Energy Syst (2011) 2: 1–31
DOI 10.1007/s12667-011-0024-y

ORIGINAL PAPER

Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling

Tito Homem-de-Mello · Vitor L. de Matos · Erlon C. Finardi

Received: 6 July 2010 / Accepted: 8 January 2011 / Published online: 22 January 2011

Abstract The long-term hydrothermal scheduling is one of the most important problems to be solved in the power systems area. This problem aims to obtain an optimal policy, under water (energy) resources uncertainty, for hydro and thermal plants over a multi-annual planning horizon. It is natural to model the problem as a multi-stage stochastic program, a class of models for which algorithms have been developed. The original stochastic process is represented by a finite scenario tree and, because of the large number of stages, a sampling-based method such as the Stochastic Dual Dynamic Programming (SDDP) algorithm is required. The purpose of this paper is two-fold. Firstly, we study the application of two alternative sampling strategies to the standard Monte Carlo—namely, Latin hypercube sampling and randomized quasi-Monte Carlo—for the generation of scenario trees, as well as for the sampling of scenarios that is part of the SDDP algorithm. Secondly, we discuss the formulation of stopping criteria for the optimization algorithm in terms of statistical hypothesis tests, which allows us to propose an alternative criterion that is more robust than that originally proposed for the SDDP. We test these ideas on a problem associated with the whole Brazilian power system, with a three-year planning horizon.

T. Homem-de-Mello (✉)
Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL, USA
e-mail: thmello@uic.edu

V.L. de Matos · E.C. Finardi
LabPlan—Laboratório de Planejamento de Sistemas de Energia Elétrica, Universidade Federal de Santa Catarina, Florianopolis, Brazil

V.L. de Matos
e-mail: vitor@labplan.ufsc.br

E.C. Finardi
e-mail: erlon@labplan.ufsc.br

1 Introduction

In hydrothermal power systems, energy planning plays an important role for the minimization of operation costs, which consist of thermal fuel costs and energy deficit penalties. This importance is accentuated by the fact that the operation of the reservoirs depends on the operation in previous periods, which requires planning far in advance. The hydrothermal scheduling problem is a very complex one, given that all the particularities related to this problem cannot be accommodated in a single optimization model. In order to deal with this task, a chain of models is normally used, dividing the main problem into a hierarchy of smaller subproblems with different planning horizons and degrees of detail.

In the Long-Term Hydrothermal Scheduling Problem (LTHSP), which is the focus of discussion in this paper, the main objective is to obtain an optimal policy that minimize the expected costs of operating power plants over a long planning horizon (divided into periods), taking into account constraints related to the system and power plants. The decisions, which are made at the end of each planning period, include the volumes of the reservoirs—or, equivalently, the amount of potential hydroelectric energy—and the amount of thermal generation required to satisfy demand. The model needs to take into consideration the water inflow seasonality and uncertainty, and the possibility of transferring energy among the reservoirs over the planning horizon. The main outcome of the LTHSP is the future cost function, which represents the system operation costs in the future as a function of the stored water and the water inflow. In addition, the future cost function is used as an input (in the form of a boundary condition) to the Medium-Term Hydrothermal Scheduling Problem [8], which is the subsequent optimization model in the aforementioned chain of models.

The LTHSP is a large scale linear stochastic optimization problem, with temporal and spatial coupling. The temporal coupling is explained by the fact that the water can be managed over time, observing the storage capacity of each reservoir. The spatial coupling is due to the fact that the dispatch of a particular hydro plant alters the inflow to the downstream plants. The stochastic characteristic arises from the impossibility to forecast future inflows accurately. Inflows are modeled as random variables with known probability distribution. In a long planning horizon (e.g., 5 to 10 years) it is more important to represent the inflows uncertainties than the nonlinearities associated with hydro production and thermal costs (see [10]). Therefore, the problem is represented by a linear model, with a more detailed representation of the stochastic model. Moreover, decisions must be revised periodically over time, to account for changes in the inflows forecast. Thus, the problem can be formulated as a multi-stage stochastic program (MSSP).

Due to the aforementioned characteristics, the LTHSP is a large scale problem, so specialized stochastic programming techniques are required to find a solution with good quality. There are several strategies to solve an MSSP, all of which assume the existence of a scenario tree that approximate the distribution of the inflows (which are typically continuous). One way to build a scenario tree is by sampling a group of possible realizations from the distribution. A traditional sampling technique is the so-called Monte Carlo method (MC), in which the samples are independent and identically distributed (iid). Nevertheless, there are other techniques that can produce a more representative scenario tree, such as Quasi-Monte Carlo (QMC) and Latin hypercube sampling (LHS). These techniques, which are popular in other domains such as statistics and financial engineering, are reviewed later in the paper.

Once a scenario tree is built, it is necessary to select a solution strategy to solve it. In the LTHSP, the large number of stages makes it computationally infeasible to solve the problem associated with the whole scenario tree. Therefore, it is necessary to use an optimization algorithm that incorporates sampling of scenarios from the scenario tree. One such algorithm is the Stochastic Dual Dynamic Programming (SDDP) proposed in [31]; some variations of it have also been studied in the literature. Since sampling of scenarios is involved, one can in principle use any of the aforementioned techniques (MC, QMC and LHS) for that purpose as well.

One important issue that arises when using optimization algorithms that involve sampling is the definition of appropriate stopping criteria for the algorithm. The situation is more complicated than that of algorithms that deal exclusively with deterministic functions, since the stopping criteria are based on statistical estimates and as such must take into account the possibility of stopping either too soon or too late. While some methods have been proposed in the literature on sampling-based stochastic optimization, they do not fully quantify the probability of making an incorrect decision.

The goal of this paper is two-fold: on one hand, we want to compare the use of the three sampling techniques (MC, QMC and LHS) in both aspects discussed above—i.e., to build the scenario tree and to sample the scenarios for the optimization algorithm. We are also interested in studying if there is any combined effect between the technique used for the scenario tree and the technique used for sampling scenarios. On the other hand, we discuss the formulation of stopping criteria for the optimization algorithm in terms of statistical hypothesis tests. Such a formulation allows us not only to evaluate the quality of the criterion used by the SDDP algorithm—which, as we show, has some flaws—but also to propose an alternative criterion that guarantees a more robust solution, as we can impose bounds on the probabilities of stopping too soon or too late. In addition, the hypothesis tests yield a performance measure that we can use to compare the different sampling techniques.

These ideas are tested on a LTHSP that arises in the Brazilian interconnected power system, which contains almost 180 hydro and 150 thermal plants. To reduce the problem to a manageable size, the hydroelectric power plants are merged into Energy Equivalent Reservoirs (EER) as in [6], where the EER's attributes describe the hydro plants characteristics in terms of energy instead of water. We consider a three-year planning horizon, with decisions being revised on a monthly basis (so, the problem has 36 stages). The model is presented in the Appendix. The results suggest that the alternative sampling techniques can be very effective. Moreover, the new stopping criterion appears to fix the problems observed with the stopping criterion originally proposed with SDDP, which often leads to stopping the algorithm prematurely.

