solution is to consider decomposition algorithms. See Escudero et al. (2017) for a comprehensive overview of important types of those algorithms. This work presents a new alternative interior-point method (IPM) (Gondzio, 2012; Wright, 1997) for multistage stochastic optimization problems, and it is based on a nested splitting formulation for the *step variables* (i.e., state strategic variables that link a strategic node to only its immediate successors in the multistage stochastic tree). The splitting formulation for the solution of stochastic problems is not new; as far as we know, it was introduced in Lustig et al. (1991) for two-stage stochastic problems. The purpose of that splitting was to avoid the constraint matrix having dense columns, which are known to be a drawback for IPMs. Other practical split-variable reformulations of multistage stochastic optimization for IPMs are presented in Ruszczyński (1993), but without reporting any computational evidence. Steinbach (2001) extends the IPM proposed in an earlier work based on KKT structures to exploit the multistage scenario tree sparsity. Limited computational experience is reported. Blomvall & Lindberg (2002) consider a primal barrier IPM for solving a stochastic multistage convex nonlinear programming (NLP) problem, where a dynamic programming approach is embedded for solving the subproblems in the multistage scenario tree. Gondzio & Grothey (2007) present an approach for solving large-scale stochastic multistage NLP problems by considering the primal-dual IPM OOPS (Object-Oriented Parallel Solver), introduced in an earlier work. The IPM-based primal-dual column generation approach introduced in Gondzio et al. (2016) efficiently solves large instances of two-stage stochastic optimization problems, but it is not easily generalized to the multistage case. This same set of two-stage instances was also solved in Tanneau et al. (2021) with a Dantzig–Wolfe decomposition using the Tulip interior-point academic code. However, extending this approach to multistage stochastic problems is also not straightforward. In addition, the dimensions of the two-stage instances tested in those two previous references using IPMs are much smaller than the ones of the multistage problems solved in this work. Hübner et al. (2017) present a IPM for large-scale stochastic multistage NLP problems by exploiting its KKT structures in the subproblems, each with a moderate number of variables, in a depth-first distribution of the nodes in the scenario tree over the hardware threads. Another approach that relies on IPMs is the dual decomposition implemented in the DSP (Decomposition for Structured Programming) stochastic solver (Kibaek & Zavala, 2018), which includes an interior-point cutting-plane generator. DSP, however, is not competitive for large-scale problems.

In this new proposal, the nested strategic-operational two-stage trees are rooted at the strategic nodes of the multistage stochastic tree. The form in which those structures are represented impacts the constraints of the model. This new solution approach considers a split-variable formulation for the step variables in the first stage nodes of the strategic and operational structures. Therefore, each variable in any strategic node has copies, which are related to the first stage of both types of interlinked two-stage subproblems. The first copy is related to the strategic two-stage subproblem and the second to the operational one. The efficiency of the new approach relies on exploiting the primal block-angular structure of the resulting split-variable reformulation of the multistage stochastic problem. This is done by means of the specialized IPM, which was initially introduced in Castro (2000) for multicommodity flows, and later extended to other classes of primal block-angular problems (Castro, 2007; Castro & Cuesta, 2011). This algorithm —which was implemented in a package named BlockIP (Castro, 2016)— solves the normal equations associated with the Newton direction of the IPM by combining Cholesky factorizations with a preconditioned conjugate gradient (PCG). The main contributions of this work are:

(i) A two-stage split-variable feature as embedded in the deterministic equivalent model for the multistage multiscaling stochastic problem. Then, the proposal links the split variables of a strategic node through the strategic immediate successor nodes jointly with the operational ones of that strategic node in the multistage tree. For that purpose, a circular scheme is used for the classical nonanticipativity constraints that has to be satisfied "if two different scenarios s and s' are indistinguishable at time t on the basis of information available about them at time t" (Rockafellar & Wets, 1991); see also Birge (1985). That way the dense columns in the model are avoided, among other advantages.

(ii) A particular form of the preconditioner is introduced for multistage stochastic optimization problems, so that the linear systems of equations can be efficiently solved at each iteration. As it will be shown, for any multistage stochastic optimization problem, with both strategic and operational decisions, the proposed preconditioner is block diagonal and each block is a ν-shifted tridiagonal matrix, where the number of blocks is the number of nested two-stage subtrees, and ν is the number of variables replicated in the related two-stage tree.

(iii) One of the significant features of the proposal is that the complexity of the preconditioner for a multistage stochastic problem does not grow in comparison with that of a two-stage problem. This is in contrast to alternative methods, in which solving a multistage problem is significantly more difficult than solving a two-stage one.

The overall approach has been implemented in a package named MSSO-BlockIP (MultiStage Stochastic Optimization based on BlockIP). As it will be seen in the computational results, and in terms of both CPU time and required gigabytes of memory, MSSO-BlockIP outperformed even the primal-dual barrier method implemented in the state-of-the-art solver CPLEX v20.1 for multistage stochastic optimization problems with up to 839 million variables, up to 100 million constraints, and more than 100,000 nodes in scenario trees with 8 stages. This new approach requires far fewer computational resources for huge problems, thereby also significantly reducing energy consumption and, thus CO_2, emissions. We note that a simpler variant of this approach has already proven to be efficient for two-stage stochastic optimization

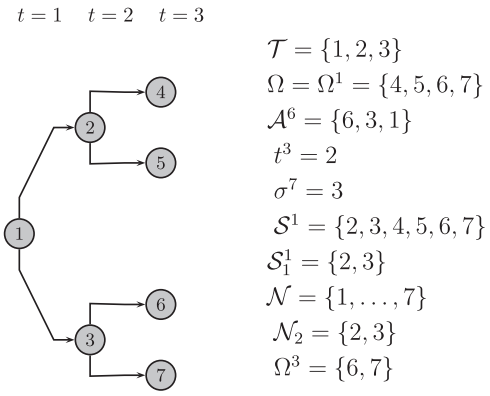

Fig. 1. Strategic multistage scenario tree.

problems (Castro & de la Lama-Zubirán, 2020); and that algorithm is indeed a particular case of the one introduced in this work, namely, when the number of stages is only two and there are no operational decisions.

The rest of the work is organized as follows. For completeness and to introduce some notations to be used throughout the work, Section 2 outlines the main concepts underlying strategic multistage stochastic trees that have operational two-stage trees embedded in them. Section 3 presents the two multistage stochastic metamodels used in this work. The first model is presented in compact form and the second in its split-variable formulation, which is more amenable to IPMs. Section 4 introduces the specialized IPM for multistage stochastic problems. Its implementation, MSSO-BlockIP, is presented in Section 5, which also reports the results of computational experiments comparing our approach with CPLEX v20.1 in the solution of two different applications—described in Appendices Appendix A and Appendix B—namely, for strategic and operational supply network design, and strategic revenue management. Section 6 draws the main conclusions and outlines future research plans.

2. Strategic multistage operational two-stage stochastic trees

The notation is taken from Escudero & Monge (2021).

2.1. Strategic multistage stochastic tree

Let a *strategic* scenario be the realization of the uncertain strategic parameters along the time horizon. A *strategic node* for a given stage has one-to-one correspondence with the group of strategic scenarios that have the same realization of the uncertain parameters up to the stage. This information structure can be visualized as the tree depicted in Fig. 1, where each root-to-leaf path represents a specific scenario and, then, it corresponds to a realization of the whole set of the uncertain parameters. Let us point out that it is beyond the scope of this work to present a methodology for multistage scenario tree generation and reduction; see e.g., Dupacova et al. (2000); Heitsch & Römisch (2009); Henrion & Römisch (2018); Hoyland et al. (2003); Leövey & Römisch (2015); Li & Floudas (2016); Pflug & Pichler (2014), among others.

Lexicographically ordered sets in the strategic tree

$\mathcal{T}$, stages.

$\mathcal{N}$, nodes in the scenario tree.

$\mathcal{N}_t$, nodes in stage t, where $\mathcal{N}_t \subset \mathcal{N}$, $t \in \mathcal{T}$. By construction, $|\mathcal{N}_1| = 1$.

Ω, scenarios. Each one comprises the nodes in the Hamiltonian path from root node 1 to a node, say, ω in the last stage, up through the stages in set $\mathcal{T}$; therefore, $\omega \in \mathcal{N}_{|\mathcal{T}|}$. For convenience, a scenario has traditionally been denoted by its last node in the path.

$\Omega^n \subset \Omega$, scenarios containing node n in the path from root node 1 to their last node $\omega \in \mathcal{N}_{|\mathcal{T}|}$. Note that $\Omega^1 = \Omega$.

$\mathcal{A}^n$, node n and its ancestors, $n \in \mathcal{N}$. Note that $\mathcal{A}^1$ only contains node $1 \in \mathcal{N}_1$.

$\mathcal{S}^n$, successors of node n, $n \in \mathcal{N}$. Note that $\mathcal{S}^n = \emptyset$, $n \in \mathcal{N}_{|\mathcal{T}|}$; and $\mathcal{S}^1 = \mathcal{N} \setminus \{1\}$.

$\mathcal{S}_1^n \subset \mathcal{S}^n$, immediate successors of node n, $n \in \mathcal{N}$.

Other elements in strategic node n for $n \in \mathcal{N}$

w^n, weight factor representing the likelihood that is associated with node n. Note that $w^n = \sum_{\omega \in \Omega^n} w^\omega$, where w^ω gives the modeler-driven likelihood associated with scenario ω, such that $\sum_{\omega \in \Omega} w^\omega = 1$.

t^n, stage that node n belongs to, therefore, $n \in \mathcal{N}_{t^n}$.

σ^n, immediate ancestor node of node n.
Note: It is assumed that $\sigma^1 = 0$.

$s^n(i)$, ith node in set $\mathcal{S}_1^n : t^n < |\mathcal{T}|$, $i = 1, \ldots, \ell^n$, where $\ell^n = |\mathcal{S}_1^n|$.

As an illustration, let us consider an instance with $|\mathcal{T}| = 3$ stages (say, years) and a scenario tree where the number of strategic immediate successor nodes of node n is $\ell^n = 2$, $n \in \mathcal{N} : t^n < |\mathcal{T}|$. Therefore, the cardinality of the strategic scenario tree is $|\mathcal{N}| = \sum_{t \in \mathcal{T}} |\mathcal{N}_t| = 1 + 2 + 4 = 2^3 - 1 = 7$ nodes, see Fig. 1.

2.2. Operational uncertainty in the stages in set $\mathcal{T}$ along the time horizon

The operational uncertainty is represented in a finite set of stage-dependent operational scenarios in each stage t, $t \in \mathcal{T}$. It is thus assumed that the operational uncertainty originates in any of the previous or current stage—independent of the strategic uncertainty. Let us introduce the following additional notation:

Π_t, set of operational scenarios in stage t.

w^π, weight of operational scenario π, $\pi \in \Pi_t$, such that $\sum_{\pi \in \Pi_t} w^\pi = 1$.

$\pi_t(i)$, ith operational node in set Π_t, $i = 1, \ldots, |\Pi_t|$.

A solution approach for a strategic multistage stochastic problem with stage-related uncertainty requires that the operational decisions for a given strategic realization are structured in a two-stage stochastic tree at any stage. The first stage is made of the appropriate strategic node and the second stage is composed of the operational scenarios. As an illustration, Fig. 2 depicts a scenario tree with the same strategic node set as in Fig. 1, plus a set of operational scenarios (i.e., it is a multistage multiscale scenario tree), where $|\Pi_t| = 2$, $\forall t \in \mathcal{T}$.

Note that in the unlikely case where the strategic nodes are also stagewise-dependent on the operational ones, the tree depicted in Fig. 2 will instead result in a gigantic multistage scenario tree containing the full combination of strategic and operational scenarios. As an illustration, a joint multistage scenario tree for an instance with $|T| = 5$ and $\ell^n = 2$ for $t^n \in \mathcal{T} : t^n < 5$ has 23,405 nodes and 16,384 scenarios for $|\Pi_t| = 4$; and 629,145 nodes and 528,288 scenarios for $|\Pi_t| = 8\,\forall t \in \mathcal{T}$; see Escudero & Monge (2021).

3. Strategic multistage operational two-stage stochastic metamodels

The compact version of the strategic multistage operational two-stage metamodel can be expressed as

$$\min \quad \sum_{n\in\mathcal{N}} w^n\Big[a^n x^n + \frac{1}{2}x^{n\top}Q_x^n x^n + b^n z^n + \frac{1}{2}z^{n\top}Q_z^n z^n + \sum_{\pi\in\Pi_{t^n}}\Big(w^\pi c^\pi y_n^\pi + \frac{1}{2}y_n^{\pi\top}Q^\pi y_n^\pi\Big)\Big] \tag{1a}$$

$$\text{s. to} \quad (T^n x^{\sigma^n})_{:t^n>1} + W^n x^n + M^n z^n = h^n \qquad \forall n\in\mathcal{N} \tag{1b}$$

$$T^\pi x^n + W^\pi y_n^\pi = h^\pi \qquad \forall \pi\in\Pi_{t^n},\ n\in\mathcal{N} \tag{1c}$$

$$0 \le x^n \le u_x^n,\ 0 \le z^n \le u_z^n \qquad \forall n\in\mathcal{N} \tag{1d}$$

$$0 \le y_n^\pi \le u_y^\pi \qquad \forall \pi\in\Pi_{t^n},\ n\in\mathcal{N}, \tag{1e}$$

where the new parameters are as follows: a^n, b^n, c^π and Q_x^n, Q_z^n, Q^π are the vectors and (positive semidefinite) matrices of the linear and quadratic terms of the objective function for the variables x^n, z^n and y_n^π, respectively; T^n and W^n, respectively, are the constraint matrices of the state strategic variables x^{σ^n} in the first stage and x^n in the second one (both in the related embedded strategic two-stage submodel); M^n is the constraint matrix of the local strategic variables z^n in the first stage strategic node n in the related embedded strategic two-stage submodel; T^π and W^π, respectively, are the constraint matrices of the state strategic variables x^n in the first stage and the operational variables y_n^π in the second one of the embedded operational two-stage submodels; h^n and h^π are the right-hand-side (RHS) of the two-stage strategic and operational constraints, respectively; and u_x^n, u_z^n and u_y^π are the upper bounds of the variables in the vectors x^n, z^n and y_n^π, respectively. Figure 3 shows the structure of the constraint matrix for metamodel (1) in the Fig. 2 example, which we have simplified by omitting the columns related to the z^n variables (i.e., terms $M^n z^n$).

