Optimal scenario set partitioning for multistage stochastic programming with the progressive hedging algorithm

Pierre-Luc Carpentier[a,b,c,*], Michel Gendreau[a,b,c], Fabian Bastin[a,b,d]

[a] *NSERC/Hydro-Québec Industrial Research Chair on the Stochastic Optimization of Electricity Generation, Montréal, Québec, Canada*
[b] *CIRRELT, Université de Montréal, C.P. 6128, Succ. Centre-ville, Montréal, Québec, Canada H3C 3J7*
[c] *Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, Montréal, Québec, Canada*
[d] *Department of Computer Science and Operations Research, Université de Montréal, Montréal, Québec, Canada*

Abstract

In this paper, we propose a new approach to reduce the total running time (RT) of the progressive hedging algorithm (PHA) for solving multistage stochastic programs (MSPs) defined on a scenario tree. Instead of using the conventional scenario decomposition scheme, we apply a multi-scenario decomposition scheme and partition the scenario set in order to minimize the number of non-anticipativity constraints (NACs) on which an augmented Lagrangian relaxation (ALR) must be applied. Our partitioning method is a heuristic algorithm that takes into account the complex branching structure of general multistage scenario trees. Minimizing the number of relaxed NACs (RNACs) enhances the PHA's convergence rate by decreasing the variability of subproblems solutions at duplicated tree nodes. This is due to the fact that minimizing the RNACs reduces the anticipativity level of subproblems by increasing the branching level of subtrees. This makes RNACs easier to satisfy. Our partitioning method also reduces the total RT per iteration in two different ways. Firstly, it decreases the number of linear and quadratic penalty terms that need to be included in subproblems objective function.

*Corresponding author. Tel. +1 514 343 7575, fax. +1 514 343 7121.
Email address: pierre-luc.carpentier@polymtl.ca (Pierre-Luc Carpentier)

October 2, 2013

Secondly, it reduces the total number of duplicated decision variables and constraints at tree nodes that are associated with RNACs. The proposed method is tested on an hydroelectricity generation scheduling problem covering a 52 weeks planning horizon.

Keywords: Stochastic programming, Large scale optimization, Augmented Lagrangian, Scenario tree, Progressive hedging

---

1. Introduction

Practical applications of stochastic programming (SP) methods for solving uncertain optimization problems are quite numerous and cover a wide spectrum of domains (Dupačová , 2002; Rusczynzki and Shapiro , 2003). The book by Gassman and Ziemba (2013) presents different applications of these methods in energy, logistics and production planning, finance and telecommunications. Most of these applications contain one or several random parameters (e.g. inflows, prices, interest rates, yield, demand, electrical load, wind/solar generation) that are characterized by a (joint) continuous probability distribution. A popular approach to represent continuously distributed parameters in SP models consists in replacing the original continuous distribution by a discrete distribution possessing a finite number of possible outcomes (scenarios). This type of approximation leads to a scenario tree representation of uncertainty. Fig. 1a shows a simple example of a scenario tree with three stages, four scenarios and seven nodes. Different methods were proposed over the years for constructing a scenario tree from a set of synthetic or historical time series (e.g. Pflug , 2001; Høyland and Wallace , 2001; Latorre et al. , 2007). Heitsch and Römisch (2009) proposed a construction method based the theoretical results of Heitsch et al. (2006). The SCENRED2 package of the General Algebraic Modeling System (GAMS) is a computer implementation of this technique.

Multistage stochastic programs (MSPs) defined on scenario trees can be reformulated into deterministic equivalent programs (DEPs) of finite size (variables, constraints). The number of decision variables and constraints is usually proportional to the number of nodes contained in the scenario tree. Therefore, the size of DEPs grows exponentially with the discretization level (number of branching stages, number of branches per stage) used to describe the original probability distribution. In general, most real-world MSPs cannot be solved directly using a commercial solver (e.g. GLPK,

Gurobi, XPress-MP, CPLEX) when an accurate representation of random parameter is used. The required amount of random access memory (RAM) is typically the main limiting factor when solving large-scale linear or quadratic programs directly.

Over the past decades, different decomposition methods were proposed in the literature for solving efficiently large-scale SPs (Rusczynzki , 2003; Sagastizábal , 2012). Solution methods based on Benders (1962) decomposition (e.g. Van Slyke and Wets , 1969; Birge , 1985; Birge and Louveaux , 1988; Pereira and Pinto , 1991; Laporte and Louveaux , 1993; Küchler and Vigerske , 2007; Carpentier et al. , 2013b) are widely used for solving linear MSPs. These methods use a stage-wise decomposition scheme and work by constructing convex and piecewise linear recourse functions. The progressive hedging algorithm (PHA) proposed by Rockafellar and Wets (1991) is another popular method for solving SPs defined on scenario trees. This method was applied successfully in various type of problems including electricity generation planning (Gonçalves et al. , 2011; Carpentier et al. , 2013a), network design (Crainic et al. , 2011), network flow (Mulvey and Vladimirou , 1991), lot-sizing (Haugen et al. , 2001) and resources allocation (Watson and Woodruff , 2011). To use the traditional version of the PHA, a decision vector $x_{t\omega} \in \mathbb{R}^m$ must be defined at each stage (time period) $t = 1, ..., T$ for all scenarios $\omega \in \Omega$ contained in the tree. All non-anticipativity constraints (NACs) must formulated explicitly as linear equality constraints

$$x_{t(n)\omega} = \hat{x}_n, \qquad \forall \omega \in \Omega, n \in \mathcal{N}^*_\omega : \lambda_{n\omega} \tag{1}$$

to ensure that feasible solutions are scenario-invariant at each tree node. The function $t(n)$ returns the stage index associated with tree node n, $\hat{x}_n \in \mathbb{R}^m$ is the decision vector at node n, $\lambda_{n\omega} \in \mathbb{R}^m$ is the vector of Lagrange multipliers associated with NACs for scenario ω at node n, $\mathcal{N}^*_\omega$ is a set of duplicated nodes that are visited by scenario ω. By definition, duplicated nodes are the ones that are visited by two scenarios or more. The traditional version of the PHA works by applying an augmented Lagrangian relaxation (ALR) on all constraints 1. Then, a scenario-decomposition scheme is applied on the resulting optimization problem. In practice, the running time of the PHA can become too slow to be applicable if the convergence rate is too low or if the running time per iteration is too important. This can happen if the number of NACs is too large or if scenarios differ too much. Indeed, the number of linear-quadratic penalty terms depends directly on the number of NACs to be satisfied. The required number of iterations to converge is sensitive to the

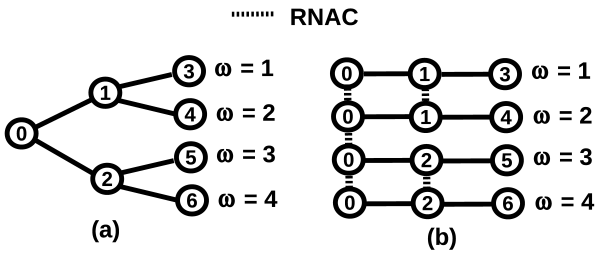

Figure 1: (a) Example of a scenario tree. (b) Illustration of the scenario-decomposition scheme.