This paper is organized as follows. We first introduce some important concepts of MSSP problems and the SDDP solution strategy in Sect. 2. In Sect. 3 we present hypothesis test formulations of the stopping criteria, showing why the criterion in SDDP is inadequate and proposing an alternative criterion. Section 4 presents a brief overview of the LHS and QMC sampling techniques, which are alternatives to standard Monte Carlo. The numerical experiments for the Brazilian system are shown and discussed in Sect. 5. Finally, Sect. 6 presents some conclusions and final remarks.

2 Multi-stage stochastic programming models

2.1 Formulation

Multi-stage stochastic programs have been widely used in multiple areas such as transportation, revenue management and finance in addition to energy planning. A general MSSP for a problem with $T+1$ stages can be written as

$$\begin{aligned} \text{[MSSP]} \quad \min \ & f_0' x_0 + \mathbb{E}_{\xi_1}\left[Q_1(x_0,\xi_1)\right] \\ \text{subject to} \quad & A_0 x_0 = c_0, \\ & x_0 \geq 0 \end{aligned}$$

The function Q_1 is defined recursively as

$$\begin{aligned} Q_t(x_0,\ldots,x_{t-1},\xi_1,\ldots,\xi_t) = \min \ & f_t' x_t + \mathbb{E}_{\xi_{t+1}}\left[Q_{t+1}(x_0,\ldots,x_t,\xi_1,\ldots,\xi_{t+1})\right] \\ \text{subject to} \quad & A_t x_t = c_t - \sum_{m=0}^{t-1} B_m x_m, \\ & x_t \geq 0 \end{aligned} \tag{1}$$

$t = 1,\ldots,T-1$. In the above formulation, the random element ξ_t denotes the random components of f_t, A_t, B_t, c_t. Notice that we use the notation $\mathbb{E}_{\xi_{t+1}}[Q_{t+1}(x_0,\ldots, x_t,\xi_1,\ldots,\xi_{t+1})]$ to indicate the conditional expectation $\mathbb{E}[Q_{t+1}(x_0,\ldots,x_t,\xi_1, \ldots,\xi_{t+1}) \,|\, \xi_1,\ldots,\xi_t]$. The function Q_T for the final stage T is defined the same way as the general Q_t in (1), except that it does not contain the expectation term in the objective function.

Multi-stage stochastic programs are notoriously hard to solve. One of the main difficulties lies in the fact that the optimal solution at stage t depends on the history of the process up to that stage. Thus, different realizations of the process will lead to different solutions, which means that the expectations in the MSSP cannot be calculated exactly unless the total number of scenarios is relatively small—which is certainly not the case in the long-term hydrothermal scheduling problems. Thus, it is necessary to resort to *sampling* techniques.

Sampling methods for stochastic optimization problems of the form

$$\min_{x \in X} \mathbb{E}_\zeta \left[G(x, \zeta) \right] \tag{2}$$

—which includes nonlinear and combinatorial problems—have received a great deal of attention in recent years due to their ability to approximate well, often with a small number of samples, problems that have very large number of scenarios; see, for instance [20] for numerical reports. The approximation is defined by replacing problem (2) with

$$\min_{x \in X} \frac{1}{N} \sum_{i=1}^{N} G(x, \zeta^i),$$

where $\zeta^1, \ldots, \zeta^N$ are samples from the distribution of ζ. Such an approach is usually called sample average approximation (SAA) in the literature. The reasons for the good performance of the SAA approach are well understood and have to do with how well-conditioned the problem is [38]. The SAA approach has solid foundation in terms of convergence properties as a function of the sample size N, see for instance [34] for a compilation of results. Much work has also been done to determine statistical error bounds for the SAA problem for fixed sample size; examples of such work are [2, 22] and in particular [4] for the case of multi-stage problems.

Although the MSSP model fits the framework of (2), the application of sampling methods to that class of problems is more delicate. As discussed in [35], a procedure whereby some samples of the vector $\xi := (\xi_1, \ldots, \xi_T)$ are generated and the corresponding approximating problems are solved will not work well. It is not difficult to see why—the nested structure of MSSP requires that the expectation *at each stage* be approximated by a sample average, which cannot be guaranteed by simply drawing samples of ξ. To circumvent the problem, a conditional sampling scheme, in which samples of ξ_t are generated for each sampled value of ξ_{t-1}, must be used. Of course, this implies that the number of samples required to obtain a good approximation of MSSP grows exponentially with the number of stages.

The idea of generating samples using a conditional approach constitutes one way of forming a *scenario tree*. Other approaches to construct scenario trees have been discussed in the literature (see, for instance [12] for a recent review) but here we focus exclusively on sampling method since we can rely on the statistical inference provided in [35]. Moreover, one of the goals of this paper is to study the effect of alternative sampling techniques on the overall convergence of an algorithm for MSSP.

Once a scenario tree is generated (by whichever method), an MSSP is typically solved by using some decomposition technique—see, for instance [33] for a review. Many of the existing solution methods, though, assume that the tree is relatively small so that scenarios can actually be enumerated. This is not the case in the LTHSP, where the goal of planning for a long-term horizon requires the

MSSP to have many stages (for example, 120 stages, corresponding to monthly decisions over a ten-year horizon). Even if a crude scenario tree is built to approximate the original problem—say, with 2 realizations per stage—the total number of scenarios 2^T will be very large, thus making enumeration of scenarios impractical.

To address the problem of large scenario trees, some algorithms have been proposed whereby sampling of scenarios *from the tree* is incorporated into an optimization procedure.[1] Note that the input to these algorithms is a scenario tree, so the sampling that is conducted within the algorithm is independent of any sampling that may have been performed in order to generate a scenario tree. Examples of such algorithms are SDDP [31], CUPPS [3], Abridged Nested Decomposition [7], and ReSa [14]. A convergence analysis of this class of algorithms is provided in [32]. In this paper we focus on the SDDP (Stochastic Dual Dynamic Programming) algorithm, which we discuss next.

2.2 The stochastic dual dynamic programming method

We provide now a brief description of the SDDP algorithm (more comprehensive discussions can be found in [31] and [32]). The SDDP is a sampling-based algorithm for solving multi-stage stochastic linear problems. The algorithm can be divided into two steps: forward and backward passes. In the forward pass, the algorithm samples a group of scenarios—full path from the first to the last stage—from the scenario tree; then, it optimizes each node of these scenarios starting at the first stage and proceeding to the next one until it solves the last stage. In the backward pass, the algorithm starts at the last stage and adds information to the previous stage with respect to the expected operation future cost until the first stage is reached. That information comes in the form of a Benders cut, which is a linear approximation of the future cost function. Note that, because of the convexity of the cost function, the piecewise linear function defined by successive Benders cuts yields a lower approximation to the future cost function (i.e. the graph of the approximating function is "underneath" the graph of the true function). In the context of the LTHSP, the cut approximately measures how much the operation future cost depends on the amount of water that will be available in the beginning of the next time stage.

As shown in [32], one way to guarantee asymptotic convergence of the SDDP algorithm is to ensure that every scenario has a positive probability of being generated during the course of the algorithm. To accomplish that, new scenarios from the scenario tree are sampled at each iteration. If convergence has not been reached yet, new information is added about the future cost in the backward pass, which is followed by a new forward pass. The algorithm continues to perform the forward and backward passes until the optimal value z_L of the lower approximation to the expected operation cost (obtained from the Benders cuts) is equal to the expected operation cost z_U of the current policy, which indicates that convergence has been achieved.

[1]This is sometimes called internal sampling in the literature, as opposed to external sampling, as in the SAA approach, where the sample is fixed outside the optimization algorithm.