To overcome the existence of dense columns in (1) (e.g., x^i, $i = 1, 2, \ldots$, in Fig. 3), split-variable reformulations are required when the models are solved by IPMs (Castro & de la Lama-Zubirán, 2020; Lustig et al., 1991; Ruszczyński, 1993). Our approach considers that formulation by using the following copies of the variables:

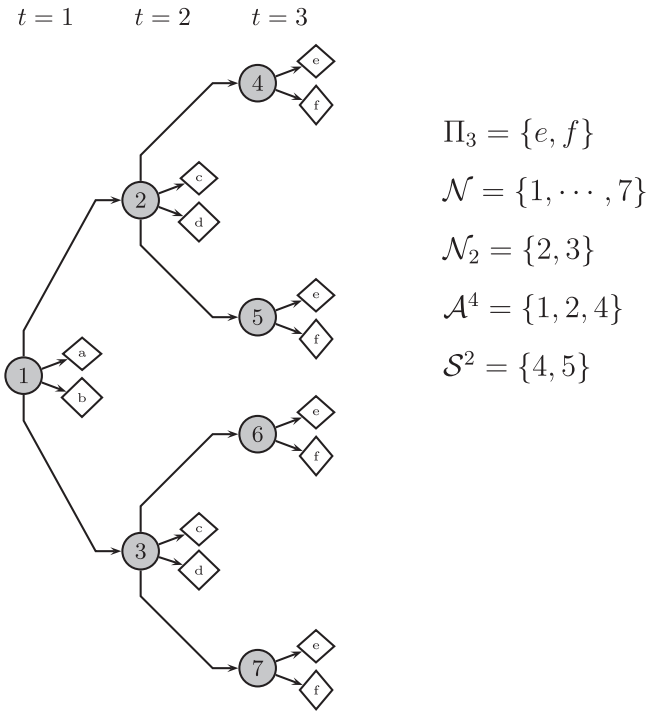

Fig. 2. Strategic multistage scenario tree with operational two-stage scenario trees.

- x_n^s, copy of x^n in strategic node s, where n is the strategic node that roots the strategic two-stage tree and s is a second stage node, for $s\in\mathcal{S}_1^n$, $n\in\mathcal{N} : t^n < |\mathcal{T}|$.
- x_n^π, copy of x^n in operational node π, where n is the strategic node that roots the operational two-stage tree and $\pi\in\Pi_{t^n}$.

The copies of those variables above are forced to have the same value through a set of linking constraints. To simplify notation, let us drop the superindex n in $s^n(i)$ and ℓ^n, $n\in\mathcal{N}$, and the subindex t in $\pi_t(i)$, $t\in\mathcal{T}$, when no ambiguity exists in the context being studied. Therefore, for the strategic two-stage tree we impose $x^n - x_n^{s(1)} = 0$ and $x_n^{s(i)} - x_n^{s(i+1)} = 0$, $i = 1, \ldots, \ell^n - 1$, for

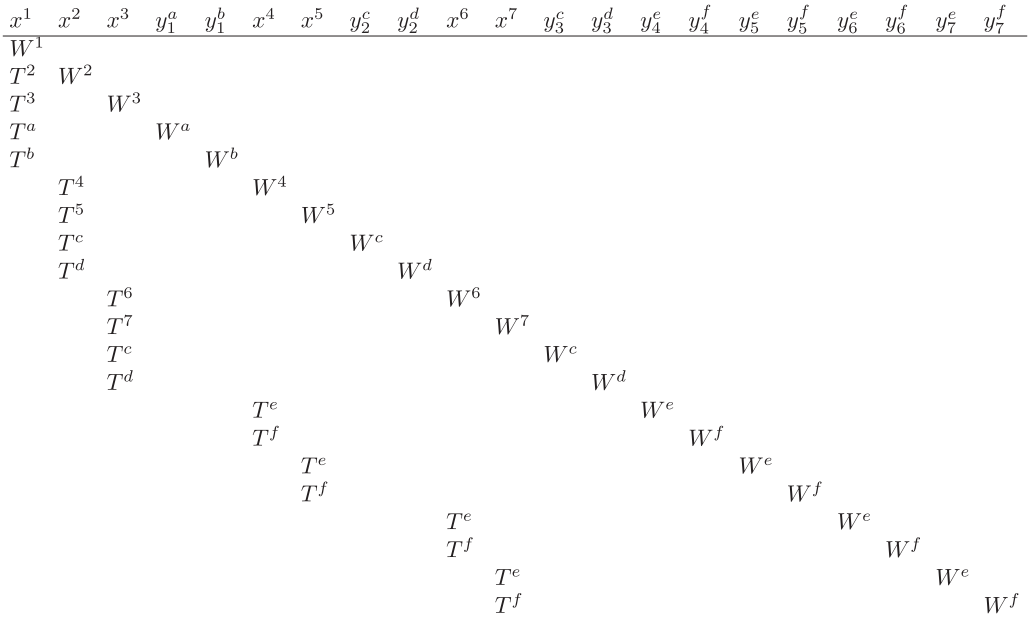

Fig. 3. Constraint matrix of metamodel (1) for the example in Fig. 2, omitting columns z^n, $n\in\mathcal{N}$.

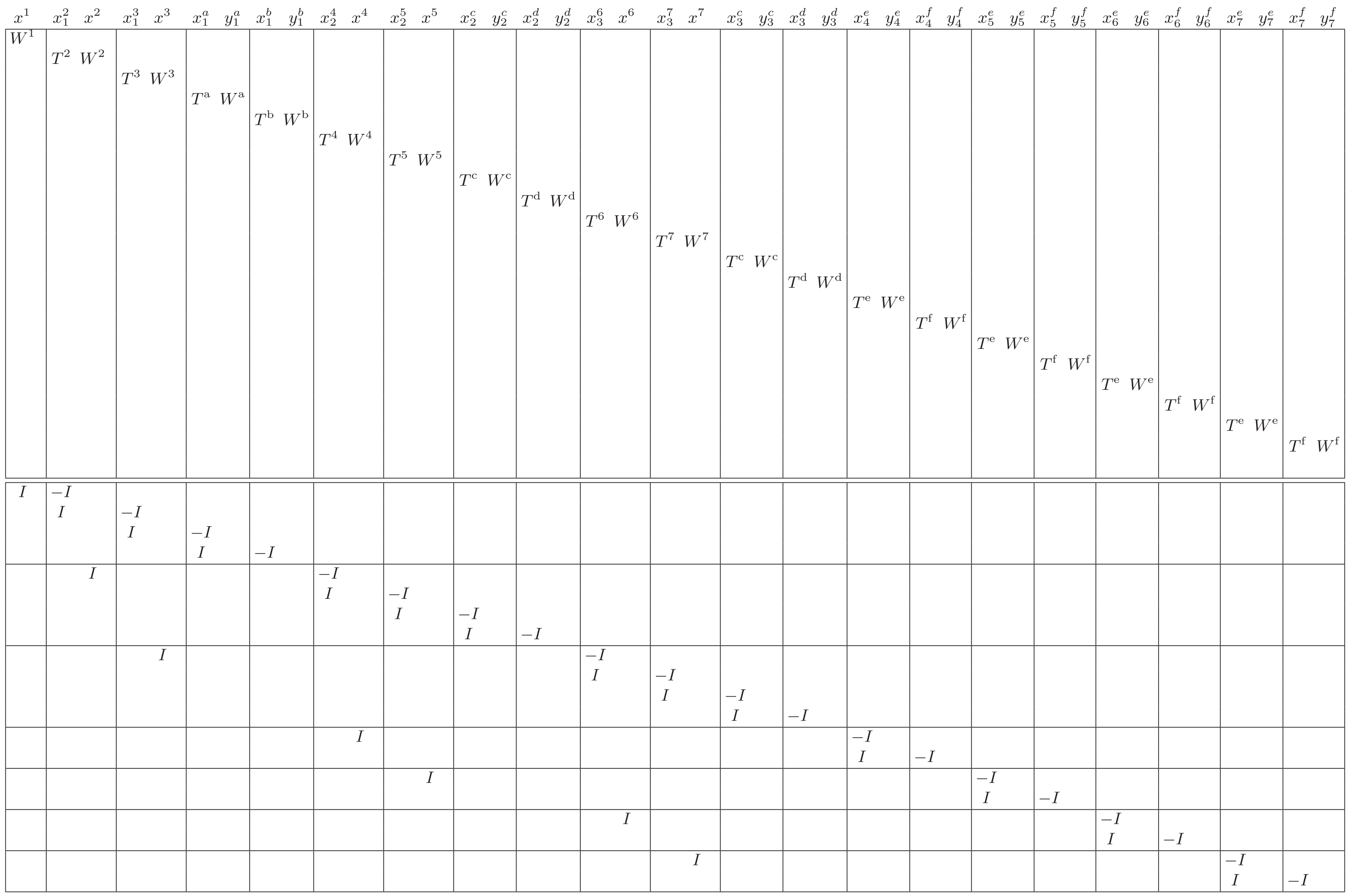

Fig. 4. Constraint matrix of meta formulation (2) for the example in Fig. 2, omitting columns z^n.

each node n that is not in the last stage (that is, $n \in \mathcal{N} : t^n < |\mathcal{T}|$). And for the operational two-stage tree, we have for any node $n \in \mathcal{N}$: $x_n^{s(\ell)} - x_n^{\pi(1)} = 0$ if $t^n < |\mathcal{T}|$, or $x^n - x_n^{\pi(1)} = 0$ if $t^n = |\mathcal{T}|$; and $x_n^{\pi(i)} - x_n^{\pi(i+1)} = 0$, $i = 1, \dots, |\Pi_{t^n}| - 1$. Our split-variable scheme first covers the entire set of strategic immediate successors of node n and, next, all the operational scenarios for any strategic node. It is worth pointing out that $x_n^{s(\ell)} - x_n^{\pi(1)} = 0$ could be replaced by $x_n - x_n^{\pi(1)} = 0$, among other alternatives. However, the chosen scheme has the advantage that the variables in set $\{x^n, x_n^{s(1)}, \dots, x_n^{s(\ell)}, x_n^{\pi(1)}, \dots, x_n^{|\Pi_{t^n}|}\}$ simply appear in two linking constraints, with the exception of x^n and $x_n^{|\Pi_{t^n}|}$ which appear in only one (see Fig. 4 for an example).

Thus, the split-variable formulation of metamodel (1) considered by our approach is:

$$\min \quad \sum_{n \in \mathcal{N}} w^n \Big[a^n x^n + \frac{1}{2} x^{n\top} Q_x^n x^n + b^n z^n + \frac{1}{2} z^{n\top} Q_z^n z^n + \sum_{\pi \in \Pi_{t^n}} \Big(w^\pi c^\pi y_n^\pi + \frac{1}{2} y_n^{\pi\top} Q^\pi y_n^\pi \Big) \Big] \tag{2a}$$

$$\text{s. to} \quad (T^n x_{\sigma^n}^n)_{:t^n>1} + W^n x^n + M^n z^n = h^n \qquad \forall n \in \mathcal{N} \tag{2b}$$

$$T^\pi x_n^\pi + W^\pi y_n^\pi = h^\pi \qquad \forall \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{2c}$$

$$x^n - x_n^{s(1)} = 0,\ x_n^{s(i)} - x_n^{s(i+1)} = 0 \quad \forall i = 1, \dots, \ell^n - 1,\ n \in \mathcal{N} : t^n < |\mathcal{T}| \tag{2d}$$

$$x_n^{\bullet} - x_n^{\pi(1)} = 0,\ x_n^{\pi(i)} - x_n^{\pi(i+1)} = 0 \quad \forall i = 1, \dots, |\Pi_{t^n}| - 1,\ n \in \mathcal{N} \tag{2e}$$

$$0 \le x^n \le u_x^n,\ 0 \le z^n \le u_z^n \qquad \forall n \in \mathcal{N} \tag{2f}$$

$$0 \le y_n^\pi \le u_y^\pi \qquad \forall \pi \in \Pi_{t^n},\ n \in \mathcal{N}, \tag{2g}$$

where $x_n^{\bullet} \equiv x_n^{s(\ell)}$ for $n \in \mathcal{N} : t^n < |\mathcal{T}|$ and $x_n^{\bullet} \equiv x^n$ for $n \in \mathcal{N}_{|\mathcal{T}|}$. Constraints (2d) and (2e) are the linking equations for the split-variables that are used in, respectively, the strategic and operational nodes (that is, in (2b) and (2c)). Notice that those linking equations are given in a circular form of the classical nonanticipativity constraints which our IPM will take benefit from. Fig. 4 depicts the structure of the constraint matrix of the meta formulation (2) for the same example in Fig. 3 for metamodel (1), omitting columns z^n to save space.

4. The specialized IPM for multistage stochastic optimization

In the rest of the work, and due to notation saving purposes, let variables vector x replace the notation x, y, z that have been used in metamodels (1) and (2). Let us consider the following general

formulation of a primal block-angular optimization problem:

$$\begin{aligned} \min_{x^1,\ldots,x^k,x^0} \quad & \sum_{i=1}^{k} \left({c^i}^\top x^i + {x^i}^\top Q^i x^i \right) \\ \text{s. to} \quad & \begin{bmatrix} N_1 & & & & \\ & N_2 & & & \\ & & \ddots & & \\ & & & N_k & \\ R_1 & R_2 & \ldots & R_k & I \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^k \\ x^0 \end{bmatrix} = \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^k \\ b^0 \end{bmatrix} \\ & 0 \le x^i \le u^i \quad i = 0, \ldots, k. \end{aligned} \tag{3}$$

The matrices $N_i \in \mathbb{R}^{\overline{m}_i \times \overline{n}_i}$ and $R_i \in \mathbb{R}^{l \times \overline{n}_i}$, $i = 1, \ldots, k$, define, respectively, the block and linking constraints, where k is the number of blocks, l is the number of linking constraints, and $\overline{m}_i$ and $\overline{n}_i$ denote the number of constraints and variables of block i. It will be seen below that for MSSO problems the number of blocks k is the number of strategic and operational nodes in the multistage scenario tree, and that matrices N_i and R_i are related to constraints (2b), (2c) and (2d), (2e), respectively. Vectors $x^i \in \mathbb{R}^{\overline{n}_i}$ and $u^i \in \mathbb{R}^{\overline{n}_i}$ are the variables and their upper bounds, respectively, for block $i = 1, \ldots, k$. The components of $x^0 \in \mathbb{R}^l$ are the slacks of the linking constraints; if they are equalities their upper bounds $u^0 \in \mathbb{R}^l$ can be set to 0 or to a very small feasibility tolerance. The objective function considers both linear and convex quadratic separable costs, as defined by vectors $c^i \in \mathbb{R}^{\overline{n}_i}$ and matrices $Q^i \in \mathbb{R}^{\overline{n}_i \times \overline{n}_i}$ ($Q^i = {Q^i}^\top$ and $Q^i \succeq 0$), $i = 1, \ldots, k$. The vectors $b^i \in \mathbb{R}^{\overline{m}_i}$, $i = 1, \ldots, k$ and $b^0 \in \mathbb{R}^l$ define, respectively, the RHS of the block and the linking constraints.