variability among the subproblems solutions at tree nodes that are visited by two scenarios or more. The performance of the PHA is often reported to be sensitive to the penalty parameter choice. Unfortunately, tuning this parameter can be a time consuming task. Fig. 1b illustrates an application of the scenario-decomposition scheme on the scenario tree shown on Fig. 1a. In this example, $\mathcal{N}_1^* = \mathcal{N}_2^* = \{0, 1\}$ and $\mathcal{N}_3^* = \mathcal{N}_4^* = \{0, 2\}$. An ALR must be applied on the following NACs

$$x_{1,1} = x_{1,2} = x_{1,3} = x_{1,4} = \hat{x}_0, \tag{2}$$

$$x_{2,1} = x_{2,2} = \hat{x}_1,\ x_{2,3} = x_{2,4} = \hat{x}_2. \tag{3}$$

A different version of the PHA was proposed by Crainic et al. (2013) for solving two-stage stochastic network design problems. Instead of applying the traditional scenario-decomposition scheme, these authors applied a multi-scenario decomposition scheme. With their approach, each subproblem is a small two-stage SP (TSP) defined on a group of scenarios. These authors partition the scenario set Ω into disjoint scenario groups Ω_c for $c \in \mathcal{C}$. Scenarios are grouped according to their similarity or dissimilarity level. The results of this study show that using such a partitioning method enhanced the performance of the PHA for solving TSPs. However, their approach is designed specifically for solving TSPs and it could not be used directly when dealing with MSPs. Indeed, the similarity-based method proposed by Crainic et al. (2013) works well when TSPs are considered because the topology of two-stage scenario tree is quite simple. In any two-stage scenario trees, all leaves share the same ancestor node (the root) and, consequently, the total number of RNACs is always equal to the number of scenario groups. This is true no matter which scenarios are grouped together. Unfortunately,

this is usually not the case for multistage scenario trees (MSTs). Indeed, MSTs usually possess a complex branching structure and, because of this, the resulting number of RNACs will depend on which scenarios are grouped together. Fig. 2 shows a simple illustrative example where two different partitioning schemes are applied to the tree shown on Fig. 1a. For the scheme shown on Fig. 2a, the groups $c = 1$ and $c = 2$ are defined on scenarios $\Omega_1 = \{1, 2\}$ and $\Omega_2 = \{3, 4\}$, respectively. With this scheme, only the two following NACs need to be formulated explicitly (and relaxed)

$$\tilde{x}_{0,1} = \tilde{x}_{0,2} = \hat{x}_0$$

where $\tilde{x}_{nc}$ is the decision at node n in group c. The remaining NACs associated with nodes 1 and 2 are dealt with directly when solving subproblems defined on groups 1 and 2, respectively. For the scheme shown on Fig. 2b, the groups $c = 1$ and $c = 2$ are defined on scenarios $\Omega_1 = \{1, 3\}$ and $\Omega_2 = \{2, 4\}$, respectively. The following NACs need to be formulated explicitly

$$\tilde{x}_{0,1} = \tilde{x}_{0,2} = \hat{x}_0,\ \tilde{x}_{1,1} = \tilde{x}_{1,2} = \hat{x}_1,\ \tilde{x}_{2,1} = \tilde{x}_{2,2} = \hat{x}_2.$$

The aim of this paper is to propose a new scenario set partitioning method designed specifically for solving MSPs. The proposed partitioning method is a heuristic algorithm designed to build an optimal partition of Ω that minimizes the total number of relaxed NACs (RNACs). Our method takes into account the complex branching structure of general multistage scenario trees. In the context of MSPs, minimizing the total number of relaxed NACs (RNACs) enables to reduce the running time of the PHA for many different reasons.

- Firstly, it makes subproblems easier to solve by decreasing the number of linear and quadratic terms that need to be added in subproblems objective function to penalize violations of RNACs. This effect reduces the running time per iteration.

- Secondly, it reduces the total number of constraints and decision variables that need to be duplicated in all subproblems. In turn, this effect can also reduce the running time per iteration.

- Thirdly, minimizing the number of RNACs decreases the variability among subproblems solutions at duplicated tree nodes $n \in \mathcal{N}_c^*$ by enhancing the branching level in scenario subtrees. This effect accelerates the PHA's convergence rate by making RNACs easier to satisfy.

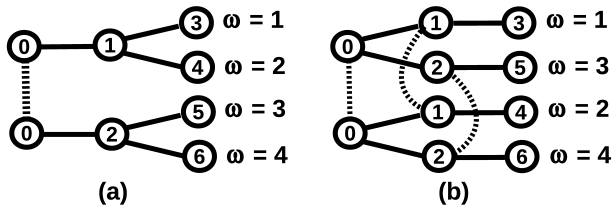

Figure 2: Examples of grouping schemes for a MST.

The paper is organized as follows. A general mathematical formulation for MSPs is presented in Section 2. Section 3 presents a non conventional version of the PHA which is based on a multi-scenario decomposition scheme. Section 4 describes the optimal scenario set partitioning method. Section 5 describes a numerical experiment. Numerical results are presented in Section 6. Comments and conclusions are drawn in Section 7.

2. Problem formulation

2.1. Multistage stochastic program

We consider the following optimization problem defined on T time periods

$$(\mathcal{P}) \qquad \min \mathbb{E}\left[\sum_{t=1}^{T} g_t(x_t, \xi_t)\right] \tag{4}$$

subject to

$$A_t(\xi_t)x_t + B_t(\xi_t)x_{t-1} = b_t(\xi_t) \quad , \quad \forall t = 1, ..., T \tag{5}$$

$$x_t \in \mathcal{X}_t \quad , \quad \forall t = 1, ..., T \tag{6}$$

$$x_t \in \mathcal{F}_t(\xi_1, ..., \xi_t) \quad , \quad \forall t = 1, ..., T \tag{7}$$

where $\mathbb{E}[\,\cdot\,]$ is the expectation operator, $g_t(\cdot, \cdot)$ are convex functions representing the operating costs at time period t, x_t is the decision vector at time period t and ξ_t is a random vector at time period t. Each component of ξ_t represents one of the problem's random parameter (e.g. interest rate, price, demand, inflows, ...) during time period t. Equations (5) and (6) represent dynamic (e.g. water budget, inventory dynamics, ramping constraints, ...) and static (e.g. system limits, mechanisms, ...) constraints, respectively.

Coefficient of technological matrices A_t, B_t and vectors b_t are treated as random variables. The sets of static constraints $\mathcal{X}_t$ are assumed to be non-empty and convex. Non-anticipativity constraints (7) ensure that each x_t is chosen using only known informations $\xi_1, ..., \xi_t$ and cannot depend on future (unknown) realizations of random parameters $\xi_{t+1}, ..., \xi_T$. Each set $\mathcal{F}_t$ contains all solutions that meet this criterion at time t.

2.2. Scenario tree

Each random vector ξ_t is characterized by a known probability distribution $\mathbb{P}_t(\ \cdot \mid \xi_{t-1}, ..., \xi_{t-p})$ which is conditional to previous observations $\xi_{t-1}, ..., \xi_{t-p}$ made over the $p \geq 1$ last periods. We assume that $\mathbb{P}_t$ possess a finite number of possible outcomes at each $t = 1, ..., T$. We also assume that all random parameters are exogenous to the controlled system. Therefore, $\mathbb{P}_t$ is not influenced by x_t. Making these assumptions enables us to represent the stochastic process $\{\xi_t : t = 1, ..., T\}$ using a finite scenario tree $\mathcal{T}$. Each node $n \in \mathcal{N}$ has an occurrence probability of p_n. Each scenario $\omega \in \Omega$ in $\mathcal{T}$ corresponds to a path from the root $0 \in \mathcal{N}$ to a particular leaf $\ell(\omega) \in \mathcal{L}$. The probability of a given scenario ω correspond to the probability $p_{\ell(\omega)}$ of its leaf $\ell(\omega)$. The branching structure of $\mathcal{T}$ is represented by a function $a(n)$ which returns the ancestor of any node n. The vector ξ_n represents the realization of random parameters at node n. The set $\Omega(n)$ contains all scenarios visiting node n. Each set $\Delta(n)$ contains all child nodes of n. Fig. 1a shows a simple example with $T = 3$, $\mathcal{N} = \{0, 1, 2, 3, 4, 5, 6\}$, $\mathcal{L} = \{3, 4, 5, 6\}$, $\Omega = \{1, 2, 3, 4\}$, $a(1) = a(2) = 0$.