However, the huge size of the scenario tree precludes the exact calculation of the expected total operation cost. This value can be estimated by using the scenarios that were sampled in the forward pass. Let $\bar{z}_N$ denote this estimator. Then, convergence is accepted when $\bar{z}_N$ is statistically indifferent from z_L. In order to check that property, the SDDP algorithm proposed in [31] tests whether z_L lies in the 95% confidence interval corresponding to the estimator $\bar{z}_N$. This convergence criterion is discussed in detail in Sect. 3. In summary, the SDDP algorithm can be described as follows:

1. Initialize the iteration index, $k = 1$;
2. Sample N scenarios from the scenario tree;
3. Solve the forward pass:
   (a) For each $i = 1, \ldots, N$ and $t = 1, \ldots, T$, solve the LP problem associated with that node and save z_{ti}, which is the cost of the node in stage t and scenario i;
   (b) For each $i = 1, \ldots, N$, compute the cost of scenario i as:

$$z^i = \sum_{t=1}^{T} z_{ti} \tag{3}$$

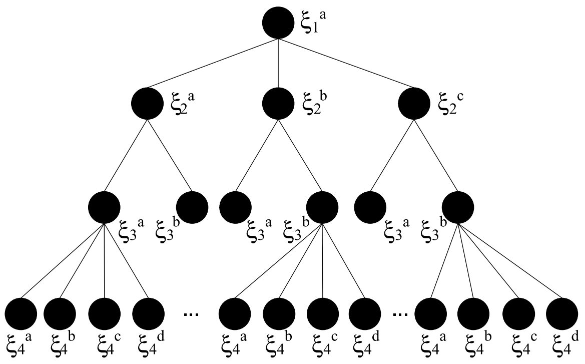
4. Check the convergence criterion:
   (a) Calculate the lower bound z_L, which is the optimal value of the lower approximation to the expected operation cost obtained from the Benders cuts.
   (b) Calculate the estimate for the upper bound and the standard deviation:

$$\bar{z}_N = \frac{1}{N}\sum_{i=1}^{N} z^i \tag{4}$$

$$s_N^2 := \frac{1}{N-1}\sum_{i=1}^{N}(z^i - \bar{z}_N)^2 \tag{5}$$

   (c) If z_L lies in the confidence interval $[\bar{z}_N - 1.96\frac{s_N}{\sqrt{N}}; \bar{z}_N + 1.96\frac{s_N}{\sqrt{N}}]$, STOP. Otherwise, proceed to step 4(d).
   (d) If k is equal to the iteration limit, stop. Otherwise, proceed to step 5.
5. Solve the backward pass:
   For each $t = T-1, \ldots, 1$ and $i = 1, \ldots, N$:
   (a) Solve the LP problem associated with each node in stage $t + 1$ that belongs to the set of descendent nodes of the node in stage t and scenario i;
   (b) Calculate the cut, which is going to be used in stage t by all scenarios $i = 1, \ldots, N$.
6. Increment the iteration counter, $k = k + 1$, and go to step 2.

One important characteristic of the SDDP is the sharing of cuts described in step 5(b). In the SDDP, the Benders cuts must be shared among all nodes in a given stage; this means that a cut created for a particular scenario, at stage t, must be added to all nodes in the stage t. Therefore, instead of adding one cut each scenario per



ξ_1^a
ξ_2^a ξ_2^b ξ_2^c
ξ_3^a ξ_3^b ξ_3^a ξ_3^b ξ_3^a ξ_3^b
ξ_4^a ξ_4^b ξ_4^c ξ_4^d ... ξ_4^a ξ_4^b ξ_4^c ξ_4^d ... ξ_4^a ξ_4^b ξ_4^c ξ_4^d

Fig. 1 Scenario tree with common samples

iteration, the SDDP adds N cuts per iteration, where N is the number of sampled scenarios in the forward pass. We refer to [16] for further discussion on issues of cut-sharing, and to [5] for a detailed discussion on the reasons why cut sharing is necessary in the SDDP algorithm. Figure 1 shows a strategy to ensure sharing of cuts among scenarios, whereby the scenario tree is constructed with common samples (see also [4]). In a scenario tree with common samples, a single set of observations is sampled for each stage and the same set is used for all nodes in the same stage. Of course, this assumes that the random variables sampled in each stage are independent of the random variables in previous stages. While not all MSSPs possess such a feature, the assumption holds when the underlying stochastic process $\xi := (\xi_1, \dots, \xi_T)$ is driven by a time series model of the form

$$\xi_{t+1} = h(\xi_0, \dots, \xi_t, U_t), \tag{6}$$

where $U_0, \dots, U_{T-1}$ are independent uniform random vectors in $(0, 1)^d$, and ξ_0 is a constant. Note also that the assumption of a uniform distribution is not restrictive, since one can use methods such as the inverse transform to generate values for other distributions. The class of problems for which (6) holds is very broad; see, e.g., [13]. In particular, the LTHSP fits this framework, since inflows are well represented by a periodic auto-regressive (PAR) model [26].

3 A discussion on the stopping criterion

One difficulty that arises when using sampling-based methods for stochastic optimization in practical settings is how to decide on an appropriate number of samples. In the context of the SAA approach, some theoretical estimates are given in [38] which lead to important insights about the complexity of sampling approximations, although such estimates are typically too conservative for practical use. This issue is directly related to the choice of an adequate *stopping criterion* for the optimization algorithm used to solve the problem—while many methods can be shown to converge to the correct values as the sample size goes to infinity, one must deal with the fact that any algorithm must stop after a finite number of iterations using a finite sample size. Therefore, the question that arises is how the estimation error that results from using a finite sample size can be taken into account when devising criteria for stopping the algorithm. Some early discussion about stopping criteria when using sampling methods can be found in [24] and [37].

In Sect. 2.2 we described the SDDP algorithm and its stopping criterion. We now present brief discussion of the stopping criterion as a *hypothesis test*, and argue that such stopping criterion has some flaws. Later, we will propose an additional criterion that aims at overcoming the flaws in the current test.

3.1 The stopping criterion as a hypothesis test

We start by formulating the stopping criterion in terms of a hypothesis test. For completeness and to set the notation used later, we quickly review some basic notions (see, e.g., [41] for a detailed discussion). A hypothesis test usually takes the form

$$H_0 : \theta \in \Theta \quad \text{against} \quad H_1 : \theta \notin \Theta,$$

where θ is an unknown parameter of the underlying distribution and Θ is a given set. To test H_0, we compute a test statistic T, a certain region R and proceed as follows: if $T \in R$, then we *reject* the hypothesis; if $T \notin R$, then we retain (i.e., do not reject) the hypothesis. Two types of error may occur as a consequence of a hypothesis test: a *type I error* occurs when H_0 is rejected even though it is true; a *type II error* occurs when H_0 is retained even though it is false. The probabilities of type I and II errors are closely related to two quantities:

- The *power function* of a test is defined as

  $$\beta(\theta) := P_\theta(T \in R),$$

  where P_θ indicates the probability of an event for a given value of the parameter θ. Note that when $\theta \notin \Theta$—i.e., when H_0 is false—the quantity $\beta(\theta)$ gives the probability that the test is correctly rejected. Thus, in that case the value $1 - \beta(\theta)$ gives the probability of a type II error for a given value of θ.