By using an appropriate reordering of variables and constraints, any multistage stochastic optimization problem (2) can be recast as a primal block-angular problem (3). This is illustrated in Fig. 4 for the particular case of the scenario tree in Fig. 2. Note that variable vectors z^n, $n \in \mathcal{N}$ are omitted to save space, but, this does not affect the matrix structure depicted in Fig. 4 because they are local to each node and are therefore not replicated. The order of the variables is based on a breadth-first-search (BFS) in the scenario tree (that is, nodes are explored by stages). And for each node n in the BFS, the first variables considered are those involved in the (nested) strategic two-stage tree composed of node n and its immediate successors $s \in \mathcal{S}_1^n$ (namely, variable vectors x^n, all the copies x_n^s, the local strategic variables vector z^s, and the state strategic ones x^s), followed by the variables in the operational two-stage tree comprising the nodes n and $\pi \in \Pi_{t^n}$ (that is, variable vectors x_n^π and y_n^π). As for the constraints, (2b) and (2c) are moved to the first rows (block constraints in (3)), while (2d) and (2e) correspond to the linking constraints. Observe that with this particular ordering the two groups of (block and linking) constraints have a stair-type form. In Fig. 4 vertical lines separate the k blocks of the problem; a double horizontal line separates the diagonal block constraints $N_i x^i = b^i$, $i = 1, \ldots, k$, from the linking constraints; and, within the linking constraints part, a horizontal line separates the constraints of different (nested) two-stage trees. For instance, the linking constraints in Fig. 4 are partitioned into seven groups, which correspond to the seven (nested) two-stage trees in Fig. 2, each one composed of the scenario root node n, followed by the first strategic node $s^n(1)$ until the last node in $\mathcal{S}_1^n$, and trailed by the first operational node $\pi(1)$ until the last node in Π_{t^n}, namely, $\{1, 2, 3, 1^a, 1^b\}$, $\{2, 4, 5, 2^c, 2^d\}$, $\{3, 6, 7, 3^c, 3^d\}$, $\{4, 4^e, 4^f\}$, $\{5, 5^e, 5^f\}$, $\{6, 6^e, 6^f\}$, $\{7, 7^e, 7^f\}$. Note also that the number of blocks k is equal to the number of nodes in the multistage scenario tree, that is, $k = \sum_{n \in \mathcal{N}} (1 + |\Pi_{t^n}|)$. For the particular case of a problem where $|\Pi| = |\Pi_t|$ for all $t \in \mathcal{T}$, and $\ell = |\mathcal{S}_1^n|$ for all $n \in \mathcal{N} : t^n < |\mathcal{T}|$, the number of blocks can be computed as $k = \frac{\ell^{|\mathcal{T}|} - 1}{\ell - 1} (|\Pi| + 1)$ (e.g., in Fig. 4 we have $\ell = 2$, $|\mathcal{T}| = 3$ and $|\Pi| = 2$, so $k = 21$, which is the number of (strategic and operational) nodes in the tree in Fig. 2). It is worth pointing out the high sparsity of the matrices $R_i \in \mathbb{R}^{l \times \overline{n}_i}$, where all coefficients are zero, except for at most three diagonal (identity) matrices I or $-I$ (see e.g., Fig. 4). This high sparsity will be crucial for the efficient solution of problem (3). We also remark that (3) can deal with more general MSSO models than those of formulation (2), for instance, for problems where the set of operational scenarios is different for each node of the same stage, that is, we have $\Pi^n, n \in \mathcal{N}$ instead of $\Pi_t, t \in \mathcal{T}$. This is a case in which the operational uncertainty is stagewise-dependent (as oppose to stage-dependent), but the strategic uncertainty continues to be non-dependent of individual realizations of the operational uncertainty.

The previous discussion can be summarized in the following result:

Proposition 1. *Any multistage stochastic optimization problem that has both strategic and operational decisions, and is based on the split-variable formulation (2) can be recast as a primal block-angular problem (3).*

Proof. Using the following reordering of variables and constraints, problem (2) is reformulated as (3):

The ordering of variables is based on a BFS in the scenario tree (that is, nodes are explored by stages). For each node n in the BFS, the first variables are those in the strategic two-stage tree composed of node n and its immediate successors $s \in \mathcal{S}_1^n$, followed by the variables in the operational two-stage tree comprising the nodes n and $\pi \in \Pi_{t^n}$. That is, for each node n in the BFS, the order of variables is: variable vectors x^n, all the copies x_n^s, the local strategic variables vector z^s, the state strategic ones x^s, and variable vectors x_n^π and y_n^π of operational nodes.

Constraints (2b) and (2c) are moved to the first rows (block constraints in (3)), ordered by node n in the BFS. Constraints (2d) and (2e) form the linking constraints, and they are ordered, firstly by each two-stage tree in the scenario tree, and, secondly, using the BFS order of the nodes within each two-stage tree. □

Problems in the form of (3) can be solved by the specialized IPM of Castro (2000, 2007); Castro & Cuesta (2011), which was recently implemented in the BlockIP package (Castro, 2016). This approach is based on an infeasible long-step primal-dual path-following method (Wright, 1997) that solves the normal equations at each IPM iteration by exploiting the particular structure of the constraint matrix of (3).

For completeness, we summarize the primal-dual path-following IPM. Problem (3) can be written in standard form as

$$\begin{aligned} \min_x \quad & c^\top x + \tfrac{1}{2} x^\top Q x \\ \text{s. to} \quad & Ax = b \\ & 0 \le x \le u, \end{aligned} \tag{4}$$

where $c, x, u \in \mathbb{R}^{\overline{n}}$, $A \in \mathbb{R}^{\overline{m} \times \overline{n}}$, $Q \in \mathbb{R}^{\overline{n} \times \overline{n}}$ and $b \in \mathbb{R}^{\overline{m}}$, $\overline{m}$ and $\overline{n}$, respectively, being the overall number of constraints and variables of the problem (in our context, $\overline{n} = \sum_{i=1}^k \overline{n}_i + l$, $\overline{m} = \sum_{i=1}^k \overline{m}_i + l$).

The dual problem of (4) is

$$\begin{aligned} \max \quad & b^\top \lambda - \tfrac{1}{2} x^\top Q x - \lambda_u^\top u \\ \text{s. to} \quad & A^\top \lambda - Qx + \lambda_0 - \lambda_u = c \\ & \lambda_0, \lambda_u \ge 0, \end{aligned} \tag{5}$$

where $\lambda \in \mathbb{R}^{\overline{m}}$, λ_0 and $\lambda_u \in \mathbb{R}^{\overline{n}}$ are, respectively, the Lagrange multipliers of the equality constraints and the lower and upper bounds of (4).

Replacing the bounds in (4) with a logarithmic barrier having parameter $\mu \in \mathbb{R}^+$, the μ-perturbed version of the KKT conditions

of (4) becomes

$$r_c \equiv c - (A^\top \lambda - Qx + \lambda_0 - \lambda_u) = 0, \tag{6a}$$

$$r_b \equiv b - Ax = 0, \tag{6b}$$

$$r_{x\lambda_0} \equiv \mu e - X\Lambda_0 e = 0, \tag{6c}$$

$$r_{s\lambda_u} \equiv \mu e - S\Lambda_u e = 0, \tag{6d}$$

$$(x, s, \lambda_0, \lambda_u) \geq 0\ , \tag{6e}$$

where $e \in \mathbb{R}^{\bar{n}}$ is a vector of ones, $X, U, \Lambda_0, \Lambda_u \in \mathbb{R}^{\bar{n}\times\bar{n}}$ are diagonal matrices made up of, respectively, vectors $x, u, \lambda_0, \lambda_u$, and S is defined as $S = U - X$. Equations (6a) and (6b) impose, respectively, dual and primal feasibility, whereas (6c) and (6d) impose (μ-perturbed) complementarity. The set of unique solutions from (6) for each μ value is known as the central path. When $\mu \to 0$, these solutions converge to those of (4) and (5). The primal-dual path-following algorithm solves the nonlinear system (6) by a sequence of damped Newton directions (that is, with step length reduction to preserve (6e)), reducing the μ parameter at each iteration, and staying close to the central path. The monographs Gondzio (2012); Wright (1997) provide an excellent discussion about primal-dual path-following algorithms.

The Newton direction $(\Delta x, \Delta\lambda, \Delta\lambda_0, \Delta\lambda_u)$ is obtained by the solution of the system

$$\begin{bmatrix} -Q & A^\top & I & \\ A & & & \\ \Lambda_0 & & X & \\ -\Lambda_u & & & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\lambda_0 \\ \Delta\lambda_u \end{bmatrix} = \begin{bmatrix} r_c \\ r_b \\ r_{x\lambda_0} \\ r_{s\lambda_u} \end{bmatrix}. \tag{7}$$

By eliminating $\Delta\lambda_u$ and $\Delta\lambda_0$ in (7), as follows,

$$\Delta\lambda_0 = X^{-1} r_{x\lambda_0} - X^{-1}\Lambda_0 \Delta x \tag{8a}$$

$$\Delta\lambda_u = S^{-1} r_{s\lambda_u} + S^{-1}\Lambda_u \Delta x, \tag{8b}$$

we obtain a symmetric indefinite system known as the *augmented system*:

$$\begin{bmatrix} -\Theta^{-1} & A^\top \\ A & \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} r \\ r_b \end{bmatrix}, \tag{9}$$

where Θ and r are defined as

$$\Theta = (Q + S^{-1}\Lambda_u + X^{-1}\Lambda_0)^{-1} \qquad r = r_c + S^{-1} r_{s\lambda_u} - X^{-1} r_{x\lambda_0}. \tag{10}$$

If, in addition, we eliminate Δx from the last group of equations in (9), the *normal equations* form is obtained:

$$(A\Theta A^\top)\Delta\lambda = g \qquad \text{where } g = r_b + A\Theta r \tag{11a}$$

$$\Delta x = \Theta(A^\top \Delta\lambda - r). \tag{11b}$$

The Newton direction is computed from (8a), (8b), (11a) and (11b). For linear (i.e., $Q = 0$) or separable quadratic problems Θ is a positive diagonal matrix and can be easily computed and inverted.

Computationally, the most time-consuming step of the algorithm is solving system (11a) at each iteration of the IPM. An efficient solution approach for this system is needed mainly for large-scale problems such as the multistage stochastic optimization models (1) and (2), which can easily reach millions of variables and constraints even for scenario trees with a small number of stages (see Section 5). The IPM specialization used in this work solves the normal equations by exploiting the structure of matrix A in (3). Appropriately partitioning Θ, the matrix $A\Theta A^\top$ of the normal equations can be recast as

$$A\Theta A^\top = \begin{bmatrix} N_1\Theta_1 N_1^\top & & & N_1\Theta_1 R_1^\top \\ & \ddots & & \vdots \\ & & N_k\Theta_k N_k^\top & N_k\Theta_k R_k^\top \\ R_1\Theta_1 N_1^\top & \dots & R_k\Theta_k N_k^\top & \Theta_0 + \sum_{i=1}^k R_i\Theta_i R_i^\top \end{bmatrix} = \begin{bmatrix} B & C \\ C^\top & E \end{bmatrix}, \tag{12}$$

where $B \in \mathbb{R}^{\tilde{m}\times\tilde{m}}$ $(\tilde{m} = \sum_{i=1}^k \bar{m}_i)$, $C \in \mathbb{R}^{\tilde{m}\times l}$, and $E \in \mathbb{R}^{l\times l}$ are the blocks of $A\Theta A^\top$; and Θ_i, $i = 0, \dots, k$, are the submatrices of Θ associated with the $k+1$ groups of variables $(x^0, x^1, \dots, x^k)$ in (3). Considering a partitioning of the RHS of (11a) $g = [g_1^\top \ g_2^\top]^\top$, where $g_1 \in \mathbb{R}^{\tilde{m}}$ and $g_2 \in \mathbb{R}^l$, and the direction of Lagrange multipliers λ, $\Delta\lambda = [\Delta\lambda_1^\top \ \Delta\lambda_2^\top]^\top$, with $\Delta\lambda_1 \in \mathbb{R}^{\tilde{m}}$ and $\Delta\lambda_2 \in \mathbb{R}^l$, the normal equations can be written as

$$\begin{bmatrix} B & C \\ C^T & E \end{bmatrix} \begin{bmatrix} \Delta\lambda_1 \\ \Delta\lambda_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}. \tag{13}$$

Eliminating $\Delta\lambda_1$ from the first group of equations in (13), we get

$$(E - C^\top B^{-1} C)\Delta\lambda_2 = (g_2 - C^\top B^{-1} g_1) \tag{14}$$

$$B\Delta\lambda_1 = (g_1 - C\Delta\lambda_2). \tag{15}$$

Following Castro (2000, 2007); Castro & Cuesta (2011), system (15) will be solved by performing one Cholesky factorization for each diagonal block $N_i\Theta^i N_i^\top, i = 1, \dots, k$ of matrix B. Computing the matrix of system (14) can be very expensive, because it involves the inverse of B. Even if computed, it might result in a dense and large matrix, whose factorization would be prohibitive. Therefore, system (14) will be solved by an iterative method, namely the preconditioned conjugate gradient (PCG). The dimension of this system is l (the number of linking constraints, as previously stated), which can be very large in practice. Therefore a good preconditioner is crucial for a fast solution of (14).

The preconditioner initially developed in Castro (2000) for multicommodity flows can be used for any primal block-angular problem (Castro, 2007). It is based on the following Neumann series of the inverse of the matrix of system (14):

$$(E - C^\top B^{-1} C)^{-1} = \left(\sum_{i=1}^{\infty} (E^{-1}(C^\top B^{-1} C))^{i-1} \right) E^{-1}. \tag{16}$$

It was proven in Castro (2000) that the eigenvalues of $E^{-1}(C^\top B^{-1} C)$ are in ([0,1), and that the infinite sum in (16) converges. The preconditioner is obtained by considering a number of terms (say, ϕ) of the infinite sum. Its efficiency depends on the following two factors:

- The spectral radius ρ (i.e., the largest eigenvalue) of the matrix $(E^{-1}(C^\top B^{-1} C))$. When ρ is not excessively close to 1, the contribution of higher order terms in the series decreases quickly, and then a small ϕ is enough for a good approximation of the inverse of $E - C^\top B^{-1} C$. Unfortunately, the value of the spectral radius ρ is problem dependent and cannot be controlled or determined a priori. We have just a few results stating that ρ is smaller for quadratic problems than for linear ones (see Castro & Cuesta, 2011, Theorem 1, Proposition 2). Although computing ρ is not practical, it can be approximated using the Ritz values of the PCG, as described in Bocanegra et al. (2013). This allows

monitoring their values along the IPM iterations. Although, in theory, the greater the ϕ, the better the preconditioner, increasing ϕ by one means solving an additional system with matrices E and B at each PCG iteration. In practice, it has been observed (Castro, 2000; 2007; 2016; Castro & de la Lama-Zubirán, 2020) that the best results are obtained for $\phi = 1$ (that is, the preconditioner is E^{-1}) or $\phi = 2$ (in this case the preconditioner is $(E^{-1}(C^\top B^{-1}C))E^{-1}$). In general, for very large problems, $\phi = 2$ can be very time consuming due to the extra computations needed (Castro, 2016). For this reason, this work obtains all the computational results with $\phi = 1$.

- The efficient solution of systems with matrix E is instrumental for the performance of the method. Unlike the spectral radius ρ, which is not possible to determine a priori whether or not it will be far from 1, the particular structure of E for any multistage stochastic instance can be analyzed before starting the IPM iterations. This is done in the next subsection, which shows that systems with E are easy to solve.

By abuse of notation, matrix E is denoted as the preconditioner throughout the rest of the work.