2.3. Deterministic equivalent program

The problem $\mathcal{P}$ defined on a scenario tree $\mathcal{T}$ can be transformed into the following DEP

$$(\mathcal{E}) \qquad \min \sum_{n \in \mathcal{N}} p_n g_{t(n)}(\hat{x}_n, \xi_n) \tag{8}$$

subject to

$$A_n \hat{x}_n + B_n \hat{x}_{a(n)} = b_n \quad , \quad \forall n \in \mathcal{N}, \tag{9}$$

$$\hat{x}_n \in \mathcal{X}_n \quad , \quad \forall n \in \mathcal{N}. \tag{10}$$

The program $\mathcal{E}$ is obtained from $\mathcal{P}$ by replacing all occurrences of stage-wise decision vectors x_t by a node-wise decision vector $\hat{x}_n$ at node n. In

the objective function, the expectation operator is replaced by a finite sum. Each term is weighted by the probability of the corresponding tree node. Constraints 9 and 10 correspond to constraints 5 and 6, respectively. Non-anticipativity constraints 7 of $\mathcal{P}$ are represented implicitly in $\mathcal{E}$ because of the node-wise index system that we use. Indeed, any feasible solution to $\mathcal{E}$ is scenario-invariant at all tree nodes $n \in \mathcal{N}$. Therefore, x_n only depends on available informations at time period $t(n)$.

3. Solution method

3.1. Decomposition scheme

Instead of using the single-scenario decomposition scheme that is conventionally used with the PHA, we apply a multi-scenario decomposition scheme on $\mathcal{E}$. The scenario set Ω is partitioned into disjoint subsets (groups) Ω_c where $c \in \mathcal{C}$ is the group index. In this paper, we denote by $\mathcal{T}_c$ the subtree associated with all scenarios $\omega \in \Omega_c$ in group c. Each set $\mathcal{N}_c$ contains all the nodes that are visited by the scenarios $\omega \in \Omega_c$ in group c. The occurrence probability of group c is defined as follow

$$\tilde{p}_c := \sum_{\omega \in \Omega_c} p_{\ell(\omega)}. \tag{11}$$

The probability $\hat{p}_{nc}$ of node n conditional to group c is defined as follow

$$\hat{p}_{nc} := p_n / \tilde{p}_c. \tag{12}$$

The conventional single-scenario decomposition scheme is a particular case of the multi-scenario decomposition scheme where $|\Omega_c| = 1$ for all $c \in \mathcal{C}$.

3.2. Mathematical formulation

The program $\mathcal{E}$ defined on scenario groups $c \in \mathcal{C}$ is reformulated into the following equivalent program

$$(\tilde{\mathcal{E}}) \qquad \min \sum_{c \in \mathcal{C}} \tilde{p}_c \sum_{n \in \mathcal{N}_c} \hat{p}_{nc} g_{t(n)}(\tilde{x}_{nc}, \xi_n) \tag{13}$$

subject to

$$A_n \tilde{x}_{nc} + B_n \tilde{x}_{a(n)c} = b_n \quad , \quad \forall c \in \mathcal{C},\ n \in \mathcal{N}_c \tag{14}$$

$$\tilde{x}_{nc} \in \mathcal{X}_n \quad , \quad \forall c \in \mathcal{C},\ n \in \mathcal{N}_c, \tag{15}$$

$$\tilde{x}_{nc} = \hat{x}_n \quad , \quad \forall c \in \mathcal{C},\ n \in \mathcal{N}_c^* \ :\ \tilde{\lambda}_{nc}. \tag{16}$$

The formulation $\tilde{\mathcal{E}}$ can be obtained from $\mathcal{E}$ by making the following transformations. The objective function (13) and constraints (14)–(15) are obtained by replacing all occurrences of $\hat{x}_n$ at nodes $n \in \mathcal{N}$ by an alternative decision vector $\tilde{x}_{nc}$ defined at nodes $n \in \mathcal{N}_c$ visited by groups $c \in \mathcal{C}$. In (13), the probability p_n of node n is replaced by $\tilde{p}_c \hat{p}_{nc}$ according to (12). The NACs (16) ensure that any feasible solution of $\tilde{\mathcal{E}}$ is group-invariant at all tree nodes and $\tilde{\lambda}_{nc}$ is the vector of Lagrange multipliers associated with node n and group c. Each set $\mathcal{N}_c^*$ contains all nodes visited by group c and by at least another group. For the example shown on Fig. 2a, $\mathcal{N}_1^* = \mathcal{N}_2^* = \{0\}$.

3.3. Augmented Lagrangian

We apply an ALR on constraints (16) and the resulting objective function to be minimized is

$$\mathcal{A}_\rho(\tilde{x}, \hat{x}, \tilde{\lambda}) = \sum_{c \in \mathcal{C}} \tilde{p}_c \left(\sum_{n \in \mathcal{N}_c} \hat{p}_{nc} g_{t(n)}(\tilde{x}_{nc}, \xi_n) + ... \right.$$
$$\left. ... + \sum_{n \in \mathcal{N}_c^*} \left(\tilde{\lambda}'_{nc}(\tilde{x}_{nc} - \hat{x}_n) + \frac{\rho}{2} \|\tilde{x}_{nc} - \hat{x}_n\|^2 \right) \right)$$

subject to constraints (14)–(15). The penalty parameter ρ is a positive constant that needs to be tuned, $\tilde{x} = (\tilde{x}_{nc})$ is the vector of group-wise decision vectors, $\hat{x} = (\hat{x}_n)$ is the vector of node-wise decision vectors, $\tilde{\lambda} = (\tilde{\lambda}_{nc})$ is the vector of Lagrange multipliers associated with constraints (16). All vectors are assumed to be column vectors, $(\cdot)'$ is the transpose operator and $\| \cdot \|$ is the Euclidean norm.

3.4. Progressive hedging algorithm

The algorithm begins with an initial penalty parameter ρ_0, a suboptimal node-wise solution $\hat{x}^0 = (\hat{x}_n^0)$ and Lagrange multiplier $\tilde{\lambda}^0 = (\tilde{\lambda}_{nc}^0)$. Then, at each iteration $k = 0, 1, 2, ...$, the two following steps are performed.