- The *size* of a test (also called *significance level*) is defined as

  $$\alpha := \sup_{\theta \in \Theta} \beta(\theta).$$

  We can see that α can be interpreted as a "worst-case" probability of a type I error.

Typically, the significance level of a test is fixed to a given value (say, 5%), which in turn determines the region R and consequently the power function $\beta(\theta)$. It is clear from the above definitions that there is a trade-off between the significance level and the power of a test. Indeed, H_0 is rejected only when there is enough evidence to reject it with sufficiently high confidence—but the higher the confidence required, the more likely that H_0 is retained when H_0 is false, which is a type II error.

The stopping criterion described in [31] can be implicitly viewed as the following hypothesis test:

$$H_0 : \text{Current solution is optimal} \quad \text{against} \quad H_1 : \text{Current solution is not optimal.}$$

More specifically,

$$H_0 : z_U = z_L \quad \text{against} \quad H_1 : z_U \neq z_L.$$

Note that z_L is a known value (given the information generated in the previous iterations), whereas z_U is the mean of the distribution of z^i defined in (5). To test the above hypothesis, we use the test statistics $\bar{z}_N$ defined in (5) and a region R of the form $R = (-\infty, z_L - c) \cup (z_L + c, \infty)$ for some $c > 0$. Assuming that $\bar{z}_N$ has approximately a Normal distribution—which holds by the Central Limit Theorem under mild assumptions—we have

$$\begin{aligned}\beta(z_U) &= P_{z_U}(\bar{z}_N > z_L + c) + P_{z_U}(\bar{z}_N < z_L - c) \\ &= P\left(\frac{\bar{z}_N - z_U}{\sigma/\sqrt{N}} > \frac{z_L - z_U + c}{\sigma/\sqrt{N}}\right) + P\left(\frac{\bar{z}_N - z_U}{\sigma/\sqrt{N}} < \frac{z_L - z_U - c}{\sigma/\sqrt{N}}\right) \\ &\approx 1 - \Phi\left(\frac{z_L - z_U + c}{\sigma/\sqrt{N}}\right) + \Phi\left(\frac{z_L - z_U - c}{\sigma/\sqrt{N}}\right),\end{aligned} \tag{7}$$

where σ^2 is the variance of z^i and Φ is the cumulative distribution function of the Normal(0, 1) distribution. Moreover, we have

$$\begin{aligned}\alpha = \beta(z_L) &\approx 1 - \Phi\left(\frac{c}{\sigma/\sqrt{N}}\right) + \Phi\left(\frac{-c}{\sigma/\sqrt{N}}\right) \\ &= 2\left(1 - \Phi\left(\frac{c}{\sigma/\sqrt{N}}\right)\right).\end{aligned}$$

Thus, for a fixed significance level α we have

$$c = \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\frac{\sigma}{\sqrt{N}}, \tag{8}$$

so we reject H_0 (optimality of the current iterate) if and only if $\bar{z}_N > z_L + c$ or $\bar{z}_N < z_L - c$, i.e., we retain H_0 (therefore accepting optimality) if and only if

$$\bar{z}_N \in [z_L - c,\ z_L + c]$$

or, equivalently,

$$z_L \in [\bar{z}_N - c,\ \bar{z}_N + c],$$

where c is given in (8). Noting that $[\bar{z}_N - c,\ \bar{z}_N + c]$ is a $(1-\alpha)\%$ confidence interval for z_U, we see that this is precisely the test conducted in [31] (with σ replaced with the sample standard deviation).

Let us look at the power of the above test when the current solution is *not* optimal, i.e., when H_0 is false. Using the value of c defined in (8) and denoting $\nu_{1-\alpha/2} := \Phi^{-1}(1-\alpha/2)$ we obtain from (7) that

$$\beta(z_U) \approx 1 - \Phi\left(\frac{z_L - z_U}{\sigma/\sqrt{N}} + \nu_{1-\alpha/2}\right) + \Phi\left(\frac{z_L - z_U}{\sigma/\sqrt{N}} - \nu_{1-\alpha/2}\right).$$

It easy to see that $\beta(\cdot)$ is an increasing function on $\{z_U > z_L\}$. That is, the power of the test is higher when either (i) z_U is much bigger than z_L, or (ii) the sample size N is large. This demonstrates why such a stopping criterion is inadequate—we accept optimality when H_0 is retained, but that could happen in situations where the power of the test is low. In such cases it is likely that the algorithm is stopped prematurely (i.e., a type II error occurs). Of course, the power of the test can be increased by increasing the value of α, but as discussed earlier that comes at the expense of a larger type I error probability. In the present context, a type I error means not identifying optimality, which implies unnecessary calculations. Shapiro [36] also points out to problems with this stopping criterion in his analysis of the SDDP method.

Hypothesis tests are usually formulated to protect from type I error, which means that we reject H_0 only when there is enough evidence. That is, the above test avoids proceeding to the next iteration of the SDDP unless there is enough evidence that we have not achieved the convergence. Consequently, if H_0 is retained we do not have any confidence that the algorithm should actually stop. To overcome this problem, an alternative hypothesis test is proposed in the next section.

3.2 A one-sided test of NON-optimality

One problem with the test described above is that it takes optimality as the "status quo," which is rejected only if there is enough evidence against it. As seen earlier, this may lead to situations where the convergence is accepted without any evidence for it. Therefore, it is natural to consider an alternative hypothesis test where the status quo is non-optimality. In that case, the algorithm will stop only if there is enough evidence for optimality—a notion that is closer to what is desired for a good stopping criterion. Define then

H_0 : Current solution is not optimal against H_1 : Current solution is optimal,

which can be written as

$$H_0 : z_U > z_L \quad \text{against} \quad H_1 : z_U = z_L.$$

By doing similar calculations as before, we see that we reject H_0 (and therefore accept optimality) if and only if

$$z_L \geq \bar{z}_N + \nu_{1-\alpha}\frac{\sigma}{\sqrt{N}}.$$

By replacing σ with its sampling estimate, the test becomes

$$\bar{z}_N \leq z_L - \nu_{1-\alpha}\frac{s_N}{\sqrt{N}}. \tag{9}$$

Note that, in this setting, stopping the algorithm prematurely is equivalent to a type I error. We then have two alternatives:

(a) Fix a value for α and test condition (9) until it is satisfied, or

(b) Compute the *p-value* of the test. This is the smallest value of α for which the test is rejected. This can be obtained by solving (for α) the equation

$$\bar{z}_N = z_L - \nu_{1-\alpha} \frac{s_N}{\sqrt{N}},$$

which yields

$$\alpha = 1 - \Phi\left(\frac{z_L - \bar{z}_N}{s_N/\sqrt{N}}\right).$$

The smaller the p-value is, the stronger the evidence that the hypothesis is false. This is because, as seen earlier, α can be interpreted as a "worst-case" probability of a type I error.

It is clear that this test is much more conservative—it stops only when $\bar{z}_N$ is sufficiently smaller than z_L, unless one is willing to accept a high p-value. Indeed, the power of this test is

$$\beta(z_U) = \Phi\left(\frac{z_L - z_U}{\sigma/\sqrt{N}} - \nu_{1-\alpha}\right),$$

so we see that $\beta(z_U) \le \Phi(-\nu_{1-\alpha}) = \Phi(\nu_\alpha) = \alpha$. Note that higher sample sizes make the test harder to be rejected unless the current solution is optimal; thus, it may be more difficult to identify near-optimal solutions.