4.1. The structure of preconditioner E

According to (12), preconditioner E is defined as

$$\mathbb{R}^{l\times l} \ni E = \Theta_0 + \sum_{i=1}^{k} R_i \Theta_i R_i^\top. \tag{17}$$

Let $\bar{k}$ denote the number of (nested) *computational* strategic-operational two-stage trees in the multistage scenario tree, where $\bar{k} = |\mathcal{N}|$ for $|\Pi_{|\mathcal{T}|}| > 0$ and otherwise, $\bar{k} = |\mathcal{N} \setminus \mathcal{N}_{|\mathcal{T}|}|$. The first node in any of those trees is a strategic node n and the node set in the second stage is

$$\mathcal{S}_1^n \cup \Pi_{t^n} \text{ if } t^n < |\mathcal{T}|, \quad \text{or} \quad \Pi_{t^n} \text{ if } t^n = |\mathcal{T}|. \tag{18}$$

For problems where the number of strategic and operational nodes is constant for each node (that is, $\ell = |\mathcal{S}_1^n|$ for all $n \in \mathcal{N} : t^n < |\mathcal{T}|$, and $|\Pi| = |\Pi_t|$ for all $t \in \mathcal{T}$), the number of nested two-stage trees is

$$\bar{k} = \begin{cases} \dfrac{\ell^{|\mathcal{T}|} - 1}{\ell - 1} & \text{if } |\Pi| > 0 \\ \dfrac{\ell^{|\mathcal{T}|-1} - 1}{\ell - 1} & \text{if } |\Pi| = 0\,. \end{cases} \tag{19}$$

As discussed in the previous section, the linking constraints can also be partitioned by rows in groups of constraints for each nested two-stage tree (e.g., see again Fig. 4 which is associated to the scenario tree depicted in Fig. 2). Then, the linking constraint matrix can be decomposed as

$$\begin{bmatrix} R_1 & R_2 & \dots & R_k \end{bmatrix} = \begin{bmatrix} R_{11} & R_{21} & \dots & R_{k1} \\ R_{12} & R_{22} & \dots & R_{k2} \\ \vdots & & & \\ R_{1\bar{k}} & R_{2\bar{k}} & \dots & R_{k\bar{k}}. \end{bmatrix}, \tag{20}$$

Each submatrix R_{ij} of (20), $i = 1, \dots, k$, for the two-stage tree j, $j = 1, \dots, \bar{k}$, is any of the following matrices (see Fig. 4 for an example):

- A zero matrix, if the variables of block i (which is associated with some node in $\mathcal{N}$ of the scenario tree) do not intervene in the two-stage tree j.
- A matrix containing an identity submatrix I, and zeros elsewhere, if block i corresponds to the root node $n \in \mathcal{N}$ of the strategic two-stage tree j. This identity starts the splitting of variables x^n, and its dimension is the number of components of x^n.
- A matrix containing a submatrix of the form $\begin{bmatrix} -I \\ I \end{bmatrix}$ associated with a node in set $\mathcal{S}_1^n \cup \Pi_{t^n}$, other than the last node, where n is the root node of the two-stage tree. The dimension of I and $-I$ is the number of components of x^n. This matrix continues the splitting of x^n between nodes in $\mathcal{S}_1^n \cup \Pi_{t^n}$.
- A matrix containing a submatrix $-I$, associated with the last node in set $\mathcal{S}_1^n \cup \Pi_{t^n}$, where n is the root node of the two-stage tree. This matrix ends the splitting of x^n, and its dimension is the number of components of x^n.

Then, from (17) and (20), preconditioner E can be rewritten as

$$\begin{aligned} E &= \Theta_0 + \begin{bmatrix} \sum_{i=1}^{k} R_{i1}\Theta_i R_{i1}^\top & \dots & \sum_{i=1}^{k} R_{i1}\Theta_i R_{i\bar{k}}^\top \\ \vdots & & \vdots \\ \sum_{i=1}^{k} R_{i\bar{k}}\Theta_i R_{i1}^\top & \dots & \sum_{i=1}^{k} R_{i\bar{k}}\Theta_i R_{i\bar{k}}^\top \end{bmatrix} \\ &= \Theta_0 + \begin{bmatrix} \sum_{i=1}^{k} R_{i1}\Theta_i R_{i1}^\top & & \\ & \ddots & \\ & & \sum_{i=1}^{k} R_{i\bar{k}}\Theta_i R_{i\bar{k}}^\top \end{bmatrix}, \end{aligned} \tag{21}$$

where the last equality comes from the fact that each two-stage tree has its own split-variables, in other words, the matrices I, $\begin{bmatrix} -I \\ I \end{bmatrix}$, and $-I$ of R_{ij} and $R_{ij'}$, $j \neq j'$, are located in different columns, and, thus, $R_{ij}\Theta_i R_{ij'}^\top = 0$. Therefore, considering an appropriate partition Θ_{0j}, $j = 1, \dots, \bar{k}$, of the diagonal matrix Θ_0, it follows that E is a block diagonal matrix with $\bar{k}$ block submatrices $\Theta_{0j} + \sum_{i=1}^{k} R_{ij}\Theta_i R_{ij}^\top$, $j = 1, \dots, \bar{k}$, each of them associated with a (nested) two-stage tree of the multistage scenario tree. For any two-stage tree $j \in \{1, \dots, \bar{k}\}$ that is associated with some node $n \in \mathcal{N}$, the structure of $\begin{bmatrix} R_{1j} & \dots & R_{kj} \end{bmatrix}$ is as follows

$$\begin{bmatrix} R_{1j} & \dots & R_{kj} \end{bmatrix} = \begin{array}{c} \begin{matrix} x^n & & x_n^1 & x_n^2 & \dots & x_n^{\xi-1} & x_n^{\xi} \end{matrix} \\ \begin{bmatrix} \dots & I & \dots & -I & & & \\ & & & I & -I & & \\ & & & & \ddots & & \\ & & & & & I & -I \end{bmatrix} \end{array}, \tag{22}$$

where, by abuse of notation, x_n^i, $i = 1, \dots, \xi$, represents either the copy of x^n in the ith strategic immediate successor node of root node n, $i \in \mathcal{S}_1^n$, or it is the copy of x^n in the πth operational node, $\pi \in \Pi_{t^n}$. Note that $\xi = |\mathcal{S}_1^n| + |\Pi_{t^n}|$.

From (22), and by block multiplication, we get

$$\sum_{i=1}^{k} R_{ij}\Theta_i R_{ij}^\top = \begin{bmatrix} \Theta^{x^n} + \Theta^{x_n^1} & -\Theta^{x_n^1} & & & \\ -\Theta^{x_n^1} & \Theta^{x_n^1} + \Theta^{x_n^2} & -\Theta^{x_n^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\Theta^{x_n^{\xi-2}} & \Theta^{x_n^{\xi-2}} + \Theta^{x_n^{\xi-1}} & -\Theta^{x_n^{\xi-1}} \\ & & & -\Theta^{x_n^{\xi-1}} & \Theta^{x_n^{\xi-1}} + \Theta^{x_n^{\xi}} \end{bmatrix}, \tag{23}$$

that is, $\sum_{i=1}^{k} R_{ij}\Theta_i R_{ij}^\top$ is a *v-shifted tridiagonal matrix*, where v is the number of components of x^n, which is also the dimension of

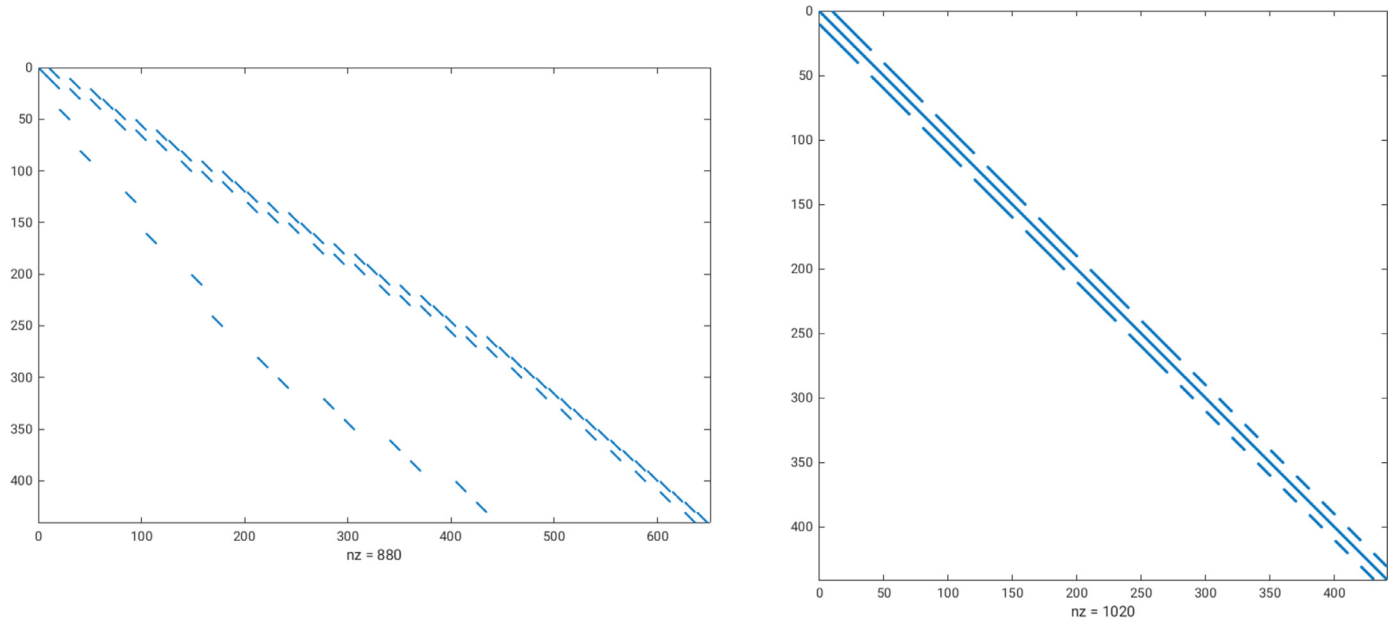

Fig. 5. Structure of (a) linking constraints and (b) preconditioner E, for a problem with $|\mathcal{T}| = 4$ stages, $|\mathcal{S}_1^n| = 2$ for $n \in \mathcal{N} : t^n < |\mathcal{T}|$ and $|\Pi_{t^n}| = 2$, $n \in \mathcal{N}$.

the Θ-matrices. This type of matrix is a generalization of a tridiagonal one where the superdiagonal (nonzero diagonal above the main diagonal) and subdiagonal (nonzero diagonal below the main diagonal) are shifted ν positions from the main diagonal. In other words, elements (i, j) are non-zero only if $|i - j|$ is either 0 or ν. The matrices with such a structure can be efficiently factorized with zero fill-in by extending a standard factorization for tridiagonal ones. Therefore, systems with the preconditioner E are reduced to the solution of $\bar{k}$ independent smaller systems, each one involving the (fast) factorization of a ν-shifted tridiagonal matrix. In addition, note that systems with E can be easily parallelized for the $\bar{k}$ smaller systems.

The description above proves the following result:

Proposition 2. *For any multistage stochastic optimization problem, with both strategic and operational decisions, based on the split-variable formulation (2), matrix E, as defined in (17), is block diagonal with $\bar{k}$ blocks; and each block j is a ν^j-shifted tridiagonal matrix, where $\bar{k}$ is the number of nested two-stage trees in the multistage scenario tree, and ν^j is the number of variables replicated in the two-stage tree $j \in \{1, \ldots, \bar{k}\}$.*

Proof. From the definitions of E and $\begin{bmatrix} R_1 & \ldots & R_k \end{bmatrix}$ in, respectively, (17) and (20), the structure of preconditioner E is given by (21), where the last equality of (21) comes from the fact that $R_{ij}\Theta_i R_{ij'}^\top = 0, j \neq j'$. For any two-stage tree $j \in \{1, \ldots, \bar{k}\}$ that is associated with a node $n \in \mathcal{N}$, the structure of $\begin{bmatrix} R_{1j} & \ldots & R_{kj} \end{bmatrix}$ is given by (22). From (22), and by block multiplication, we get that any diagonal block of (21) has the form of (23), proving that preconditioner E is a block diagonal matrix with $\bar{k}$ blocks, each block being a ν^j-shifted tridiagonal matrix. □

Figure 5 shows the structure of $\begin{bmatrix} R_1 & \ldots & R_k \end{bmatrix}$ and preconditioner E for a problem with $|\mathcal{T}| = 4$ stages, $|\mathcal{S}_1^n| = 2$ for $n \in \mathcal{N} : t^n < |\mathcal{T}|$ and $|\Pi| = |\Pi_{t^n}| = 2$ for every node $n \in \mathcal{N}$ (then, $|\mathcal{N}| = 15$), where the number of split-variables at every two-stage tree is $\nu = 10$. Note that, according to (19), since $|\Pi| > 0$ the number of two-stage trees (i.e., tridiagonal blocks of E) in Fig. 5 is $\bar{k} = \frac{2^4-1}{2-1} = 15$, and each of those trees has four and two two-stage strategic-operational nodes for $t^n < 4$ and $t^n = 4$, respectively, see (18).

It is worth pointing out that the approach in Castro & de la Lama-Zubirán (2020) for the split-variable formulation of two-stage stochastic problems is a particular case (when $\bar{k} = 1$) of the more general preconditioner E presented in this work for the split-variable formulation (2). One of the most significant features of the proposal is that the complexity of the preconditioner for a multistage stochastic problem (where E has $\bar{k}$ ν-shifted tridiagonal matrices) does not grow in comparison with that of a large two-stage problem (where E would be a unique large ν-shifted tridiagonal matrix). In other approaches, like a nested Benders decomposition (see Birge, 1985), solving a multistage case is generally more complex than solving a two-stage problem.

5. Computational results

This section presents the computational validation of the new approach based on two pilot applications. The first one is *supply network design* presented in Appendix A, where the multistage stochastic scenario tree has strategic and operational nodes. The second application is *revenue management*, presented in Appendix B, where the multistage stochastic scenario tree has only strategic nodes. For each of the instances in the testbed of each pilot application, the new approach MSSO-BlockIP is compared with the state-of-the-art solver CPLEX v20.1 using both splitting and non-splitting modeling schemes.

5.1. Implementation details and computational environment

The specialized IPM described in Section 4 was implemented in C++ giving rise to the BlockIP package (Castro, 2007; 2016). In this work, BlockIP has been extended to deal with the preconditioner E described in the previous section. The new code, named MSSO-BlockIP, runs on top of BlockIP, and it is appropriate for very large multistage stochastic optimization problems with both strategic and operational uncertainties. MSSO-BlockIP

and its user's guide can be retrieved from http://www-eio.upc.edu/~jcastro/MSSO-BlockIP.html. Given the sizes of the instances to be tested (up to hundreds of millions of variables and tens of millions of constraints), we considered an optimality tolerance of 10^{-2} in MSSO-BlockIP; that is, we require a primal and dual solution with a relative duality gap (i.e., difference between the objective functions in (4) and (5)) of less than 10^{-2}. For such huge problems, the only reliable and non-heuristic code (thus, able to compute an optimal solution) that can be used for comparison with MSSO-BlockIP is some state-of-the-art implementation of an IPM. In this work, we used the standard primal-dual implementation of the barrier algorithm in CPLEX v20.1; for huge problems this variant is expected to outperform the homogeneous self-dual IPM in CPLEX v20.1, which is selected by default in some cases. Default values were used for all the CPLEX parameters except for the crossover postprocess, which was deactivated (i.e., the solver provides an interior-point solution instead of a basic one). Another modified CPLEX parameter was the optimality tolerance, which was also set to 10^{-2} in order to make a fair comparison with MSSO-BlockIP. This is an additional argument for using the standard primal-dual barrier instead of the homogeneous self-dual algorithm, since the latter would not benefit from reducing the optimality tolerance; on the other hand, the standard primal-dual code can trigger an early stop with feasible primal and dual solutions, as well as with a desired duality gap.