Step 1. Find a new group-wise solution $\tilde{x}^{k+1} = (\tilde{x}_{nc}^{k+1})$ by minimizing $\mathcal{A}_{\rho_k}(\tilde{x}, \hat{x}^k, \tilde{\lambda}^k)$ for $\tilde{x}$ subject to constraints (14)–(15). Because we consider $(\tilde{x}^k, \tilde{\lambda}^k)$ as fixed values, this problem is separable by group. Each group-subproblem is a relatively small MSP defined on a subtree $\mathcal{T}_c$ associated with a particular group c. The subproblem associated with group c is

$$(\mathcal{S}_c^k) \qquad \min \tilde{p}_c \sum_{n \in \mathcal{N}_c} \hat{p}_{nc} g_{t(n)}(\tilde{x}_{nc}, \xi_n) + \sum_{n \in \mathcal{N}_c^*} \left((\tilde{\lambda}_{nc}^k)'(\tilde{x}_{nc} - \hat{x}_n^k) + \frac{\rho_k}{2} \|\tilde{x}_{nc} - \hat{x}_n^k\|^2 \right)$$

subject to

$$A_n\tilde{x}_{nc} + B_n\tilde{x}_{a(n)c} = b_n \quad , \quad \forall n \in \mathcal{N}_c, \tag{17}$$

$$\tilde{x}_{nc} \in \mathcal{X}_n \quad , \quad \forall n \in \mathcal{N}_c. \tag{18}$$

Step 2. a) Compute the new group-averaged solution

$$\hat{x}_n^{k+1} \leftarrow \sum_{c\in\mathcal{C}(n)} \tilde{p}_c\hat{p}_{nc}\tilde{x}_{nc}^k / \sum_{c\in\mathcal{C}(n)} \tilde{p}_c\hat{p}_{nc}, \qquad \forall n \in \mathcal{N}_*$$

where $\mathcal{C}(n)$ is the set of groups visiting node n.

b) Update the Lagrange multipliers

$$\tilde{\lambda}_{nc}^{k+1} \leftarrow \tilde{\lambda}_{nc}^k + \rho_k(\tilde{x}_{nc}^{k+1} - \hat{x}_n^{k+1}), \qquad \forall c \in \mathcal{C},\ n \in \mathcal{N}_c^*.$$

c) Update the penalty parameter using

$$\rho_{k+1} \leftarrow \mu\rho_k \tag{19}$$

where $\mu \geq 1$ is a constant that needs to be tuned. Equation 19 is the traditional penalty parameter update formula for general ALR methods (Nocedal and Wright , 2006).

d) Stop if

$$\zeta_k \leftarrow \frac{1}{T}\sum_{c\in\mathcal{C}} \tilde{p}_c \sum_{n\in\mathcal{N}_c^*} \|\tilde{x}_{nc}^{k+1} - \hat{x}_n^{k+1}\|^2 < \epsilon. \tag{20}$$

Otherwise, return to step 1. The stopping criterion ϵ is a positive constant. The metric ζ_k measures the violation level of RNACs at iteration k. In this problem, all decision variables are continuous and all constraints and the objective function are convex. Therefore, the PHA is an exact solution method for this problem. Rockafellar and Wets (1991) presented a proof of convergence for the PHA.

4. Scenario set partitioning

4.1. Optimal scenario set partitioning problem (OSPP)

The aim of the optimal scenario set partitioning problem (OSPP) is to build a partition of Ω that minimizes the total number of NACs on which an ALR must be applied. Each scenario group Ω_c must not contain more

than $N_{\max}$ scenarios to ensure that the size of all subproblems is manageable. The parameter $N_{\max}$ must be chosen to ensure that it does not exceed the computer's available memory. The OSPP is formulated as the following problem

$$\min \sum_{c \in \mathcal{C}} m \, |\mathcal{N}_c^*| \tag{21}$$

subject to

$$|\Omega_c| \leq N_{\max} \quad , \quad \forall c \in \mathcal{C} \tag{22}$$

$$\bigcup_{c \in \mathcal{C}} \Omega_c = \Omega \quad , \tag{23}$$

$$\Omega_c \cap \Omega_d = \emptyset \quad , \quad \forall c, d \in \mathcal{C}, c \neq d \tag{24}$$

The objective function (21) to be minimized is the total number of RNACs linking groups $c \in \mathcal{C}$ and m is a positive constant representing the number of decision variables in x_t. The constraints (22) ensures that all groups cannot contain more than $N_{\max}$ scenarios. The constraints (23) ensure that all scenarios are assigned to a particular group. The constraints (23) ensure that all subsets Ω_c are disjoint.

4.2. Heuristic partitioning method

The OSPP is a combinatorial optimization problem. In fact, the number of feasible partitions grows very rapidly with the number of scenarios in Ω. Even if the size of Ω is moderate (e.g. 100–1,000 scenarios), the OSPP cannot be solved by exhaustive enumeration of all possible partitions. To our knowledge, no exact or heuristic optimization methods has been proposed in the literature for solving this problem.

In this section, we propose a new heuristic method designed to solve the OSPP. The Algorithm 1 summarizes all steps of the scenario set partitioning heuristic (SSPH). This algorithm receives a general scenario tree $\mathcal{T}$ and the parameter $N_{\max}$ which represent the maximal number of scenarios that can be contained in a single group. The proposed method chooses how many groups are required and returns Ω_c, $\tilde{p}_c$ and $\hat{p}_{nc}$ for each $c \in \mathcal{C}$.

The Algorithm 1 builds a finite number of groups c sequentially for $c = 1, 2, ..., |\mathcal{C}|$ by selecting a different reference node $\tilde{n}$ among the set of candidate nodes $\tilde{\mathcal{N}}$. The algorithm starts by building group $c = 1$ and only the root node 0 is contained in $\tilde{\mathcal{N}}$. At each iteration the main `while` loop, the algorithm selects a new reference node $\tilde{n}$ and removes it from $\tilde{\mathcal{N}}$. The

Algorithm 1 Scenarios set partitioning heuristic (SSPH).

$c \leftarrow 1, \tilde{\mathcal{N}} \leftarrow \{0\}$
while $\tilde{\mathcal{N}} \neq \emptyset$ **do**
  $\tilde{n} \leftarrow \arg\max\{|\Omega(n)| : n \in \tilde{\mathcal{N}}\}$
  $\tilde{\mathcal{N}} \leftarrow \tilde{\mathcal{N}} - \{\tilde{n}\}$
  if $|\Omega(\tilde{n})| > N_{\max}$ **then**
    $\tilde{\mathcal{N}} \leftarrow \tilde{\mathcal{N}} \cup \Delta(\tilde{n})$
  else
    $\Omega_c \leftarrow \Omega(\tilde{n})$
    $\tilde{p}_c \leftarrow p_{\tilde{n}}$
    for $n \in \mathcal{D}(\tilde{n})$ **do**
      $\hat{p}_{nc} \leftarrow p_n / \tilde{p}_c$
    end for
    for $n \in \mathcal{K}(\tilde{n})$ **do**
      $\hat{p}_{nc} \leftarrow 1$
    end for
    $c \leftarrow c + 1$
  end if
end while

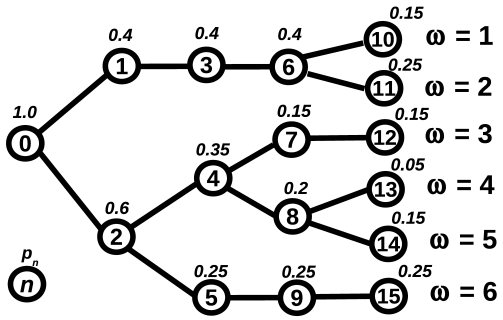

Figure 3: Example of a scenario tree.

node $\tilde{n}$ chosen is the one that is visited by the most scenarios in the tree. If the node $\tilde{n}$ is visited by more scenarios than is allowed in a single group $N_{\max}$, then this node is not the reference node of current group c to be constructed and all child nodes of $\tilde{n}$ are added to the set $\tilde{\mathcal{N}}$. Otherwise, the node $\tilde{n}$ is the reference node of group c, Ω_c is defined by all scenarios visiting node $\tilde{n}$, the probability of group c correspond to the occurrence probability of n and the conditional probability of each node $n \in \mathcal{N}_c$ visited by $\omega \in \Omega_c$ is computed as follow:

- The conditional probability of tree nodes $n \in \mathcal{D}(\tilde{n})$ that are located downstream of $\tilde{n}$ is computed using equation (12).
- The conditional probability of tree nodes $n \in \mathcal{K}(\tilde{n})$ that are located upstream of $\tilde{n}$ (including $\tilde{n}$ itself) is equal to one.