Although this stopping criterion appears to be the best way to avoid premature stops, convergence may not be achieved in most cases because the test is too conservative. In such cases the SDDP algorithm will stop when it reaches the maximum iteration limits, which means that it has possibly run more iterations than necessary, thus increasing the computational burden.

3.3 Controlling both type I and type II errors

The hypothesis tests described in Sects. 3.1 and 3.2 both have deficiencies. In the former we have no evidence of convergence when the algorithm stops, whereas in the latter we have evidence of convergence when the algorithm stops but in most cases the algorithm will probably not stop. In order to overcome those difficulties, we propose to keep the first test and incorporate the probability of a type II error into the stopping criterion.

It follows from the discussion in Sect. 3.1 that the probability of a type II error (i.e., stopping the algorithm prematurely) *increases* when either z_U is close to z_L or the sample size is small. Of course, if z_U is close to z_L then the current solution is good; thus, we can measure the quality of the stopping criterion by evaluating how far z_U must be from z_L to yield a sufficiently low value of the probability of a type II error. Also, notice that, since z_U is an upper bound for the optimal value and z_L is a lower bound, we must have $z_U \ge z_L$. Thus, we can test the one-sided hypothesis

$$H_0 : z_U \le z_L \quad \text{against} \quad H_1 : z_U > z_L.$$

The calculations for the probabilities of type I and type II errors are very similar to those above. The idea is then to compute a tolerance based on a pre-specified upper bound γ on the probability of a type-II error when $z_U/z_L > 1+\delta$. That is, we bound the probability of incorrectly identifying optimality when the true upper bound exceeds the lower bound by more than a percentage δ. This yields

$$\delta = (\nu_{1-\alpha} + \nu_{1-\gamma})\frac{s_N}{z_L\sqrt{N}}. \tag{10}$$

For example, if the user is satisfied as long as the upper bound is no more than 10% above the lower bound, then a value of $\delta \le 0.1$ (for satisfactory pre-specified values of α and γ) indicates a good test, since in that case we know that the probability of stopping the algorithm prematurely when the upper bound is above that tolerance is at most γ.

To summarize, at each iteration we proceed as follows.

1. Compute $r_N := \bar{z}_N/z_L$ (note that we may have $r_N < 1$).
2. Compute the quantity

$$\rho_N := r_N - \nu_{1-\alpha}\frac{s_N}{z_L\sqrt{N}} = \frac{1}{z_L}\left(\bar{z}_N - \nu_{1-\alpha}\frac{s_N}{\sqrt{N}}\right), \tag{11}$$

   where $\bar{z}_N$ and s_N are defined in (4)–(5).
3. If $\rho_N > 1$, then the stopping criterion is not satisfied, so we proceed to the next iteration; otherwise, check the tolerance δ in (10). If it is satisfactory, then STOP; otherwise, we suggest to either increase α or γ, or increase the sample size N, and then re-compute $\bar{z}_N$, s_N and the confidence interval.[2]

4 Using alternative sampling techniques

We discuss now the use of alternative sampling techniques for the multi-stage algorithm. We start with a brief description of Quasi-Monte Carlo and Latin hypercube sampling, which follows the discussion in [15] but is included here for completeness. In the context of the multi-stage problem discussed earlier, both QMC and LHS can in principle be used in two ways—to construct a scenario tree from the original stochastic process or to sample scenarios from the tree. These strategies are described next.

4.1 Quasi-Monte Carlo

Let U be an s-dimensional random vector with uniform distribution on $(0,1)^s$, $f : (0,1)^s \longrightarrow \mathbb{R}$ an arbitrary function and consider the problem of estimating $I := \mathbb{E}[f(U)]$. The basic idea of QMC is to calculate a sample average estimate as in the standard Monte Carlo but, instead of drawing a random sample from the

[2]In our current implementation, we simply proceed to the next iteration if δ is not satisfactory.

uniform distribution on $(0, 1)^s$, a certain set of points $u^1, \dots, u^N$ on space $(0, 1)^s$ is carefully chosen. The deterministic estimate

$$I_{\text{QMC}} := \frac{1}{N} \sum_{i=1}^{N} f(u^i) \tag{12}$$

is constructed. A key result is the so-called Koksma-Hlawka inequality which, roughly speaking, states that the quality of the approximation given by I_{QMC} depends on the quality of the chosen points (measured by the difference between the corresponding empirical measure and the uniform distribution, which is quantified by the so-called *star-discrepancy*) as well as on the nature of the function f (measured by its total variation). A great deal of the research on QMC methods aims at determining ways to construct *low-discrepancy sequences*, i.e., sequences of points $u^1, u^2, \dots$ for which the star-discrepancy is small for all N. Many different methods to generate low-discrepancy sequences exist. Two of the most popular families of methods are *lattice rules* and *digital nets*; we refer to [25] and [19] for comprehensive reviews.

Figure 2 shows an example of a set of 27 QMC points on $(0, 1)^2$. Note that there are 9 major squares; each of them contains exactly 3 points (points on the left corner are considered part of a square). Moreover, each major square is divided into 3 rows and 3 columns, and each row and each column of a square contains exactly 1 point. We can see that there is a reasonable spread of the points over $(0, 1)^2$—unlike a set generated with standard Monte Carlo, which is likely to exhibit clusters of points in some parts and "holes" in other parts.

QMC methods have proven very effective in practice, in the sense that they yield much better estimates of the quantity I above. One disadvantage, however, is the difficulty to quantify the error of the obtained estimates, since the Koksma-Hlawka inequality, while useful from a theoretical perspective, cannot be used to determine errors as it depends on constants that are very difficult to calculate in

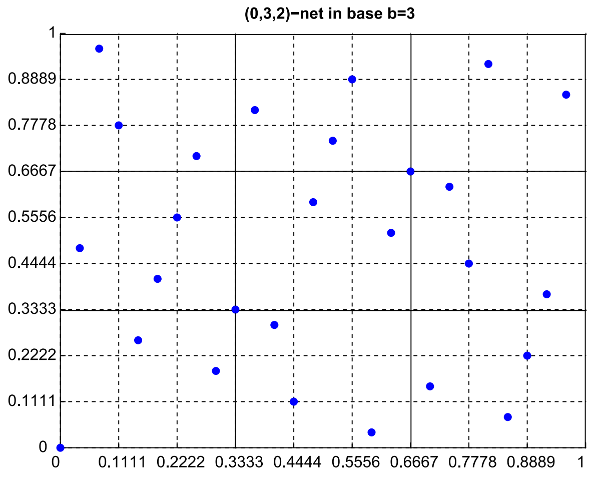


Fig. 2 A quasi-Monte Carlo set of points on $(0, 1)^2$

practice. A common way to overcome the problem is to incorporate some randomness into the choice of QMC points. By doing so, errors can be estimated using standard methods, e.g., via multiple independent replications. This is the main idea of *randomized* QMC methods (RQMC), see again [19] for a detailed discussion.