One of the most influential parameters in MSSO-BlockIP for the efficient solution of (14) is the tolerance required by the PCG. This tolerance is dynamically updated at each interior-point iteration i as $\epsilon_i = \max\{\beta\epsilon_{i-1}, \min_\epsilon\}$, where ϵ_0 is the initial tolerance, $\min_\epsilon$ is the minimum allowed tolerance, and β is a tolerance reduction factor at each interior-point iteration. For the very large MSSO problems solved in this work we used by default the conservative values: $\epsilon_0 = \min_\epsilon = 10^{-2}$, $\beta = 1$ for the supply network design instances in Section 5.2; and $\epsilon_0 = 10^{-2}$, $\beta = 0.98$ and $\min_\epsilon = 10^{-3}$ for the revenue management problems in Section 5.3. Although tighter values may significantly increase the number of PCG iterations, it was necessary in some cases to reduce β and $\min_\epsilon$ in order to obtain either a solution or a faster solution (e.g., we used $\beta = 0.98$, and $\min_\epsilon = 10^{-3}$ for a few supply network design instances; and $\beta = 0.95$, and $\min_\epsilon \in \{10^{-4}, 10^{-5}\}$ for a few revenue management problems).

The values of the objective function for the solutions will be omitted in the tables of results of next sections, in order to save space; and both MSSO-BlockIP and CPLEX v20.1 reached similar solutions, with relative differences of around 10^{-3} in the optimal values.

It is worth noting that BlockIP performs matrix factorizations of linear equation systems using the academic Ng–Peyton block sparse Cholesky package (Ng & Peyton, 1993), which uses an approximate minimum degree algorithm for reordering constraints and variables. On the other hand, modern state-of-the-art IPM commercial solvers (such as CPLEX v20.1 and others) implement highly efficient numerical linear algebra routines that exploit hardware capabilities (Mészáros, 2016). Therefore, any computational advantage of MSSO-BlockIP over CPLEX is not due to implementation details, but to the specialized algorithm introduced in this work.

The computational experiments in this work were carried out on a Fujitsu Primergy RX2530 M4 server with two 2.3 GHz Intel Xeon Gold 6140 CPUs (72 cores) and 503 gigabytes of RAM, running on a GNU/Linux operating system (openSuse 15.0). However, because of memory limitations, the two largest instances of below Section 5.2 (whose dimensions and results are reported in last two rows of Tables 1–3, namely, instances SC-T5-L4-PI-10-P40-S2000-C4000-L and SC-T5-L4-PI-10-P40-S2000-C4000-Q) had to be run on a different and larger server: a DELL PowerEdge R7525 with two 2.4 GHz AMD EPYC 7532 CPUs (128 total cores) and 768 Gigabytes of RAM, running on a GNU/Linux operating system (openSuse 15.3). Unless otherwise stated, all the runs were sequential, i.e., without exploitation of multithreading capabilities.

5.2. Supply network design planning

A multistage extension of the stochastic two-stage problem in Castro & de la Lama-Zubirán (2020) is presented in Appendix A. In this problem the raw material supply and end product production and distribution network $\mathcal{G} = (\mathcal{S} \cup \mathcal{P} \cup \mathcal{C}, A)$ is considered, where $\mathcal{S} \cup \mathcal{P} \cup \mathcal{C}$ is the set of nodes, A is the set of arcs, $\mathcal{S}$ is the set of raw materials to be supplied, $\mathcal{P}$ is the set of potential plants where the raw materials are processed, and $\mathcal{C}$ is the set of customer centers where the end product is distributed to satisfy the demand.

The aim of the problem, as presented in Appendix A, consists of deciding on the strategic manufacturing plant locations in the network, such that the total expected cost is minimized along the time horizon in both the strategic and operational scenarios. The uncertain strategic parameters are the stagewise-dependent initial capacity of the plant and its expansion unit costs, and the plant capacity's residual unit value. The uncertain operational parameters are the stage-dependent unit cost of the raw material being supplied and transported to the manufacturing plants, the manufacturing plant's required capacity for processing a unit of raw material, and the end product demand from customer centers, among others. According to the metamodels (1) and (2), two application models can be derived, depending on whether or not copies of variables are considered. These are named the split-variable formulation (24) and the compact model (25).

We generated a set of eight very large instances (four linear and four quadratic), which the authors will provide upon request. These instances contain both strategic and operational uncertainties, and their sizes are reported in Table 1. Columns headed with $|\mathcal{T}|$ and ℓ give, respectively, the number of stages and immediate successors of each node in the scenario tree, where $\ell = |\mathcal{S}_1^n| \, \forall n \in \mathcal{N} : t^n < |T|$. Column $|\Pi|$ shows the number of operational scenarios in each strategic node. Columns k, "#var. L.", "#var. Q.", "#cons.", and l show, respectively, the numbers of blocks (i.e., the number of nodes in the multistage scenario tree), linear variables, quadratic variables, block constraints $\sum_{i=1}^{k} \overline{m}_i$, and linking constraints in the problem. Instances' names are denoted as SC-Tx-Ly-PI-z-Pu-Sv-Cw-L/Q, where "x" $= |\mathcal{T}|$, "y" $= \ell$, "z"$= |\Pi|$, "u" is the number of potential plants, "v" is the number of raw materials, "w" is the number of customer centers, and L/Q denotes whether the problem is linear or quadratic. It can be noted from Table 1 that the number of linking constraints equating the split-variables is not very large. In other words, despite they are instances with huge numbers of variables, the numbers of split-variables for them are moderately low.

Table 2 shows the results for the supply network design instances that were obtained using MSSO-BlockIP and CPLEX v20.1. MSSO-BlockIP solved the split-variable formulation (2) as a primal block-angular problem. Three different runs were performed with CPLEX v20.1 in order to make a fair comparison with MSSO-BlockIP, and they were marked as variants $^{(1)}$, $^{(2)}$ and $^{(3)}$. CPLEX variant $^{(1)}$ solved the split-variable formulation (2) using the default CPLEX aggregator option, which may remove many of the splitting constraints (2d) and (2e) and, thus, reduce the size of the problem; however, it can eventually degrade the sparsity of the constraint matrix and thereby increase the fill-in of IPM factorizations. To avoid this fill-in issue, CPLEX variant $^{(2)}$ also solved the split-variable formulation (2), but it deactivated the default CPLEX aggregator option (that is, CPLEX v20.1 solves the same model as MSSO-BlockIP). Finally, CPLEX variant $^{(3)}$ solved the compact formulation (1) (without the default CPLEX aggregator, which turned

Table 1
Sizes of supply network design instances.

Instance	$\|\mathcal{T}\|$	ℓ	$\|\Pi\|$	k	# var. L.	# var. Q.	# cons.	l
SC-T4-L4-PI-10-P40-S2000-C4000-L	4	4	10	935	209,219,005	–	5,212,245	37,360
SC-T4-L4-PI-10-P40-S2000-C4000-Q	4	4	10	935	205,819,005	3,400,000	5,212,245	37,360
SC-T4-L4-PI-20-P40-S2000-C4000-L	4	4	20	1785	418,421,005	–	10,414,245	71,360
SC-T4-L4-PI-20-P40-S2000-C4000-Q	4	4	20	1785	411,621,005	6,800,000	10,414,245	71,360
SC-T3-L15-PI-15-P40-S4000-C2000-L	3	15	15	3856	889,772,161	–	22,152,921	154,200
SC-T3-L15-PI-15-P40-S4000-C2000-Q	3	15	15	3856	882,542,161	7,230,000	22,152,921	154,200
SC-T5-L4-PI-10-P40-S2000-C4000-L	5	4	10	3751	839,187,661	–	20,760,421	150,000
SC-T5-L4-PI-10-P40-S2000-C4000-Q	5	4	10	3751	825,547,661	13,640,00	20,760,421	150,000

Table 2
Results for supply network design instances. CPU times are in seconds unless otherwise stated. Fastest execution in boldface.

Instance	MSSO-BlockIP			CPLEX(1)		CPLEX(2)		CPLEX(3)	
	it.	PCG	CPU	it.	CPU	it.	CPU	it.	CPU
SC-T4-L4-PI-10-P40-S2000-C4000-L	179	9071	24,296	28	50,285	24	42,906	13	**5997**
SC-T4-L4-PI-10-P40-S2000-C4000-Q	40	6043	14,770	47	84,252	45	78,564	23	**9528**
SC-T4-L4-PI-20-P40-S2000-C4000-L	135	13,904	67,371	32	441,330	27	392,539	14	**33,004**
SC-T4-L4-PI-20-P40-S2000-C4000-Q	122	8934	**46,572**	62	842,572	48	641,410	46	108,112
SC-T3-L15-PI-15-P40-S4000-C2000-L	440	27,836	**323,956**	◇	◇	◇	◇	18	389,321
SC-T3-L15-PI-15-P40-S4000-C2000-Q	507	31,694	**371,227**	◇	◇	◇	◇	35	746,641
SC-T5-L4-PI-10-P40-S2000-C4000-L	370	34,067	**201,205**△,§	◇	◇	◇	◇	24	2,024,861△,*
SC-T5-L4-PI-10-P40-S2000-C4000-Q	217	12,655	**99,147**△,‡	◇	◇	◇	◇	57	4,632,673△,†

(1) CPLEX v20.1 solved the split-variable formulation (2), with default aggregator.
(2) CPLEX v20.1 solved the split-variable formulation (2), deactivating default aggregator.
(3) CPLEX v20.1 solved the compact formulation (1), deactivating default aggregator.
* 2,024,861 seconds = 23 days: 10 hours: 27 minutes: 41 seconds.
† 4,632,673 seconds = 53 days: 14 hours: 51 minutes: 21 seconds.
◇ Out of memory.
△ Execution performed on a different server of 755gigabytes of RAM.
§ 334,740 seconds if executed in usual server.
‡ 144,238 seconds if executed in usual server.

Table 3
Memory requirements (in gigabytes of RAM) for supply network design instances.

Instance	MSSO-BlockIP	CPLEX(1)	CPLEX(2)	CPLEX(3)
SC-T4-L4-PI-10-P40-S2000-C4000-L	42	126	126	117
SC-T4-L4-PI-10-P40-S2000-C4000-Q	43	128	128	119
SC-T4-L4-PI-20-P40-S2000-C4000-L	83	285	285	246
SC-T4-L4-PI-20-P40-S2000-C4000-Q	86	289	289	249
SC-T3-L15-PI-15-P40-S4000-C2000-L	164	> 503	> 503	480
SC-T3-L15-PI-15-P40-S4000-C2000-Q	184	> 503	> 503	482
SC-T5-L4-PI-10-P40-S2000-C4000-L	167	> 503	> 503	526
SC-T5-L4-PI-10-P40-S2000-C4000-Q	174	> 503	> 503	531

(1) CPLEX v20.1 solved the split-variable formulation (2), with default aggregator.
(2) CPLEX v20.1 solved the split-variable formulation (2), deactivating default aggregator.
(3) CPLEX v20.1 solved the compact formulation (1), with default aggregator.

out to be the fastest option for the compact model of the supply network design instances). Columns "it." and "CPU" provide the number of IPM iterations and CPU time in seconds for each run (unless otherwise stated). The overall number of PCG iterations is also given for MSSO-BlockIP. The CPU of the fastest execution is marked in boldface.

Looking at Table 2 it can be concluded that, for supply network design instances, both CPLEX variants (1) and (2) based on the split-variable formulation were never competitive with the compact variant (3). Indeed, variants (1) and (2) exhausted the 503 gigabytes of the server for the four largest instances. It is also observed that MSSO-BlockIP was significantly faster than any CPLEX variant for the largest instances, especially for the last two. These last two instances were executed on a different server (with 755 gigabytes of RAM) because even CPLEX compact variant (3) required more than 503 gigabytes. For the largest linear case, MSSO-BlockIP found a solution in 2.3 days, while CPLEX needed more than 23 days. The difference is even more dramatic for the quadratic problem: MSSO-BlockIP and CPLEX required, respectively, 1.1 and 53 days. Table 2 also shows that MSSO-BlockIP is generally more efficient for quadratic than for linear problems, which is consistent with the theoretical results found in Castro & Cuesta (2011).

MSSO-BlockIP may also be much more efficient than general IPM solvers in terms of memory requirements. This can be seen in Table 3, which reports the gigabytes of RAM required by each method for all the supply network design instances. MSSO-BlockIP clearly requires a fraction of the memory used by CPLEX in those instances. Therefore MSSO-BlockIP could be successfully run on much smaller hardware, thereby significantly saving energy and contributing to a reduction in CO_2 emissions.

Finally, since one of the strengths of state-of-the-art implementations of IPMs (such as CPLEX) is its ability to efficiently scale to multiple threads (i.e., the Newton direction is solved by highly efficient parallel sparse Cholesky solvers), we ran the six largest instances of Tables 1–3 with the CPLEX v20.1 parallel barrier. We only considered the compact model (1) with CPLEX v20.1, because, as shown in Table 2, it was (by a large margin) the most efficient formulation with the sequential barrier. We also ran these in-

Table 4
Results of parallel runs for supply network design instances. Times are wall-clock seconds. Fastest execution in boldface.