For the example shown on Fig. 3, $\mathcal{D}(4) = \{7, 8, 12, 13, 14\}$ and $\mathcal{K}(4) = \{0, 2, 4\}$. The algorithms continues as long as the set $\tilde{\mathcal{N}}$ is non-empty.

We illustrate how the Algorithm 1 works by applying it on the scenario tree shown on Fig. 3 with $N_{\max} = 3$. The algorithm returns three groups as shown on Fig. 4. The probability of each group is $\tilde{p}_1 = 0.35$, $\tilde{p}_2 = 0.4$ and $\tilde{p}_3 = 0.25$. The following NACs must be formulated explicitly

$$\tilde{x}_{0,1} = \tilde{x}_{0,2} = \tilde{x}_{0,3} = \hat{x}_0$$

$$\tilde{x}_{2,1} = \tilde{x}_{2,3} = \hat{x}_2.$$

The following intermediary results are obtained at five iterations of the `while` loop

1. $\tilde{\mathcal{N}} = \{0\}$ and, consequently, the $\tilde{n} = 0$ is selected and removed from $\tilde{\mathcal{N}}$. The selected node is visited by all scenarios ($\Omega(\tilde{n}) = \{1, ..., 6\}$). Because, $|\Omega(\tilde{n})| = 6 > 3$, the child nodes $\Delta(0) = \{1, 2\}$ are added to $\tilde{\mathcal{N}}$.
2. $\tilde{\mathcal{N}} = \{1, 2\}$. The node $\tilde{n} = 2$ is selected because $|\Omega(2)| = 4 > 2 = |\Omega(1)|$. Because $|\Omega(2)| = 4 > 3$, the child nodes $\Delta(2) = \{4, 5\}$ are added to $\tilde{\mathcal{N}}$.
3. $\tilde{\mathcal{N}} = \{1, 4, 5\}$. The node $\tilde{n} = 4$ is selected and removed from $\tilde{\mathcal{N}}$ because $|\Omega(4)| = 3 > 2 = |\Omega(1)| < 1 = |\Omega(5)|$. Because $|\Omega(4)| = 3 \leq 3$, the group $c = 1$ is defined by the scenarios $\Omega_1 = \{3, 4, 5\}$ and has a total probability $\tilde{p}_1 = 0.35$. The sets $\mathcal{K}_1 = \{0, 2, 4\}$ and $\mathcal{D}_1 = \{7, 8, 12, 13, 14\}$. The probability of each node in $c = 1$ is $\hat{p}_{0,1} = \hat{p}_{2,1} = \hat{p}_{4,1} = 1$, $\hat{p}_{7,1} = \hat{p}_{12,1} = \hat{p}_{14,1} = 0.15/0.35$, $\hat{p}_{8,1} = 0.2/0.35$ and $\hat{p}_{13,1} = 0.05/0.35$. We now start constructing the group $c = 2$.
4. $\tilde{\mathcal{N}} = \{1, 5\}$. The node $\tilde{n} = 1$ is selected and removed from $\tilde{\mathcal{N}}$ because $|\Omega(1)| = 2 > 1\,|\Omega(5)|$. Because $|\Omega(1)| = 2 \leq 3$, the group $c = 2$ is defined by the scenarios $\Omega_1 = \{1, 2\}$ and has a total probability $\tilde{p}_2 = 0.4$. The sets $\mathcal{K}_2 = \{0, 1, 3, 6\}$ and $\mathcal{D}_2 = \{10, 11\}$. The probability of each node in $c = 2$ is $\hat{p}_{0,2} = \hat{p}_{1,2} = \hat{p}_{3,2} = \hat{p}_{6,2} = 1$, $\hat{p}_{10,2} = 0.15/0.4$ and $\hat{p}_{10,2} = 0.25/0.4$. We now start constructing the group $c = 3$.
5. $\tilde{\mathcal{N}} = \{5\}$. The node $\tilde{n} = 5$ is selected and removed from $\tilde{\mathcal{N}}$. Because $|\Omega(5)| = 1 \leq 3$, the group $c = 3$ is defined by the scenarios $\Omega_1 = \{5, 6\}$ and has a total probability $\tilde{p}_3 = 0.25$. The sets $\mathcal{K}_3 = \{0, 2, 5\}$ and $\mathcal{D}_3 = \{15\}$. The probability of each node in $c = 3$ is $\hat{p}_{0,3} = \hat{p}_{2,3} = \hat{p}_{5,3} = \hat{p}_{9,3} = \hat{p}_{15,3} = 1$. $\tilde{\mathcal{N}} = \emptyset$. This is the last iteration because $\tilde{\mathcal{N}} = \emptyset$.

5. Numerical experiment

We apply the PHA described in subsection 3.4 on an hydroelectric reservoir management problem with stochastic inflows. This problem is formulated as a particular case of the general mathematical program $\mathcal{P}$. Hydrologic uncertainty is modeled by a finite scenario tree. Two different scenario set partitioning methods are compared. The first method we apply the SSPH described by Algorithm 1 using different values of $N_{\max}$. The second method builds a random partition of Ω where all groups contains N scenarios.

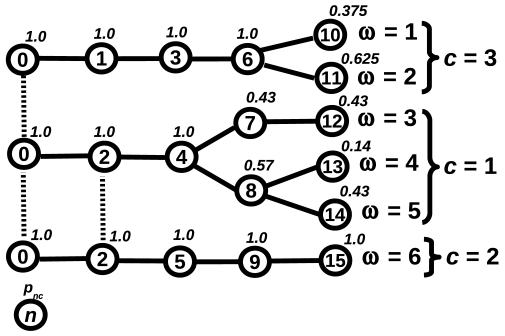

Figure 4: Subtrees.

5.1. Optimization problem statement

We consider an hydroelectricity producer that operates I hydro plants and J interconnected reservoirs over a T-period planning horizon. The objective function to be maximized is the expected value of

$$\sum_{t=1}^{T}\sum_{i=1}^{I} P_{it}\Delta t + \sum_{j=1}^{J} \alpha_j (v_{jT} - \underline{v}_j) \qquad \text{(MWh)} \tag{25}$$

where P_{it} (MW) is the power output of hydro plant i during time period t, Δt (hours) is the time step, v_{jT} (hm^3) is the volume of water stored in reservoir j at the end of time period T, $\underline{v}_j$ (hm^3) is the minimum storage of reservoir j and α_j (MWh/hm^3) is the production factor of reservoir j. The objective function (25) contains two parts. The first part represents the amount energy generated by all hydro plants $i = 1, ..., I$ during time periods $t = 1, ..., T$. The second part represents the amount of potential energy stored in reservoirs $j = 1, ..., J$ at the end time period $t = T$.