A few papers study the use of QMC methods with an SAA approach for a stochastic optimization problem of the form $\min_{x \in X}\{g(x) := \mathbb{E}[G(x, \zeta)]\}$. In [17], empirical results are provided for the use of Hammersley sequences (one form of QMC). In [29] and [30], the authors use the fact that the empirical measure defined by a QMC sequence converges weakly to the uniform distribution to show that, under mild assumptions, the estimator function for g constructed with QMC points *epiconverges* to the true function g, which guarantees convergence of optimal values and optimal solutions under appropriate further conditions; in [18] those results are applied to the case where the QMC sequence is randomized with the so-called Cranley-Patterson procedure. The numerical results in those papers (and also those in [13]) suggest considerable gains in terms of rates of convergence when using QMC methods. Homem-de-Mello [15] provides a rigorous analysis of the improvements gained by the use of a particular QMC method in terms of rates of convergence. The results, which are corroborated by numerical experiments, show that the gain can be enormous—more specifically, it is shown that, under appropriate assumptions, the sequence formed by the optimal values of the sampling problems converges at a rate of order $[(\log N)^{s-1}/N^3]^{1/2}$, which asymptotically is much better than $N^{-1/2}$ given by standard Monte Carlo.

4.2 Latin hypercube

Similarly to QMC methods, Latin hypercube sampling (LHS) is a strategy that aims to construct a sequence of points with low discrepancy—indeed, LHS can be viewed as a particular type of randomized QMC. This method, which was proposed in [23], has been thoroughly studied in terms of its theoretical properties and numerical efficacy (see, e.g., [21, 27, 40]) and can be described as follows. Suppose we would like to sample N values from the uniform distribution on $(0, 1)$. The idea of the method is to divide the $(0, 1)$ interval into N subintervals with the same width $\frac{1}{N}$. Inside each subinterval one random value is sampled. The resulting set of values is then randomly permuted, yielding a sequence that corresponds to samples $U^1, \ldots, U^N$. When sampling from the uniform distribution on $(0, 1)^s$, we repeat the procedure for each dimension. Figure 3 shows an example of four samples on $(0, 1)^2$.

The LHS method has been successfully used in stochastic optimization problems of the form $\min_{x \in X}\{g(x) := \mathbb{E}[G(x, \zeta)]\}$, see for instance [1, 20, 38]. Although [15] shows that the sequence formed by the optimal values of the sampling problems converges at the same of $N^{-1/2}$ given by standard Monte Carlo, the resulting estimates typically have smaller variance than those obtained with standard Monte Carlo, thus yielding a better performance of the optimization algorithm.

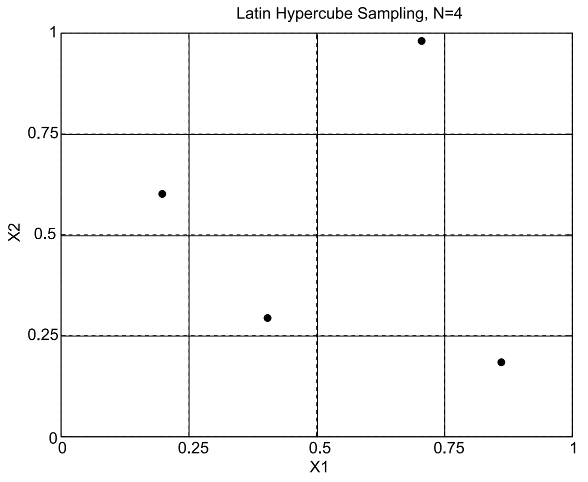


Fig. 3 Example of a Latin hypercube sample on $(0, 1)^2$

4.3 Sampling trees

We return now to multi-stage stochastic programs, and describe a procedure to generate a scenario tree from the original underlying stochastic process using QMC and LHS techniques. As discussed earlier, we assume that the underlying stochastic process $\xi := (\xi_1, \dots, \xi_T)$ is driven by a time series model of the form

$$\xi_{t+1} = h(\xi_0, \dots, \xi_t, U_t),$$

where $U_0, \dots, U_{T-1}$ are independent uniform random vectors in $(0, 1)^d$, and ξ_0 is a constant. The tree generation procedure is described as follows:

1. Let n_t denote the number of branches to be generated at each node in stage t;
2. For each $t = 0, \dots, T-1$:
   (a) Generate n_t Uniform(0, 1) d-dimensional random vectors $U_t^1, \dots, U_t^{n_t}$ using an RQMC method;
   (b) For each node in stage t, generate n_t children nodes using the relation (6) for each U_t^j, $j = 1, \dots, n_t$.

Note that the use of a randomized QMC method in step 2(a) ensures that independent trees can be generated by repeating the above procedure.

4.4 Sampling scenarios from a tree

As mentioned earlier, QMC can be used in the SDDP algorithm to sample N_E scenarios from the underlying tree. To the best of our knowledge, no applications of QMC methods to internal sampling in multi-stage problems have been reported in the literature. We propose the following algorithm:

1. Let $M \geq 5$ be a divisor of N_E.
2. For each $j = 1, \dots, M$:

   (a) Generate $N' := N_E/M$ Uniform(0, 1) T-dimensional vectors $V_j^1, \dots, V_j^{N'}$ using an RQMC method.
   (b) For each $k = 1, \dots, N'$:
      i. For each $t = 0, \dots, T-1$, use the $t+1$st component of the vector V_j^k to choose a number ℓ between 1 and n_t, and choose $\hat{\xi}_{t+1}^k$ as the ℓth child of the current node;
      ii. Generate solutions corresponding to scenario $\hat{\xi}^k = (\hat{\xi}_1^k, \dots, \hat{\xi}_T^k)$.
   (c) Calculate the cost of each of the N' solutions as in (3) and average the values as in (4). Denote the average as $\bar{z}^j$.
3. Average the values $\bar{z}^1, \dots, \bar{z}^M$ and compute the sample variance. Use these values in the test of stopping criteria.

Again, the use of a randomized QMC method in step 2(a) ensures that independent streams of numbers can be generated, so the calculation of confidence intervals is valid.

Note that in the above algorithm RQMC is applied to a T-dimensional vector. A well-known issue with QMC sampling is the fact that it can be computationally expensive to generate QMC point sets for high dimensions. Also, while the rate of convergence under QMC is asymptotically superior to the Monte Carlo rate of $\frac{1}{\sqrt{N}}$, it is dependent on the dimension and often does not become superior to Monte Carlo until N is extremely large. In practice, however, QMC often outperforms its theoretical rate of convergence, which suggests that the sample spaces of these problems are really in some lower dimension—i.e., that some random variables are considerably more important to the problem than others. This leads to the concept of *padding*, where the important variables are integrated using a QMC or randomized QMC point set and the remaining variables are integrated using something computationally less expensive such as the midpoint of the interval, a Monte Carlo sample [39], or a Latin Hypercube sample [28]. Determining the important variables, however, is a non-trivial task especially in the context of stochastic optimization, since the integrand also depends on the decision variables. Drew and Homem-de-Mello [9] discuss some strategies in the context of two-stage stochastic programs, but an extension of those ideas to multistage problems requires further investigation.

A heuristic argument that can potentially work well for multistage problems goes as follows. Recall from the above discussion that we have a T-dimensional random vector, i.e., one component per stage. It is reasonable to think that the most important components are the ones corresponding to the initial stages. Thus, when generating the vectors $V_j^1, \dots, V_j^{N'}$ in step 2(a), we suggest applying RQMC to the first t_0 components, using standard Monte Carlo or (even better) Latin Hypercube for the remaining ones. Some experimentation is required to find a good value for t_0, but we suggest starting with $t_0 = 6$. Another possibility is to choose t_0 equal to the number of terms in the auto-regressive process given by (6).