Instance	threads	MSSO-BlockIP				CPLEX(3)		
		it.	PCG	Time	Speedup	it.	Time	Speedup
SC-T4-L4-PI-20-P40-S2000-C4000-L	50	135	13,904	**8859**	7.6	20	9118	3.6
SC-T4-L4-PI-20-P40-S2000-C4000-Q	50	122	8934	**6979**	6.7	58	26,875	4.0
SC-T3-L15-PI-15-P40-S4000-C2000-L	50	440	27,836	53,652	6.0	17	**42,686**	9.1
SC-T3-L15-PI-15-P40-S4000-C2000-Q	50	507	31,694	**65,516**	5.7	48	106,405	7.0
SC-T5-L4-PI-10-P40-S2000-C4000-L	100	370	34,067	**33,470**△	6.0	27	54,136△	37.4
SC-T5-L4-PI-10-P40-s2000-C4000-Q	100	217	12,655	**15,072**△	6.6	65	128,192△	36.1

(3) CPLEX v20.1 solved the compact formulation (1).
△ Execution performed on a different server of 755gigabytes of RAM.

stances with a preliminary parallel version of MSSO-BlockIP based on openMP. It is worth noting that this preliminary parallel MSSO-BlockIP only focuses on the solution of the k systems with matrix B of (12) in parallel. However, each of those k systems is solved with the standard sequential sparse Cholesky solver of Ng & Peyton (1993), instead of with a highly efficient parallel one. The parallel results are reported in Table 4. We considered 50 threads for the first four instances, and 100 for the two largest ones (which were run in the larger server), as shown in column "threads" of Table 4. Columns "Times" give the wall-clock time (instead of CPU time), in seconds. Columns "Speedup" give the ratio t_1/t_κ, where t_1 and t_κ are, respectively, the execution times with one thread (sequential time) and κ threads (parallel time). Then, the greater the speedup, the faster the parallel execution. The sequential times are those in columns "CPU" (for MSSO-BlockIP and CPLEX(3)) in Table 2. We want to stress that the sequence of points (and number of iterations) generated by the CPLEX v20.1 parallel barrier differs from that of the sequential barrier, even imposing the option "deterministic parallel mode". On the other hand, MSSO-BlockIP always generates the same sequence of points, both in parallel and sequential runs. We see that, even with a preliminary parallel implementation, MSSO-BlockIP produced the fastest runs (which are marked in boldface in Table 4) for all but one instance. Comparing the speedups of MSSO-BlockIP and CPLEX, they were similar for the first four instances, but for the two largest problems CPLEX obtained speedups six times greater than MSSO-BlockIP (approx., 36 vs 6). A possible explanation is that, in huge problems, the solution of the linear systems of equations for the Newton direction dominates the rest of steps of the algorithm, and then a highly efficient parallel sparse Cholesky solvers is more efficient. Indeed, according to *Amdahl's law* (Bertsekas & Tsitsiklis, 1995), the theoretical speedup $\bar{s}$ (which is an upper bound for the observed speedup) is given by the expression

$$\bar{s} = \frac{1}{f/\kappa + (1-f)},$$

where κ is the number of threads, and f is the fraction of the algorithm which is run in parallel (thus $1-f$ is the fraction of the algorithm which is run sequentially, and it represents a bottleneck for the parallel execution). From Amdahl's law we have that

$$f = \frac{\kappa(\bar{s}-1)}{\bar{s}(\kappa-1)}.$$

If, from the last two rows of Table 2, we consider that $\bar{s}=6$ for MSSO-BlockIP and $\bar{s}=36$ for CPLEX, and using $\kappa=100$, the corresponding values of f would be $f = \frac{100(6-1)}{6(100-1)} \approx 0.84$ for MSSO-BlockIP, and $f = \frac{100(36-1)}{36(100-1)} \approx 0.98$. That is, the parallel fraction of the algorithm is 98% for CPLEX, whereas it is just 84% for MSSO-BlockIP. This gap could be reduced by attempting to parallelize other sections of BlockIP, such as the PCG solver. This is part of future research on BlockIP.

5.3. Revenue management (RM)

As pointed out in Talluri & van Ryzin (2004), "revenue management aims to maximize the revenue of selling limited quantities of a set of resources by means of demand management decisions. A resource in RM is usually a perishable product/service, such as seats on a single flight leg or hotel rooms for a given date. It is common in RM that multiple resources are sold in bundles".

The aim of the problem, as presented in Appendix B, consists of deciding on the number of accepted bookings for bundle-class in any stage, as well as in the stages previous to the service that will be provided, such that the expected income is maximized along the scenarios time horizon. The uncertain (strategic) parameter is the stagewise-dependent bundle-class demand, which means that we are dealing with a multistage strategic tree. As for the supply network design instances, we also consider the compact model (26) and its split-variable formulation (27) by following (1) and (2), respectively.

We generated a set of 14 large (linear) instances following Escudero et al. (2013), and the objective function was transformed to a minimization formulation. These instances are available from the authors by request. Their sizes are reported in Table 5. Column headed with "#var." shows the number of (linear) variables $\sum_{i=1}^{k} \overline{n}_i$ of the problem. The rest of columns have the same meaning as in Table 1. Instances' names are denoted as RM[-c]-Tx-Ly, where "x" $= |\mathcal{T}|$ and "y" $= \ell$. The difference between RMc-Tx-Ly and RM-Tx-Ly instances is that the former includes an additional set of constraints and variables (see the note about the tightening formulation of model (26) considered in Section B.2 of Appendix B). Table 5 shows that, unlike in the supply network design instances, the number of linking constraints is very large. This is due to RM problems having large numbers of split-variables.

Table 6 shows the results obtained for the RM instances using MSSO-BlockIP and CPLEX v20.1. The columns headings are the same as those for the supply network design instances in Table 2, with the only exception that CPLEX variant (3) was ran with the default aggregator, since it was the most efficient option for the compact RM model. The CPLEX variants (1) and (2) are based on the split-variable formulation. For RM problems, the latter outperformed the former in many instances. We note that this means the split-variable formulation (2) has collateral benefits not only for our specialized method but also for general state-of-the-art solvers. The best CPLEX variants for RM problems were generally (2) and (3).

It can be observed in Table 6 that MSSO-BlockIP required many more IPM iterations than CPLEX. This is due to the inexact Newton directions provided by the PCG, whereas CPLEX instead uses more accurate directions computed by Cholesky factorizations. It is known, however, that inexact directions can be used in IPMs without significantly affecting their convergence properties (Gondzio, 2013). The CPU time of the fastest execution is marked in boldface, and it is clearly observed that MSSO-BlockIP outperformed

Table 5
Sizes of revenue management instances.

Instance	$\|\mathcal{T}\|$	ℓ	k	# var.	# cons.	l
RMc-T5-L5	5	5	781	1,951,700	702,900	624,000
RM-T5-L5	5	5	781	1,936,100	687,300	624,000
RMc-T8-L3	8	3	3280	8,199,200	2,952,000	2,623,200
RM-T8-L3	8	3	3280	8,089,900	2,842,700	2,623,200
RMc-T3-L100	3	100	10,101	25,251,700	9,090,900	8,080,000
RM-T3-L100	3	100	10,101	25,241,600	9,080,800	8,080,000
RMc-T4-L25	4	25	16,276	40,689,200	14,648,400	13,020,000
RM-T4-L25	4	25	16,276	40,624,100	14,583,300	13,020,000
RMc-T10-L3	10	3	29,524	73,809,200	26,571,600	23,618,400
RM-T10-L3	10	3	29,524	72,825,100	25,587,500	23,618,400
RMc-T8-L5	8	5	97,656	244,139,200	87,890,400	78,124,000
RM-T8-L5	8	5	97,656	242,186,100	85,937,300	78,124,000
RMc-T6-L10	6	10	111,111	277,776,700	99,999,900	88,888,000
RM-T6-L10	6	10	111,111	276,665,600	98,888,800	88,888,000

Table 6
Results for revenue management instances. CPU times are in seconds unless otherwise stated. Fastest execution in boldface.

Instance	MSSO-BlockIP			CPLEX(1)		CPLEX(2)		CPLEX(3)	
	it.	PCG	CPU	it.	CPU	it.	CPU	it.	CPU
RMc-T5-L5	87	2136	**83**	19	268	35	177	14	135
RM-T5-L5	183	15,182	485	16	303	27	137	18	**104**
RMc-T8-L3	143	5706	**884**	24	7447	35	1985	15	2825
RM-T8-L3	239	6278	**1040**	22	2465	41	**1040**	28	12,521
RMc-T3-L100	210	9866	**4968**	44	6971	56	7845	22	23,073
RM-T3-L100	249	8726	**4558**	44	5049	40	7719	19	21,234
RMc-T4-L25	265	11,695	10,049	39	23,904	44	30,590	25	**3823**
RM-T4-L25	733	7244	9479	39	10,165	46	14,928	23	**3584**
RMc-T10-L3	471	16,874	**24,952**	38	101,444	48	91,650	–	>14d†
RM-T10-L3	355	16,661	**22,923**	33	82,746	45	60,121	–	>33d†
RMc-T8-L5	907	27,515	**151,979**	55	1,710,241*	–	>6d†	–	>14d†
RM-T8-L5	1360	15,837	**112,013**	–	>31d†	–	>10d†	–	>144d†
RMc-T6-L10	673	34,891	**197,701**	–	>17d†	–	>3.3d†	38	235,954
RM-T6-L10	1165	17,730	**129,291**	–	>22d†	–	>3.3d†	33	207,714

(1) CPLEX v20.1 solved the split-variable formulation (2), with default aggregator.
(2) CPLEX v20.1 solved the split-variable formulation (2), deactivating default aggregator.
(3) CPLEX v20.1 solved the compact formulation (1), with default aggregator.
* 1,710,241 seconds = 19 days: 19 hours: 33 minutes: 36 seconds.
– Execution was stopped early by excessive expected CPU time.
† Estimated number of days of CPU from the total number of arithmetic operations for factorizations reported in the CPLEX log file.

CPLEX in all but four of these very large instances. For example, in instance RMc-T8-L5 MSSO-BlockIP took 151,979 seconds (1 day, 18 hours, 12 minutes, and 57 seconds) of CPU time, while CPLEX variant (1) required almost 20 days of CPU. Because of this excessive amount of time with CPLEX, some runs (for the largest instances) were stopped early and the overall CPU time was estimated by comparing the total number of arithmetic operations needed for the Cholesky factorizations (reported in the CPLEX log file) with the value in CPLEX variant (1) for instance RMc-T8-L5.

Looking at MSSO-BlockIP results in Table 6, it can be observed that variant RMc-Tx-Ly outperformed variant RM-Tx-Ly in only two out of the seven instances, with CPU times being much higher in the other five. It is worth pointing out that the opposite is observed when using a stochastic programming-based decomposition algorithm (as in Escudero et al., 2013) for the revenue management compact model (26) that is detailed in Section B.2 of Appendix B. The rationale behind this is that the higher sizes of the classical tightening approaches for non-IPMs usually degrade the matrix sparsity.

As with the network design instances, MSSO-BlockIP was also much more efficient than CPLEX in terms of memory requirements, which can be observed in Table 7. This table shows the gigabytes of RAM required by each method in a subset of the largest revenue management instances. MSSO-BlockIP also required a fraction of the memory used by CPLEX and, thus, could be run on much smaller hardware.

Table 7
Memory requirements (in gigabytes of RAM) for revenue management instances.

Instance	MSSO-BlockIP	CPLEX(1)	CPLEX(2)	CPLEX(3)
RMc-T10-L3	23	102	133	166
RM-T10-L3	21	81	102	480
RMc-T8-L5	83	410	469	481
RM-T8-L5	77	317	468	454
RMc-T6-L10	92	408	438	331
RM-T6-L10	87	468	452	307

(1) CPLEX v20.1 solved the split-variable formulation (2), with default aggregator.
(2) CPLEX v20.1 solved the split-variable formulation (2), deactivating default aggregator.
(3) CPLEX v20.1 solved the compact formulation (1), with default aggregator.

Since all previous executions have been performed with $\phi = 1$, to check the effect of ϕ (the number of terms of (16) used in the preconditioner) we ran the first eight RM instances of Table 5 with $\phi \in \{2, 3\}$ (the results for $\phi = 1$ are those in the Table 6). The

Table 8
Results for some revenue management instances for different number of terms (ϕ) in the preconditioner. Fastest execution in boldface.

Instance	$\phi = 1$			$\phi = 2$			$\phi = 3$		
	it.	PCG	CPU	it.	PCG	CPU	it.	PCG	CPU
RMc-T5-L5	87	2136	**83**	95	1574	105	96	1270	121
RM-T5-L5	183	15,182	485	108	1645	**106**	110	1411	126
RMc-T8-L3	143	5706	**884**	153	4323	1145	146	3221	1215
RM-T8-L3	239	6278	1040	244	3609	**991**	249	2940	1134
RMc-T3-L100	210	9866	**4968**	197	6940	5805	217	5988	7030
RM-T3-L100	249	8726	**4558**	274	6084	5433	270	5000	6163
RMc-T4-L25	265	11,695	**10,049**	296	8365	11,509	255	6670	12,602
RM-T4-L25	733	7244	**9479**	759	5889	11,077	768	4358	11,310

Table 9
Results for some revenue management instances, with optimality tolerance 10^{-3}, for different number of terms (ϕ) in the preconditioner. Fastest execution in boldface.

Instance	$\phi = 1$			$\phi = 2$			$\phi = 3$		
	it.	PCG	CPU	it.	PCG	CPU	it.	PCG	CPU
RMc-T5-L5	131	3989	**146**	140	3107	196	136	2427	217
RM-T5-L5	230	19,058	559	140	2931	**176**	141	2383	199
RMc-T8-L3	201	13,471	**1908**	213	10,297	2546	208	7798	2788
RM-T8-L3	298	9191	**1350**	403	10,942	2652	325	5012	1813
RMc-T3-L100	226	22,371	**10,133**	227	15,401	12,084	237	14,166	15,842
RM-T3-L100	254	22,805	**10,289**	277	18,427	14,424	278	14,539	16,307
RMc-T4-L25	286	32,151	**22,602**	286	21,907	26,871	282	17,224	30,425
RM-T4-L25	494	22,896	**17,492**	506	17,348	22,565	515	13,998	25,961

results are shown in Table 8. We see that by increasing ϕ, the overall number of PCG iterations was effectively reduced, but not the CPU time, and $\phi = 1$ provided the fastest execution (except for instances RM-T5-L5 and RM-T8-L3, where $\phi = 2$ was the best choice).

To observe the effect of ϕ when tighter optimality tolerances are requested, we repeated the same runs but reducing the optimality tolerance to 10^{-3}. The results are shown in Table 9. We see that the number of PCG iterations and CPU time significantly increased with the tighter optimality tolerance. But, as in the previous Table 8, $\phi = 1$ was still the best option in most cases (only $\phi = 2$ was a better choice for instance RM-T5-L5). Therefore, it can be concluded that, in general, the best option for MSSO-BlockIP is to consider $\phi = 1$.

6. Conclusions

The new approach for multistage stochastic optimization introduced in this work, which is based on a specialized interior-point method for primal block-angular problems, has proven to be very effective for solving huge problems. Unlike previous MSSO techniques, the new method can deal with both operational and strategic uncertainties, and it can solve both linear and (convex separable) quadratic problems. The method relies on the particular structure of preconditioner E (discussed in Section 4.1), which allows efficiently solving systems of equations. The extensive computational experience reported (using two applications: supply network design and revenue management) shows that the MSSO-BlockIP package—which implements the new method—outperformed one of the best state-of-the-art solvers by a significant margin.

This work could be extended in several ways. One line of research could apply the new method to problems other than supply network design and revenue management. A second line of work would be extending this new approach to general convex separable MSSO (that is, problems with nonlinear objective functions and positive diagonal Hessian matrices), for which, in practice, no solution techniques are available for huge problems.

Acknowledgments

J. Castro was supported by the MCIN/AEI/FEDER grant RTI2018-097580-B-I00. L.E. Escudero was supported by the MCIN/AEI/10.13039/501100011033 grant PID2021-122640OB-I00. J.F. Monge was supported by the MCIN/AEI/10.13039/501100011033/ERDF grants PID2019-105952GB-I00 and PID2021-122344NB-I00, and by PROMETEO/2021/063 grant funded by the government of the Valencia Community, Spain.