We assume that the power output P_{it} of hydro plant i during time period t is a concave and piecewise linear function of the turbined outflow q_{it} ($\text{m}^3\ \text{s}^{-1}$) at i during t and of the volume of water $v_{j(i)t}$ (hm^3) stored in the reservoir $j(i)$ located immediately upstream of i at the end of t. This assumption enables us model head and generation efficiency variations at hydro plants. Fig. 5 shows an illustrative example of a unidimensional production function defined by two pieces $h \in \{1, 2\}$. In the optimization model, the relationship between the P_{it}, q_{it} and $v_{j(i)t}$ is represented by the following linear inequality

constraints

$$P_{it} \leq \gamma_{ih}^{0} + \gamma_{ih}^{1} q_{it} + \gamma_{ih}^{2} v_{it}, \qquad \forall i, t, h. \tag{26}$$

where $\gamma_{ih}^{0}, \gamma_{ih}^{1}, \gamma_{ih}^{2}$ are the linear coefficients of piece h.

The volume of water stored in reservoirs $j = 1, ..., J$ at time periods $t = 1, ..., T$ evolves from a known initial state v_{j0} according to

$$v_{jt} = v_{j,t-1} + \left(\sum_{u \in \mathcal{U}(j)} Q_{ut} - Q_{jt} + \mathcal{I}_{jt} \right) \beta \Delta t, \qquad \forall j, t \tag{27}$$

where $\mathcal{U}(j)$ is a set that contains all reservoirs that are located immediately upstream of reservoir j, Q_{jt} ($\mathrm{m}^3\ \mathrm{s}^{-1}$) is the controlled outflow of reservoir j during t, $\beta = 0.0036$ is a constant used for converting flow units into volumetric units and $\mathcal{I}_{jt}$ ($\mathrm{m}^3\ \mathrm{s}^{-1}$) is a random parameter representing the intensity of natural inflows in reservoir j during t. The controlled outflow of reservoir j during t is defined as follow

$$Q_{jt} := s_{jt} + \sum_{i \in I(j)} q_{it} \qquad (\mathrm{hm})^3$$

where s_{jt} ($\mathrm{m}^3\ \mathrm{s}^{-1}$) is the spilled outflow of j during t and $I(j)$ is a set that contains all hydro plants connected directly to reservoir j.

All decision variables should also satisfy the following box constraints

$$\underline{v}_j \leq v_{jt} \leq \overline{v}_j, \qquad \forall j, t \tag{28}$$

$$\underline{s}_j \leq s_{jt} \leq \overline{s}_j, \qquad \forall j, t \tag{29}$$

$$\underline{P}_i \leq P_{it} \leq \overline{P}_i, \qquad \forall i, t \tag{30}$$

$$\underline{q}_i \leq q_{it} \leq \overline{q}_i, \qquad \forall i, t. \tag{31}$$

where $(\underline{v}_j, \underline{s}_j, \underline{P}_i, \underline{q}_i)$ and $(\overline{v}_j, \overline{s}_j, \overline{P}_i, \overline{q}_i)$ are parameters representing the lower and upper bounds on decision variables $(v_{jt}, s_{jt}, P_{it}, q_{it})$, respectively.

This stochastic optimization problem is a particular case of the general mathematical formulation $\mathcal{P}$ with random right-hand side vectors $b_t(\xi_t)$ at $t = 1, ..., T$. Each component of random vectors

$$\xi_t := (\mathcal{I}_{jt})$$

represents the intensity of natural inflows in a particular reservoir j at the corresponding time period t. The matrices A_t, B_t and cost functions g_t are

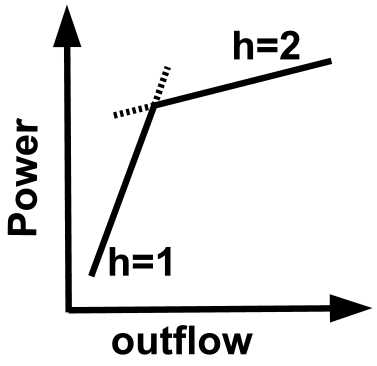

Figure 5: Hydroelectricity generation function.

deterministic and each decision vector is defined as follow

$$x_t := (P_{it}, q_{it}, v_{jt}, s_{jt}).$$

The cost functions at $t = 1, ..., T-1$ are defined as follow

$$g_t(x_t) := -\sum_{i=1}^{I} P_{it}\Delta t.$$

The cost function at $t = T$ is

$$g_T(x_T) := -\sum_{i=1}^{I} P_{iT}\Delta t - \sum_{j=1}^{J} \alpha_j(v_{jT} - \underline{v}_j).$$

The water balance equations (27) corresponds to dynamic constraints (5). The linear inequality (26) and box constraints (28)–(28) corresponds to static constraints (6).

5.2. Experimental set-up

We test our method on a power system containing $I = 4$ hydro plants and $J = 4$ reservoirs. The power system considered in this experiment has an installed capacity of 1,572 MW and is inpired by a similar reservoir system located in Québec, Canada. The reservoir system has a total storage capacity of 3,710 hm^3. Fig. 6 shows the structure of the hydrosystem. The

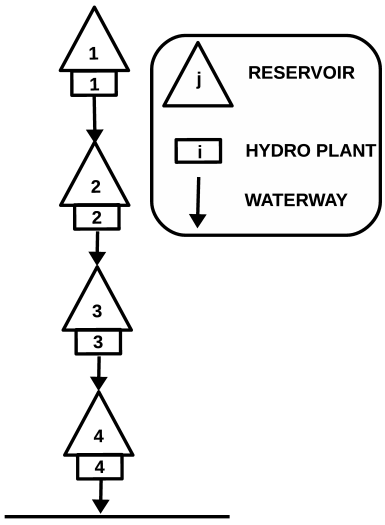

Figure 6: Reservoir system.

characteristics of each hydro plant and reservoir are summarized on Tables 1 and 2, respectively. Three hyperplanes ($h = 1, 2, 3$) are used to describe each concave and piecewise linear hydroelectric generation functions. The planning horizon is discretized in $T = 52$ time periods and a time step of $\Delta t = 168$ hours is used. The stopping condition used in this experiment is $\epsilon = 10^{-3}$.

We implemented the PHA in object-oriented C++ using the 12.5.1 version of the ILOG CPLEX and Concert libraries. Subproblems were solved using the barrier solver in parallel mode. All the numerical results presented in this paper were obtained using a personal computer running on Ubuntu 12.04 with a AMD Phenom II X6 2.8 GHz processor and 6 GB of RAM.

5.3. Scenario tree

In this experiment, we represent the hydrological stochastic process $\{\xi_t : t = 1, ..., T\}$ using the scenario tree shown on Fig. 7 which contains 500 scenarios and 16,275 nodes. The tree was constructed with the backward algorithm of the SCENRED2 package using 1,000 and 10,000 synthetically generated time series. This software is part of the General Algebraic Modeling System (GAMS) version 23.9.3. The time series were generated using

Table 1: Characteristics hydro plants

Plant	Maximum power output (MW)	Maximum turbined outflow ($m^3\ s^{-1}$)
1	259	315
2	404	375
3	647	467
4	262	485

Table 2: Characteristics reservoirs

Reservoir	Minimum storage (hm^3)	Maximum storage (hm^3)	Initial storage (hm^3)	Production factor (MWh/hm^3)
1	952	2,710	1,831	1,091
2	1,403	1,878	1,840	872
3	2,260	3,720	3,645	562
4	129	147	144	158

a MPAR(1) stochastic model which was tuned using historical data covering the period 1962–2003.

6. Results

6.1. Partitioning schemes

Table 3 shows the different partitioning schemes that were obtained using the SSPH with different values of $N_{\max}$. The scheme S1 was obtained using $N_{\max} = 1$ scenario per group. Therefore it corresponds to the classical scenario-decomposition scheme. We observe that the total number of RNACs decreases substantially as $N_{\max}$ increases. Increasing $N_{\max}$ from 10 to 100 decreases the number of RNACs by 95.4 %.