5 Numerical experiments

In order to evaluate the impact of the alternative sampling strategies and the additional stopping criterion in the LTHSP, we used the model SMERA (Stochastic Model for Energy Resource Allocation), which has been developed by LabPlan since 2007. We have incorporated all of the strategies proposed in this paper into the program. In SMERA the hydro plants are aggregated in Energy Equivalent Reservoirs (EERs), and the resulting multi-stage stochastic program is solved by means of the SDDP strategy (see [5, 6] for details). The energy inflows can be generated either by an interstage independent model or via a periodic auto-regressive (PAR) model. As discussed in [26], the PAR model is the most suitable to forecast monthly inflows and therefore it would be a natural choice to generate the scenario tree; however, because the PAR model adds nonlinearities to the LTHSP, for the purpose of this study we assume that the energy inflows are stagewise independent with a lognormal distribution. Further discussion on the issues that arise when using the PAR model can be found in [5].

The contributions presented in this paper were tested in the Brazilian interconnected power system, which contains almost 180 hydro and 150 thermal plants. The hydro plants were aggregated into four EERs, corresponding to geographic regions. We considered a three-year horizon and the scenario tree was built with a group of 20 descendent nodes for each node and stage. Note that, with such a construction, the total number of scenarios in the tree is $20^{36} \approx 6.9 \times 10^{46}$. Obviously, it is impractical to enumerate all scenarios, which justifies the application of sampling-based algorithms such as SDDP. As discussed earlier, sampling is performed in two steps of the algorithm: when sampling from the original lognormal distribution to generate the scenario tree and when sampling scenarios from the tree to proceed with the forward and backward SDDP passes. In order to get an idea of the combined effect of the various sampling strategies we applied each strategy to each of the two steps. As a consequence, we studied 15 cases, considering the three sample strategies (MC, LHS and RQMC) for scenario generation and the same three strategies plus a padding strategy with RQMC in the first few stages and LHS in the rest of the stages when sampling the scenarios for the SDDP algorithm. Those cases are detailed in Table 1. In the table, M and N' indicate the number of independent blocks and the number of samples in each block. As discussed in Sect. 4, such a division in blocks is necessary in case of the LHS and RQMC strategies in order to get confidence intervals for the estimates, since the samples are not independent. The total number of samples is thus $N = MN'$.

As one can see from Table 1, in all cases there were 200 scenarios in forward and backward passes, which were re-sampled at every iteration. In addition, the number of iterations was limited to a maximum of 30 to avoid extremely long computational times. The LTHSP was solved in a 4-processor computer, a Intel Core2 Extreme (Quad core) 3.00 GHz and 4 GB of RAM, with the use of the MPICH2 for handling the parallel processing. Additionally, the linear programming problems corresponding to each node were solved by CPLEX 11. For the RQMC method, we implemented the so-called randomized (t, s)-sequences, available from the public library of routines developed in [11].

Table 1 Sampling strategies for scenario tree and SDDP

Case	Scenario tree	SDDP	M	N'	Padding in stage
1	MC	MC	200	1	–
2	MC	LHS	5	40	–
3	MC	RQMC	5	40	–
4	MC	RQMC + LHS	5	40	12
5	MC	RQMC + LHS	5	40	6
6	LHS	MC	200	1	–
7	LHS	LHS	5	40	–
8	LHS	RQMC	5	40	–
9	LHS	RQMC + LHS	5	40	12
10	LHS	RQMC + LHS	5	40	6
11	RQMC	MC	200	1	–
12	RQMC	LHS	5	40	–
13	RQMC	RQMC	5	40	–
14	RQMC	RQMC + LHS	5	40	12
15	RQMC	RQMC + LHS	5	40	6

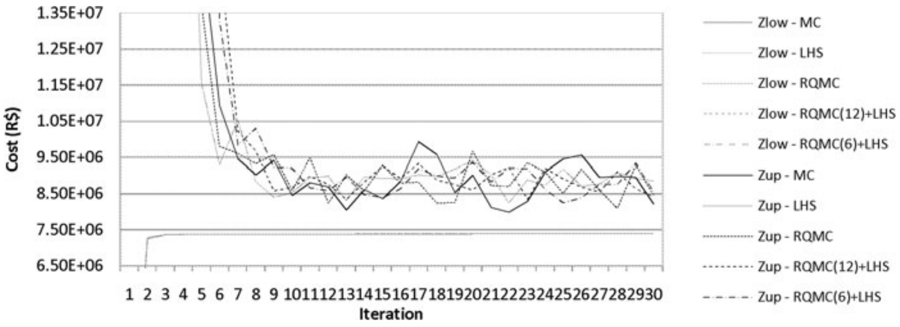


Fig. 4 Close look at upper and lower bounds, cases 1 to 5, scenario tree sampled with MC

Recall from the algorithm described in Sect. 2.2 that each iteration produces a lower bound z_L and an estimate $\bar{z}_N$ of an upper bound z_U for the optimal value of the problem. To facilitate the identification, we shall call the lower bound “Zlow” and the upper bound estimate “Zup” in the sequel. We first present the values of Zlow and Zup bound for each of the 15 cases listed in Table 1. We present one figure for each of the sampling strategies used to build the scenario tree; each figure shows the results corresponding to that tree for all of the five sampling strategies associated with sampling of scenarios. Thus, Fig. 4 presents the upper and lower bounds for cases 1 to 5, in which the scenario tree was sampled with the Monte Carlo strategy. Likewise, Fig. 5 illustrates the cases with the LHS scenario tree (cases 6 to 10) and Fig. 6 shows the cases with the RQMC scenario tree (cases 11 to 15). The upper bound has

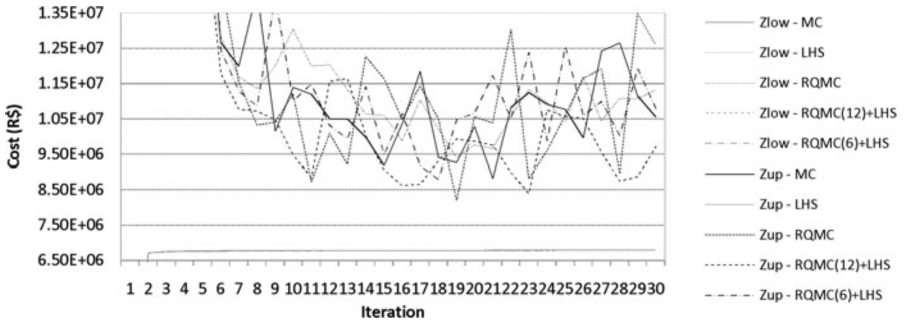


Fig. 5 Close look at upper and lower bounds, cases 6 to 10, scenario tree sampled with LHS

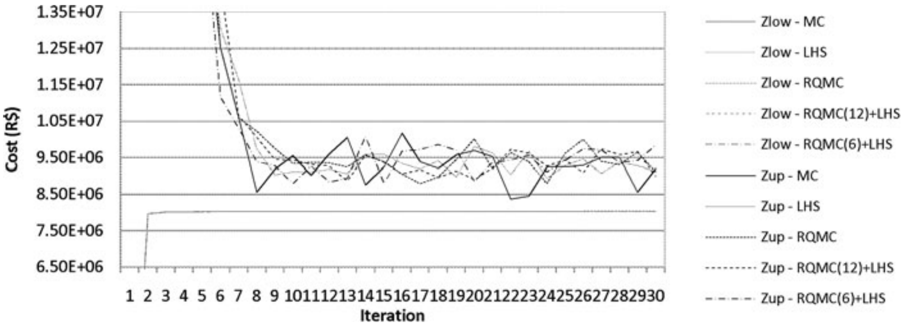


Fig. 6 Close look at upper and lower bounds, cases 11 to 15, scenario tree sampled with RQMC

been estimated using the information obtained in the forward pass scenarios at each iteration.