Appendix A. Supply network design under uncertainty: models

The multistage strategic supply network design problem considered in this work deals with strategic and operational uncertainties. The two-stage stochastic trees rooted at the strategic nodes represent the operational uncertainty that is realized in the second stage scenarios. As for the revenue management pilot application, see Appendix B, two mathematically equivalent models are considered, namely, the compact model and the split-variable formulation.

There are state strategic variables at the nodes as well as local strategic ones. The state variables link two consecutive stages (i.e., the strategic node that belongs to and their immediate successors). The plant capacity expansion in the strategic nodes is represented by *local* strategic variables. The actual capacity (a *state* strategic variable) is represented by the step variables modeling object that ensures the strategic nodes are linked only between consecutive stages.

A1. Introduction and notation

The strategic decision variables are related to the plant capacity, such that the *local* ones are the initial plant capacity and its extensions; and the *state* variables represent the resulting plant capacity as the link between the strategic nodes and their immediate successors. Therefore, the strategic variables belong to the nodes of the Hamiltonian path from root node 1 up to a node, viz., ω in

the last stage of the multistage scenario tree. The uncertain strategic parameters are the initial capacity and expansion unit costs and the plant capacity residual unit values. On the other hand, the operational decision variables are related to the raw materials being supplied and their transportation from the suppliers to the plants, their processing in the plants, and the transportation of the manufactured end product from the plants to the customer centers. The uncertain operational parameters cover all stages along the time horizon, and they are the cost of supplying, transporting and processing the raw materials in the plants, the processing coefficients in the plants, the end product transporting cost from the plants to the customer centers, and the demand from those centers. This operational uncertainty is captured in scenarios that are represented as nodes in the second stage of the stochastic two-stage trees rooted at the strategic nodes.

Let the notation of the different elements, usually capital letters and the symbol $\overline{(.)}$, denote data, while lowercase and Greek letters denote variables. Recall from Section 5.2 that $\mathcal{S}$, $\mathcal{P}$, $\mathcal{C}$, and A denote, respectively, the sets of raw materials to be supplied, plants, customer centers, and arcs connecting the nodes associated to $\mathcal{S} \cup \mathcal{P} \cup \mathcal{C}$.

Deterministic data

$\hat{x}_j^0$, Current existing capacity of plant j, $j \in \mathcal{P}$, at the beginning of the time horizon. *Note:* $\hat{x}_j^0 = 0 \, \forall j \in \mathcal{P}$ means that the infrastructure system is anew.

$\overline{x}_j$, Maximum capacity that is allowed for plant j, $j \in \mathcal{P}$.

B^t, budget available for plant investment (either initial capacity or extension) at stage t, $t \in \mathcal{T}$.

$\rho \in (0, 1)$, parameter that gives the fraction of the investment in any plant capacity over the overall plant investment at any stage.

M_j^t, plant unit maintenance cost, $j \in \mathcal{P}$, at stage t, $t \in \mathcal{T}$.

$\overline{y}_i$, maximum stock volume of raw material i that can be supplied under any operational scenario at any stage, $i \in \mathcal{S}$.

$\overline{y}_{ij}^t$, upper bound on the flow from node i to node j, $(ij) \in \mathcal{A}$, being raw material for $i \in \mathcal{S}$, $j \in \mathcal{P}$ and end product for $i \in \mathcal{P}$, $j \in \mathcal{C}$.

M_d, unit penalization of demand shortfall under any operational scenario at any stage, $k \in \mathcal{C}$.

Strategic uncertain data in node n, $n \in \mathcal{N}$

C_j^n, unit cost of the investment on the initial capacity or its expansion in plant j, $j \in \mathcal{P}$, in strategic node n.

V_j^n, unit residual value of the capacity investment on plant j, $j \in \mathcal{P}$, in node n at the end of the time horizon (i.e., $n \in \mathcal{N}_{|\mathcal{T}|}$). Usually, $V_j^n < C_j^{n'}$, $n' \in \mathcal{N}$.

Operational uncertain data under scenario π, $\pi \in \Pi_t$, $t \in \mathcal{T}$

C_{ij}^π, unit cost of raw material i supplying, its transporting to and processing in plant j, $i \in \mathcal{S}$, $j \in \mathcal{P}$, and unit cost of transporting the end product from plant i to customer center j, $i \in \mathcal{P}$, $j \in \mathcal{C}$, under scenario π, provided that $(i, j) \in \mathcal{A}$.

P_{ij}^π, capacity requirement of plant j to process a unit of raw material i, $i \in \mathcal{S}$, $j \in \mathcal{P}$, under scenario π, provided that $(ij) \in \mathcal{A}$.

D_k^π, end product demand from customer center k, $k \in \mathcal{C}$, under scenario π.
Note: Under the assumption that the parameters C_{ij}^π, P_{ij}^π and D_k^π are not stagewise-dependent but stage-dependent ones, it means that they do not depend on the plants' capacity.

State strategic variables in node n, $n \in \mathcal{N}$

x_i^n, end product capacity of plant i, $i \in \mathcal{P}$, and raw material i stock volume, $i \in \mathcal{S}$, that is available in strategic node n. Observe that x_i^n, $i \in \mathcal{P}$, is the result of the cumulated investment that is carried out in the previous strategic nodes back to stage $t = 1$, including node n. (i.e., set $\forall n' \in \mathcal{A}^n$).
Therefore, note that x_i^n, $n \in \mathcal{N}_{|\mathcal{T}|}$, is the capacity investment in plant i, $i \in \mathcal{P}$, that results at the end of the time horizon.

Local strategic variables in node n, $n \in \mathcal{N}$

δ_j^n, end product initial capacity of plant j or its expansion, $j \in \mathcal{P}$, to be invested in strategic node n at stage t^n.

$(x_\rho)_j^n$, fraction of the total plant capacity that has not been considered while deciding the capacity of plant j, $j \in \mathcal{P}$, by strategic node n during stage t^n, it is a slack variable.

$(b_a)^n$, unused budget for plant investment (either initial capacity or expansion) in strategic node n during stage t^n, it is a slack variable.

Operational variables under scenario π, $\pi \in \Pi_{t^n}$, $n \in \mathcal{N}$

$y_{ij}^{n,\pi}$, flow from node i to node j, $(ij) \in \mathcal{A}$, being raw material for $i \in \mathcal{S}$, $j \in \mathcal{P}$, and end product for $i \in \mathcal{P}$, $j \in \mathcal{C}$, under scenario π.

$(y_a)_j^{n,\pi}$, unused capacity of plant j, $j \in \mathcal{P}$, under scenario π.

$(d_a)_k^{n,\pi}$, demand shortfall from customer center k under scenario π, $k \in \mathcal{C}$.
Note: For levelizing the end product demand shortfall in the customer centers, the quadratic of $(d_a)_k^\pi$ is M_d-penalized in the objective function of the model below.

A2. Strategic multistage operational two-stage split-variable formulation

This type of formulations is more suitable for IPM solvers, see Castro & de la Lama-Zubirán (2020).

State strategic split-variables in node n, $n \in \mathcal{N}$

$x_{j,n}^s$, copy of x_j^n where n is the strategic node that roots the embedded strategic two-stage tree, where $\{s\}$ is the set of second stage nodes, $\forall j \in \mathcal{P}$, $s \in \mathcal{S}_1^n$, $n \in \mathcal{N} : t^n < |\mathcal{T}|$.

$x_{j,n}^\pi$, copy of x_j^n where n is the strategic node that roots the operational two-stage tree, where $\{\pi\}$ is the set of second stage nodes, $\forall j \in \mathcal{P}$, $\pi \in \Pi_{t^n}$, $n \in \mathcal{N}$.

The model can be expressed

$$\min \sum_{n \in \mathcal{N}} w^n \Big[\sum_{j \in \mathcal{P}} \big(M_j^{t^n} x_j^n + C_j^n \delta_j^n \big) + \sum_{\pi \in \Pi_{t^n}} w^\pi \Big(\sum_{(ij) \in \mathcal{A}} C_{ij}^\pi y_{ij}^{n,\pi} + \sum_{k \in \mathcal{C}} M_d ((d_a)_k^{n,\pi})^2 \Big) \Big] - \sum_{j \in \mathcal{P}} \sum_{n \in \mathcal{N}_{|\mathcal{T}|}} w^n V_j^n x_j^n \tag{24a}$$

The objective function (24a) minimizes the expected cost of the plant investment and their maintenance, the expected cost of the operational activity and the quadratic penalization of demand shortfall in the scenarios, minus the residual value of the plants' capacity at the end of the time horizon.

The strategic and operational split-variables definition is represented in the constraint system to be expressed as

$$x_j^n - x_{j,n}^{s(1)} = 0, \; x_{j,n}^{s-1} - x_{j,n}^s = 0 \; \forall j \in \mathcal{P}, \; s \in \mathcal{S}_1^n \setminus \{s(1)\}, \; n \in \mathcal{N} : t^n < |\mathcal{T}| \tag{24b}$$

$$x_{j,n}^\bullet - x_{j,n}^{\pi(1)} = 0, \; x_{j,n}^{\pi-1} - x_{j,n}^\pi = 0 \; \forall j \in \mathcal{P}, \; \pi \in \Pi_{t^n} \setminus \{\pi(1)\}, \; n \in \mathcal{N}, \tag{24c}$$

where (24b) define the split-variables of the strategic plant investment variables x_i^n, and (24c) does the same for the operational

copies of those variables. Note that $x^{\bullet}_{j,n} \equiv x^{s(\ell)}_{j,n}$ for $n \in \mathcal{N} : t^n < |\mathcal{T}|$ and $x^{\bullet}_{j,n} \equiv x^n_j$ for $n \in \mathcal{N}_{|\mathcal{T}|}$.

The other constraints system for the strategic multistage operational two-stage problem can be expressed

$$(x^n_{j,\sigma^n})_{:t^n>1} + \delta^n_j - x^n_j = 0 - (\hat{x}^{\sigma^n}_j)_{:t^n=1} \ \forall j \in \mathcal{P},\ n \in \mathcal{N} \tag{24d}$$

$$\sum_{i \in \mathcal{S}:(ij)\in\mathcal{A}} P^{\pi}_{ij} y^{n,\pi}_{ij} + (y_a)^{n,\pi}_j - x^{\pi}_{j,n} = 0 \ \forall j \in \mathcal{P},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{24e}$$

$$x^n_j + (x_\rho)^n_j - \rho \sum_{j' \in \mathcal{P}} x^n_{j'} = 0 \ \forall j \in \mathcal{P},\ n \in \mathcal{N} \tag{24f}$$

$$\sum_{j \in \mathcal{P}} C^n_j \delta^n_j + (b_a)^n - B^{t^n} = 0 \ \forall n \in \mathcal{N} \tag{24g}$$

$$\sum_{j \in \mathcal{P}:(ij)\in\mathcal{A}} y^{n,\pi}_{ij} = \bar{y}_i \ \forall i \in \mathcal{S},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{24h}$$

$$\sum_{i \in \mathcal{S}:(ij)\in\mathcal{A}} P^{\pi}_{ij} y^{n,\pi}_{ij} - \sum_{k \in \mathcal{C}:(jk)\in\mathcal{A}} y^{n,\pi}_{jk} = 0 \ \forall j \in \mathcal{P},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{24i}$$

$$\sum_{j \in \mathcal{P}:(jk)\in\mathcal{A}} y^{n,\pi}_{jk} + (d_a)^{n,\pi}_k = D^{\pi}_k \ \forall k \in \mathcal{C},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{24j}$$

$$0 \le \delta^n_j,\ x^n_j \le \bar{x}_j \ \forall j \in \mathcal{P},\ n \in \mathcal{N} \tag{24k}$$

$$0 \le (x_\rho)^n_j \ \forall j \in \mathcal{P},\ n \in \mathcal{N} \tag{24l}$$

$$0 \le (b_a)^n \ \forall n \in \mathcal{N} \tag{24m}$$

$$0 \le y^{n,\pi}_{ij} \le \bar{y}^{t^n}_{ij} \ \forall (ij) \in \mathcal{A},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{24n}$$

$$0 \le (y_a)^{n,\pi}_j \ \forall j \in \mathcal{P},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{24o}$$

$$0 \le (d_a)^{n,\pi}_k \ \forall k \in \mathcal{C},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{24p}$$

The strategic constraints (24d) introduce the step variable modeling object for plant capacity. It is assumed that the initial capacity or expansion δ^n_j in plant j from x^n_{j,σ^n} up to x^n_j is performed at the beginning of stage t^n. The strategic operational constraints (24e) bound the operational consumption of raw material volume in each plant. The state strategic constraints (24f) keep a ρ-based equilibrium on the plants' capacity. The local strategic constraints (24g) force plant investment budget limitations. The operational constraint (24h) bound the raw material volume to supply under the operational scenarios in order to cover the manufacturing needs in the plants, without keeping stock volume at the end of the stages. The operational constraints (24i) balance the end product volume manufactured in each plant with the total volume distributed to the customer centers under the operational scenarios. The operational constraints (24j) balance the end product manufactured in the plant set for each customer center with its demand and, so, defining the demand shortfall $(d_a)^{n,\pi}_k$, if any, under the operational scenarios.

Relationship between the split-variable formulation (24) and split-variable meta formulation (2)

The vectors of the variables of meta formulation (2) can be expressed by the following sets of variables of formulation (24), for $j \in \mathcal{P}$, $\pi \in \Pi_{t^n}$, $(ij) \in \mathcal{A}$, $i \in \mathcal{S}$, $k \in \mathcal{C}$, $n \in \mathcal{N}$:

- $\left(x^n_{\sigma^n} = (x^n_{j,\sigma^n})\right)_{t^n>1}$, $x^n = (x^n_j)$, $z^n = (\delta^n_j, (x_\rho)^n_j, (b_a)^n)$,
- $\left(x^{\pi}_{\sigma^n} = (x^{n,\pi}_{j,\sigma^n})\right)_{t^n>1}$, $y^{\pi}_n = (y^{n,\pi}_{ij}, (y_a)^{n,\pi}_j, (d_a)^{n,\pi}_k)$,

A3. Strategic multistage operational two-stage-based compact model

This type of model is more suitable for primal and dual Simplex solvers, and also for IPM solvers if the number of dense columns is not large. It can be expressed

$$\min \quad (24a) \tag{25a}$$

$$\text{s. to} \quad (\hat{x}^{\sigma^n}_j)_{:t^n=1} + (x^{\sigma^n}_j)_{:t^n>1} + \delta^n_j - x^n_j = 0 \ \forall j \in \mathcal{P},\ n \in \mathcal{N} \tag{25b}$$

$$\sum_{i \in \mathcal{S}:(ij)\in\mathcal{A}} P^{\pi}_{ij} y^{\pi}_{ij} \le x^n_j \ \forall j \in \mathcal{P},\ \pi \in \Pi_{t^n},\ n \in \mathcal{N} \tag{25c}$$

$$x^n_j \le \rho \sum_{j' \in \mathcal{P}} x^n_{j'} \ \forall j \in \mathcal{P},\ n \in \mathcal{N} \tag{25d}$$

$$\sum_{j \in \mathcal{P}} C^n_j \delta^n_j \le B^{t^n} \ \forall n \in \mathcal{N} \tag{25e}$$

$$(24h)\text{–}(24p) \tag{25f}$$

The constraints (25b) introduce the state and local strategic variables. The constraints (25c), (25d) and (25e) bound the processing volume of the required raw materials in the plants under the operational scenarios, bound the plants' capacity and bound the plants' investment, respectively. The other constraints are as in the system (24h)–(24p) for the split-variable formulation.