Table 4 shows different random partitioning schemes which were obtained using different values of N. The schemes R1a, R1b and R1c are three different replications obtained by partitioning the scenario tree into 50 groups of $N = 10$ scenarios. The average number of RNACs is 17.3 times larger in random schemes than in S2. The schemes R2a, R2b and R2c are three different replications obtained by partitioning the scenario tree into 5 groups of $N =$

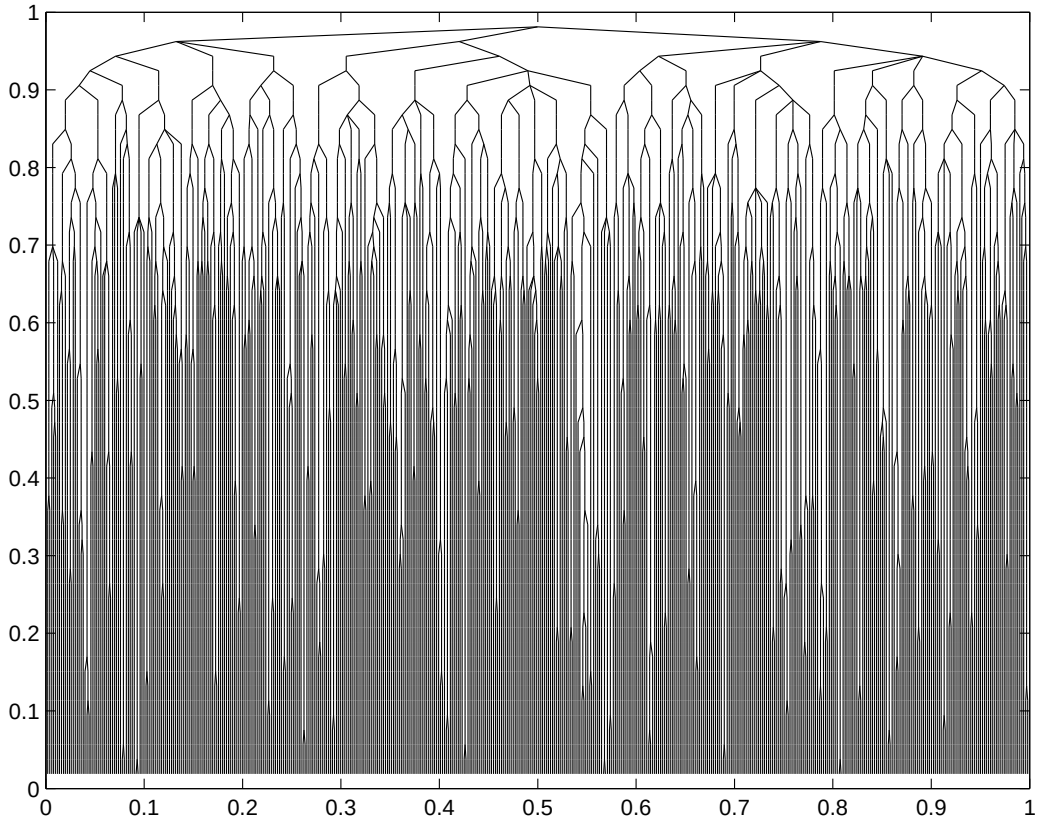

Figure 7: Scenario tree.

Table 3: Optimized partitioning schemes.

Scheme	$N_{\max}$	Number of groups	Number of RNACs
S1	1	500	52,696
S2	10	84	2,664
S3	100	12	120

100 scenarios. The average number of RNACs is 256 times larger in random schemes than in S3. We observe that increasing the size of scenario groups from $N = 10$ to $N = 100$ decreases average number of RNACs by 33.2 %.

Table 4: Random partitioning schemes.

Scheme	N	Number of groups	Number of RNACs
R1a	10	50	45,688
R1b	10	50	46,100
R1c	10	50	46,184
R2a	100	5	30,880
R2b	100	5	30,920
R2c	100	5	30,388

6.2. Evaluation of partitioning schemes

Table 5 shows the results obtained using the PHA with the optimized partitioning schemes that are shown on Table 3. These results were obtained using $\rho_0 = 10^{-4}$ and $\mu = 1.2$. We observe that the total running time of the PHA decreases substantially as $N_{\max}$ increases. This improvement is explained by a reduction in the running time per iteration and by a reduction in the number of iterations. The total running time is 7.5 times faster when $N_{\max}$ is increased from 1 to 10 scenarios. The total running time is 54 times faster when $N_{\max}$ is increased from 10 to 100 scenarios.

Table 5: Results obtained using optimized partitioning schemes.

Scheme	Iterations	Objective (TWh)	Total time (minutes)	Time per iteration (seconds)
S1	33	10.4782577	122.6	222.9
S2	21	10.4823670	16.3	46.6
S3	1	10.4823916	0.3	17.2

Table 6 shows the results obtained using the PHA with the random partitioning schemes shown on Table 4 with $\rho_0 = 10^{-4}$ and $\mu = 1.2$. On average, increasing N from 10 to 100 doubled the total running time of the algorithm. This is mainly explained by an increase in the running time per iteration. Also, the number of iterations increased slightly when N increased from 10 to 100. Overall, using a random partitioning method lead to much higher running time than when the SSPH was used. When $N = N_{\max} = 10$, the running time is 44 % higher when a random partition is used in comparison

with the SSPH. When $N = N_{\max} = 100$, the running time is 161 times higher when a random partition is used in comparison with the SSPH.

Table 6: Results obtained with random partitioning schemes.

Scheme	Iterations	Objective (TWh)	Total time (minutes)	Time per iteration (seconds)
R1a	34	10.4814887	23.3	41.1
R1b	34	10.4814797	23.3	41.1
R1c	35	10.4817173	23.9	41.0
R2a	36	10.4821520	46.6	77.7
R2b	38	10.4821276	49.0	77.4
R2c	38	10.4821188	49.2	77.6

6.3. Sensitivity to ρ_0 and μ

Table 7 shows the numerical results obtained using different values for parameters $\rho_0 \in \{10^{-4}, 10^{-3}\}$ and $\mu \in \{1.1, 1.2, 1.3\}$ using the partitioning schemes S1, S2 and S3. These parameters are used in equation 19 to update the penalty parameter ρ_k at each iteration k. We observe that number of iterations required to satisfy all RNACs and the running time decrease as ρ_0 and μ increases. This is mainly due to the fact that the penalty parameter weighs the violations of RNACs in subproblems objective function. We also observe that the PHA becomes less sensitive to the choice of μ and ρ_0 when $N_{\max}$ increases. The traditional scenario-decomposition scheme S1 ($N_{\max} = 1$) required between 17 and 51 iterations depending on which values were used for μ and ρ_0. In comparison, the scheme S2 required 10–33 iterations for the same range of parameter values. The scheme S3 required only one iteration, no matter which ρ_0 and μ were used.

7. Conclusions

In this article, we considered an enhanced version of the PHA for solving MSPs that are based on a ST representation of random parameters. The enhanced PHA is based on a multi-scenario decomposition scheme where each subproblem is a MSP defined on a scenario group. We proposed a new heuristic method to partition the scenario set into disjoint subsets (groups) to minimize the number of RNACs. We demonstrated using a numerical experiment that using this partitioning method reduces substantially running

Table 7: Sensitivity to ρ_0 and μ.