Appendix B. Revenue management under uncertainty: models

B1. Introduction and notation

The revenue management model used in this work for maximizing expected income is taken from DeMiguel & Mishra (2006). The following notation is used to present the problem in a multistage setting:

Sets

$\mathcal{R}$, resources.
$\mathcal{I}$, bundles.
$\mathcal{J}$, fare classes.
$\mathcal{I}_r$, bundles using resource r, $r \in \mathcal{R}$.

Deterministic parameters

f_{ij}, fare of bundle-class ij, $i \in \mathcal{I}$, $j \in \mathcal{J}$.
C_r, capacity on resource r, $r \in \mathcal{R}$.

Uncertain parameters

d^n_{ij}, demand for bundle-class ij in stage t^n at node n, $i \in \mathcal{I}$, $j \in \mathcal{J}$, $n \in \mathcal{N}$.

Variables for $i \in \mathcal{I}$, $j \in \mathcal{J}$, $n \in \mathcal{N}$

b^n_{ij}, number of accepted bookings for bundle-class ij at stage t^n in node n. *Note:* It is a local strategic variable.
B^n_{ij}, cumulative number of accepted bookings of bundle-class ij along the path from root node 1 to node n. *Note:* It is a state strategic variable.

B2. Strategic multistage revenue management compact model

The last stage satisfaction capacity-based model can be expressed

$$\max \sum_{n\in\mathcal{N}} w^n \sum_{i\in\mathcal{I}} \sum_{j\in J} f_{ij} b_{ij}^n \tag{26a}$$

$$\text{s. to } B_{ij}^{\sigma^n} + b_{ij}^n = B_{ij}^n \ \forall i \in \mathcal{I},\ j \in \mathcal{J},\ n \in \mathcal{N} \tag{26b}$$

$$\sum_{i\in\mathcal{I}_r} \sum_{j\in\mathcal{J}} B_{ij}^n \le C_r \ \forall r \in \mathcal{R},\ n \in \mathcal{N}_{|\mathcal{T}|} \tag{26c}$$

$$0 \le b_{ij}^n \le d_{ij}^n \ \forall i \in \mathcal{I},\ j \in \mathcal{J},\ n \in \mathcal{N}. \tag{26d}$$

Note that the constraints (26c) impose that the total number of accepted bookings along the whole booking horizon is restricted by the resource capacity.

A tightening formulation of model RM (26) (which in Tables 5–7 is denoted as RMc) that is useful for some approaches consists of replacing $\forall r \in \mathcal{R},\ n \in \mathcal{N}_{|\mathcal{T}|}$ with $\forall r \in \mathcal{R},\ n \in \mathcal{N}$ in constraints (26c) (see Escudero et al., 2013) for a stochastic dynamic programming decomposition algorithm for problem solving. Note that the new constraints impose that the overall number of accepted bookings in each stage's nodes along the booking horizon is restricted by the resource capacity.

B3. Strategic multistage revenue management split-variable formulation

As an alternative to the compact model RM (26) that is more amenable to IPMs, we use the split-variable formulation RM (27). It requires the following variable

$B_{ij,n}^s$, copy of B_{ij}^n where n is the node that roots the two-stage tree where $\{s\}$ is the set of second stage nodes, $s \in \mathcal{S}_1^n,\ n \in \mathcal{N} : t^n < |\mathcal{T}|$.

The formulation can be expressed

$$\max \sum_{n\in\mathcal{N}} w^n \sum_{i\in\mathcal{I}} \sum_{j\in J} f_{ij} b_{ij}^n \tag{27a}$$

$$\text{s. to } B_{ij}^n - B_{ij,n}^{s(1)} = 0,\ B_{ij,n}^{s-1} - B_{ij,n}^s = 0 \quad \forall s \in \mathcal{S}_1^n \setminus \{s(1)\},\ i \in \mathcal{I},\ j \in \mathcal{J},\ n \in \mathcal{N} : t^n < |\mathcal{T}| \tag{27b}$$

$$B_{ij}^n = B_{ij,\sigma^n}^n + b_{ij}^n \ \forall i \in \mathcal{I},\ j \in \mathcal{J},\ n \in \mathcal{N} \tag{27c}$$

$$\sum_{i\in\mathcal{I}_r} \sum_{j\in\mathcal{J}} B_{ij}^n \le C_r \ \forall r \in \mathcal{R},\ n \in \mathcal{N}_{|\mathcal{T}|} \tag{27d}$$

$$0 \le b_{ij}^n \le d_{ij}^n \ \forall i \in \mathcal{I},\ j \in \mathcal{J},\ n \in \mathcal{N}. \tag{27e}$$

Notice that here the constraints (27d) impose that the total number of accepted bookings along the whole booking horizon is restricted by the resource capacity. Additionally, for the RM model (26), a tightening of formulation RM (27) (that in Tables 5–7 is denoted as RMc) consists of replacing $\forall r \in \mathcal{R},\ n \in \mathcal{N}_{|\mathcal{T}|}$ with $\forall r \in \mathcal{R},\ n \in \mathcal{N}$ in constraints (27d).

References

Alonso-Ayuso, A., Escudero, L. F., Guignard, M., & Weintraub, A. (2020). On dealing with strategic and tactical decision levels in forestry planning under uncertainty. *Computers and Operations Research, 115*. https://doi.org/10.1016/j.cor.2019.104836. Article 104836

Bertsekas, D. P., & Tsitsiklis, J. N. (1995). *Parallel and distributed computation*. Englewood Cliffs: Prentice-Hall.

Birge, J. R. (1985). Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research, 33*, 989–1007. https://doi.org/10.1287/opre.33.5.989.

Birge, J. R., & Louveaux, F. V. (2011). *Introduction to stochastic programming* (2nd ed.). New York: Springer. https://doi.org/10.1007/978-1-4614-0237-4.

Blomvall, J., & Lindberg, P. O. (2002). A Riccati-based primal interior point solver for multistage stochastic programming. *European Journal of Operational Research, 143*, 452–461. https://doi.org/10.1016/S0377-2217(02)00301-6.

Bocanegra, S., Castro, J., & Oliveira, A. R. L. (2013). Improving an interior-point approach for large block-angular problems by hybrid preconditioners. *European Journal of Operational Research, 231*, 263–273. https://doi.org/10.1016/j.ejor.2013.04.007.

Cadarso, L., Escudero, L. F., & Marín, A. (2018). On strategic multistage operational two-stage stochastic 0–1 optimization for the rapid transit network design problem. *European Journal of Operational Research, 271*, 577–593. https://doi.org/10.1016/j.ejor.2018.05.041.

Castro, J. (2000). A specialized interior-point algorithm for multicommodity network flows. *SIAM Journal on Optimization, 10*, 852–877. https://doi.org/10.1137/S1052623498341879.

Castro, J. (2007). An interior-point approach for primal block-angular problems. *Computational Optimization and Applications, 36*, 195–219. https://doi.org/10.1007/s10589-006-9000-1.

Castro, J. (2016). Interior-point solver for convex separable block-angular problems. *Optimization Methods and Software, 31*, 88–109. https://doi.org/10.1080/10556788.2015.1050014.

Castro, J., & Cuesta, J. (2011). Quadratic regularizations in an interior-point method for primal block-angular problems. *Mathematical Programming A, 130*, 415–445. https://doi.org/10.1007/s10107-010-0341-2.

Castro, J., & de la Lama-Zubirán, P. (2020). A new interior-point approach for large two-stage stochastic problems. *Optimization Methods and Software*. https://doi.org/10.1080/10556788.2020.1841190.

DeMiguel, V., & Mishra, N. (2006). *What multistage stochastic programming can do for network revenue management*. London Business School. http://facultyresearch.london.edu/docs/DeMiguel-Mishra-Manuscript.pdf. Accessed 26 October 2021

Dupacova, J., Consigli, G., & Wallace, S. W. (2000). Scenarios for multistage stochastic programs. *Annals of Operations Research, 100*, 25–53. https://doi.org/10.1023/A:1019206915174.

Escudero, L. F., Garín, A., & Unzueta, A. (2017). Scenario cluster Lagrangean decomposition for risk averse in multistage stochastic optimization. *Computers and Operations Research, 85*, 154–171. https://doi.org/10.1016/j.cor.2017.04.007.

Escudero, L. F., & Monge, J. F. (2018). On capacity expansion planning under strategic and operational uncertainties based on stochastic dominance risk averse management. *Computational Management Science, 15*, 479–500. https://doi.org/10.1007/s10287-018-0318-9.

Escudero, L. F., & Monge, J. F. (2021). On multistage multiscale stochastic capacitated multiple allocation hub network expansion planning. *Mathematics, 9*. https://doi.org/10.3390/math9243177. Article 3177

Escudero, L. F., Monge, J. F., Romero-Morales, D., & Wang, J. (2013). Expected future value decomposition based bid price generation for large-scale network revenue management. *Transportation Science, 47*, 181–197. https://doi.org/10.1287/trsc.1120.0422.

Glanzer, M., & Pflug, G. C. (2020). Multiscale stochastic optimization: Modeling aspects and scenario generation. *Computational Optimization and Applications, 75*, 1–34. https://doi.org/10.1007/s10589-019-00135-4.

Gondzio, J. (2012). Interior point methods 25 years later. *European Journal of Operational Research, 218*, 587–601. https://doi.org/10.1016/j.ejor.2011.09.017.

Gondzio, J. (2013). Convergence analysis of an inexact feasible interior point method for convex quadratic programming. *SIAM Journal on Optimization, 23*, 1510–1527. https://doi.org/10.1137/120886017.

Gondzio, J., González-Brevis, P., & Munari, P. (2016). Large-scale optimization with the primal-dual column generation method. *Mathematical Programming Computation, 8*, 47–82. https://doi.org/10.1007/s12532-015-0090-6.

Gondzio, J., & Grothey, A. (2007). Solving nonlinear portfolio optimization problems with the primal-dual interior point method. *European Journal of Operational Research, 181*, 1019–1029. https://doi.org/10.1016/j.ejor.2006.03.006.

Heitsch, H., & Römisch, W. (2009). Scenario tree modeling for multistage stochastic programs. *Mathematical Programming A, 118*, 371–406. https://doi.org/10.1007/s10107-007-0197-2.

Henrion, H., & Römisch, W. (2018). Problem-based optimal scenario generation and reduction in stochastic programming. *Mathematical Programming B*. https://doi.org/10.1007/s10107-018-1337-6.

Hoyland, K., Kaut, M., & Wallace, S. W. (2003). A heuristic for moment-matching scenario generation. *Computational Optimization and Applications, 24*, 169–185. https://doi.org/10.1023/A:1021853807313.

Hübner, J., Schmidt, M., & Steinbach, M. (2017). A distributed interior-point KKT solver for multistage stochastic optimization. *INFORMS Journal on Computing, 29*, 612–630. https://doi.org/10.1287/ijoc.2017.0748.

Kaut, M., Midthun, K. T., Werner, A. S., Tomasgard, A., Hellemo, L., & Fodstad, M. (2014). Multi-horizon stochastic programming. *Computational Management Science, 15*, 479–500. https://doi.org/10.1007/s10287-013-0182-6.

Kibaek, K., & Zavala, V. M. (2018). Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. *Mathematical Programming Computation, 10*, 225–266. https://doi.org/10.1007/s12532-017-0128-z.

Leövey, H., & Römisch, W. (2015). Quasi-monte carlo methods for linear two-stage stochastic programming. *Mathematical Programming B, 151*, 315–345. https://doi.org/10.1007/s10107-015-0898-x.

Li, Z., & Floudas, C. (2016). Optimal scenario reduction framework based on distance of uncertainty distribution and output performance: II. Sequential reduction. *Computers and Chemical Engineering, 84*, 599–610. https://doi.org/10.1016/j.compchemeng.2015.05.010.

Lustig, I. J., Mulvey, J. M., & Carpenter, T. J. (1991). Formulating stochastic programs for interior point methods. *Operations Research, 39*, 757–770. https://doi.org/10.1287/opre.39.5.757.

Maggioni, F., Allevi, E., & Tomasgard, A. (2020). Bounds in multi-horizon stochastic programs. *Annals of Operations Research, 292*, 605–625. https://doi.org/10.1007/s10479-019-03244-9.

Mészáros, C. (2016). Exploiting hardware capabilities in interior point methods. *Optimization Methods and Software, 31*, 435–443. https://doi.org/10.1080/10556788.2015.1104677.

Ng, E., & Peyton, B. W. (1993). Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM Journal on Scientific Computing, 14*, 1034–1056. https://doi.org/10.1137/0914063.

Pflug, G. C., & Pichler, A. (2014). *Multistage stochastic optimization*. Switzerland: Springer. https://doi.org/10.1007/978-3-319-08843-3.

Pflug, G. C., & Pichler, A. (2015). Dynamic generation of scenario trees. *Computational Optimization and Applications, 62*, 641–668. https://doi.org/10.1007/s10589-015-9758-0.

Rockafellar, R. T., & Wets, R. J. B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research, 16*, 119–147. https://doi.org/10.1287/moor.16.1.119.

Ruszczyński, A. (1993). *Interior point methods in stochastic programming*. Laxenburg, Austria: International Institute for Applied Systems Analysis. http://pure.iiasa.ac.at/id/eprint/3804/1/WP-93-008.pdf. Accessed online 26 October 2021

Steinbach, M. (2001). Hierarchical sparsity in multistage convex stochastic programs. In S. Uryasev, & P. M. Pardalos (Eds.), *Stochastic optimization: Algorithms and applications* (pp. 385–410). Kluwer Academic Publishers. https://doi.org/10.1007/978-1-4757-6594-6_16.

Talluri, K. T., & van Ryzin, G. J. (2004). *The theory and practice of revenue management*. New York: Springer. https://doi.org/10.1007/b139000.

Tanneau, M., Anjos, M. F., & Lodi, A. (2021). Design and implementation of a modular interior-point solver for linear optimization. *Mathematical Programming Computation, 13*, 509–551. https://doi.org/10.1007/s12532-020-00200-8.

Werner, A. S., Pichler, A., Midthun, K. T., Hellemo, L., & Tomasgard, A. (2013). Risk measures in multi-horizon scenarios trees. In R. Kovacevic, G. C. Pflug, & M. T. Vespucci (Eds.), *Handbook of risk management in energy production and trading* (pp. 177–201). New York: Springer. https://doi.org/10.1007/978-1-4614-9035-7.

Wright, S. J. (1997). *Primal-Dual interior-point methods*. Philadelphia, PA: SIAM. https://doi.org/10.1137/1.9781611971453.