Scheme	ρ_0	μ	Iterations	Objective (TWh)	Total time (minutes)	Time per iteration (seconds)
S1	10^{-4}	1.10	51	10.4786199	187.9	221.0
S1	10^{-4}	1.20	33	10.4782577	122.6	222.9
S1	10^{-4}	1.30	26	10.4779222	97.0	223.8
S1	10^{-3}	1.10	28	10.4780311	104.1	223.1
S1	10^{-3}	1.20	20	10.4772046	74.3	222.9
S1	10^{-3}	1.30	17	10.4765857	62.9	222.1
S2	10^{-4}	1.10	33	10.4823691	24.9	45.2
S2	10^{-4}	1.20	21	10.4823670	16.3	46.6
S2	10^{-4}	1.30	17	10.4823587	12.8	45.0
S2	10^{-3}	1.10	13	10.4823524	9.8	45.1
S2	10^{-3}	1.20	11	10.4823410	8.2	44.7
S2	10^{-3}	1.30	10	10.4823319	7.4	44.6
S3	10^{-4}	1.10	1	10.4823916	0.3	17.3
S3	10^{-4}	1.20	1	10.4823916	0.3	17.2
S3	10^{-4}	1.30	1	10.4823916	0.3	17.2
S3	10^{-3}	1.10	1	10.4823916	0.3	17.6
S3	10^{-3}	1.20	1	10.4823916	0.3	17.5
S3	10^{-3}	1.30	1	10.4823916	0.3	17.1

time of the PHA. The proposed approach was applied on an hydroelectric reservoir management problem with uncertain natural inflows. We compared the performance of the PHA using different partitioning schemes which were built using our approach with a random partitioning method. We also tested the sensitivity of the PHA to the penalty parameter choice for different decomposition schemes. Our numerical results demonstrate that minimizing the number of RNACs leads to much lower running times compared with a random partitioning scheme. Increasing the size of subproblems reduces the number of iterations and reduces the running time per iteration. The proposed approach also decreases the sensitivity of the PHA to the penalty parameter choice.

References

Archibald, T.W., Buchanan, C.S., McKinnon, K.I.M, & Thomas, L.C. (1999). Nested Benders decomposition and dynamic programming for reservoir optimization. *Journal of the Operational Research Society*, 50, 468–479.

Benders, J.F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4, 238–252.

Birge, J.R. (1985). Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs. *Operations Research*, 33, 989–1007.

Birge, J.R., & Louveaux, F.V. (1988). A multicut algorithm for two-stage stochastic linear programs.*European Journal on Operational Research*, 34, 384–392.

Birge, J.R., & Louveaux, F.V. (2011). *Introduction to Stochastic Programming*. (2nd edition). New York: Springer.

Carpentier, P.-L., Gendreau, M., & Bastin, F. (2013a). Long-term management of an hydroelectric multireservoir system under uncertainty using the progressive hedging algorithm. *Water Resources Research*, 49, 2812–2827.

Carpentier, P.-L., Gendreau, M., & Bastin, F. (2013b). The Extended L-Shaped Method for Mid-Term Planning of Hydroelectricity Generation *IEEE Transactions on Power Systems*, (Submitted).

Crainic, T.G., Fu, X., Gendreau, M., Rei, W., Wallace, S.W. (2011). Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58, 114–124.

Crainic, T.G., Hewitt, M., Rei, W. (2012). Scenario Clustering in a Progressive Hedging-Based Meta-Heuristic for Stochastic Network Design. *Publication CIRRELT-2013-52*, 1–25.

Dantzig, G.B. (1955). Linear programming under uncertainty. *Management Science*, 3, 197–206.

Dupačová, J. (2002). Applications of stochastic programming: Achievements and questions. *European Journal of Operational Research*, 140, 281–290.

Gassman, H.I., & Ziemba, W.T. (2013). *Stochastic Programming: Applications in Finance, Energy, Planning and Energy*. World Scientific.

Gonçalves, R.E.C., Finardi, E.C., Da Silva, E.L. (2011). Applying different decomposition schemes using the progressive hedging algorithm to the operation planning problem of a hydrothermal system. *Electric Power Systems Research*, 83, 19–27.

Haugen, K.K., Lokketangen, A., & Woodruff, D.L. (2001). Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operational Research*, 132, 116–122.

Heitsch, H., Römisch, W., & Strugarek, C. (2006). Stability of multistage stochastic programs. *SIAM Journal on Optimization*, 17, 511–525.

Heitsch, H., & Römisch, W. (2009). Scenario tree modeling for multistage stochastic programs. *Mathematical Programming Series A*, 118, 371–406.

Høyland, K., & Wallace, S.W. (2001). Generating Scenario Trees for Multistage Decision Problems. *Management Science*, 47, 295–307.

Kallrath, J., Pardalos, P., Rebennack, S., Scheidt (2009). *Optimization in the Energy Industry*. Berlin: Springer.

King, A.J., & Wallace, S.W. (2012). *Modeling with Stochastic Programming*. New York: Springer.

Küchler, C., & Vigerske, S. (2007). Decomposition of Multistage Stochastic Programs with Recombining Scenario Trees. *Stochastic Programming E-Print Series*.

Laporte, G., & Louveaux, F.V. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13, 133–142.

Latorre, J.M., Cerisola, S., & Ramos, A. (2007). Clustering algorithms for scenario tree generation: Application to natural hydro inflows. *European Journal of Operational Research*, 181, 1339–1353.

Mulvey, J.M., Rosenbaum, D., & Shetty, B. (1997). Strategic financial risk management and operations research. *European Journal of Operational Research*, 97, 1–16.

Mulvey, J.M., & Vladimirou, H. (1991). Applying the progressive hedging algorithm to stochastic generalized networks. *Annals of Operations Research*, 31, 399–424.

Nocedal, J., & Wright, S. (2006). *Numerical Optimization*. (2nd edition). Springer Series in Operations Research and Financial Engineering. New York: Springer.

Pereira, M.V.F, & Pinto, L.M.V.G. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52, 359–375.

Pflug, G.C. (2001). Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming Series B*, 89, 251–271.

Powell, W.B., & Topaloglu, H. (2003). Stochastic Programming in Transportation and Logistics. In Rusczynzki, A., & Shapiro, A. (Eds.), *Stochastic Programming* pp. 555–635. New York: Elsevier.

Wallace, S.W., & Ziemba, W.T. (2005). *Applications of Stochastic Programming*. SIAM.

Wallace, S.W., & Fleten, S.-E. (2003). Stochastic Programming Models in Energy. In Rusczynzki, A., & Shapiro, A. (Eds.), *Stochastic Programming* pp. 637-677. New York: Elsevier.

Rockafellar, R.T., & Wets, R.J.B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16, 119–147.

Rusczynzki, A. (2003). Decomposition methods. In Rusczynzki, A., & Shapiro, A. (Eds.), *Stochastic Programming* pp. 141–211. New York: Elsevier.

Rusczynzki, A., & Shapiro, A. (Eds.), *Stochastic Programming*. New York: Elsevier.

Sagastizábal, C. (2012). Divide to conquer: decomposition methods for energy optimization. *Mathematical Programming Series B*, 134, 187–222.

Van Slyke, R.M., & Wets, R.B. (1969). L-shaped linear programs with applications to optimal control and stochastic programming.*SIAM Journal on Applied Mathematics*, 17, 638–663.

Watson, J.P. & Woodruff, D.L. (2011). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems.*Computational Management Science*, 8, 355–370.

Wolf, C., & Kobertstein, A. (2013). Dynamic sequencing and cut consolidation for the parallel hybrid-cut nested L-shaped methods.*European Journal on Operational Research*, 230, 143–156.