One can observe that Zup oscillates significantly above Zlow in all cases, which suggests that convergence has not been achieved, even if we take into account the statistical error of the upper bound estimate. This problem is worse in the case of the LHS scenario tree (Fig. 5), but we could not find a plausible explanation for this behavior.

On the other hand, the lower bound Zlow is almost the same within each scenario tree, but different among the trees. The former indicates that the scenarios sampled in the forward pass across iterations generated similar cuts regardless of the method used, whereas the latter indicates that the statistical error associated with sampling scenario trees from the original process may be significant. The numbers show that from the 10th iteration onward the lower bound increases very little, which indicates that the operation policy improves slowly towards the optimum. In that sense, the upper bound oscillation might be associated with the statistical error of the estimate and does not convey information about convergence itself.

The variability in the lower bound across scenario trees also highlights the fact that, since only one scenario tree was generated for each sampling strategy, we cannot measure how much the scenario tree is representative of the original lognormal distribution. Thus, even though all three scenario trees were sampled from the same original lognormal distribution, we cannot compare cases with different scenario trees directly. In order to compare the scenario tree sampling strategies it would be necessary to evaluate the operation policy in terms of the original distribution as discussed in [35] and [4], but that is not the focus of this paper. As a consequence, our analysis is focused mainly on the performance of the forward pass sampling strategies, although we do make some remarks about the scenario tree generation methods according to results that were obtained in our study but not presented here due to space limitation.

The lack of convergence suggested by Figs. 4, 5, 6 can be made more clear by analyzing the lower limit ratio ρ_N defined in (11), which is shown in Figs. 7, 8, 9. Recall from the discussion in Sect. 3 that a value of ρ_N bigger than 1 corresponds

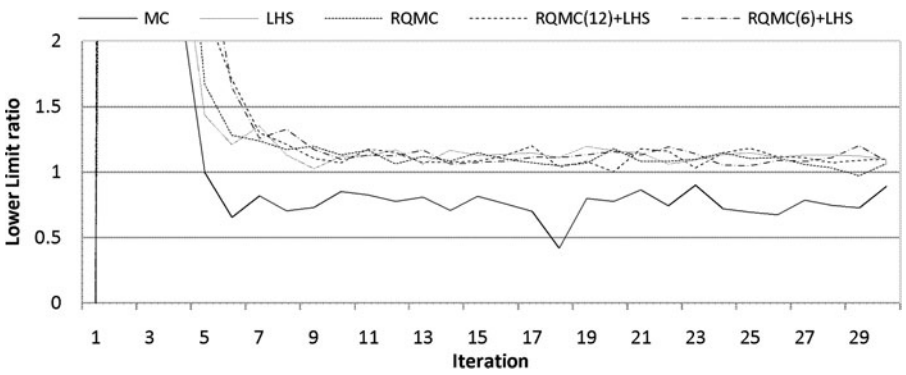


Fig. 7 Lower limit ratio, cases 1 to 5, scenario tree sampled with MC

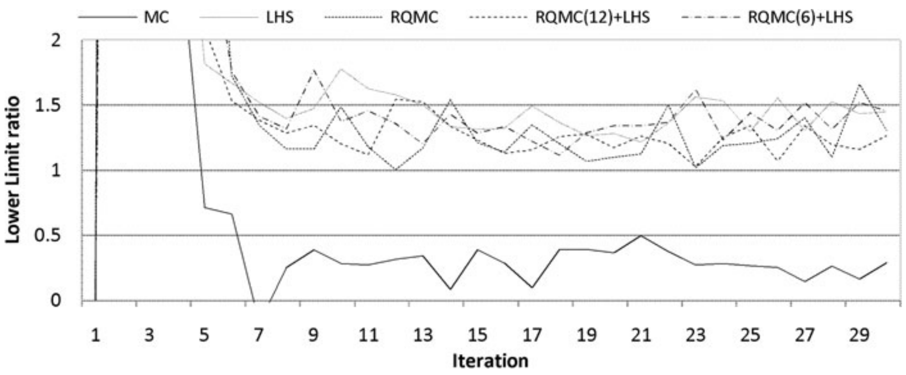


Fig. 8 Lower limit ratio, cases 6 to 10, scenario tree sampled with LHS

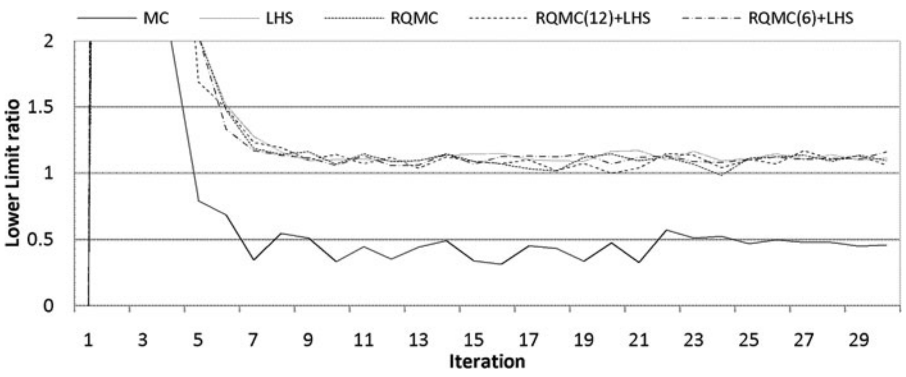


Fig. 9 Lower limit ratio, cases 11 to 15, scenario tree sampled with RQMC

to rejecting the hypothesis of optimality (we used a significance level $\alpha = 5\%$), indicating that convergence was not achieved. We can see that, out of 15 cases, only in three of them was the ratio ρ_N smaller than 1—those are the cases in which the SDDP scenarios are sampled with the Monte Carlo strategy. However, as discussed in Sect. 3 this phenomenon does not guarantee that we have a good operation policy for the given scenario tree, it only says that we do not have enough evidence that more iterations are necessary.

Therefore, in order to evaluate if the operation policy is a good one, we study the additional criterion described in Sect. 3.3. Recall that the criterion ensures an upper bound γ—we used 5%—on the probability of incorrectly claiming convergence when the true *relative* optimality gap is above δ. It is important to point out that the choice for δ depends on how much precision one requires. In our case, we consider that a value of $\delta \leq 0.1$ (i.e., a maximum 10% tolerance for the relative optimality gap) is adequate.

Figures 10, 11, 12 present the values of δ for all 15 cases, in which it is possible to notice that the δ is reasonably small in most of them. There are three exceptions, which are exactly the cases in which the SDDP scenarios are sampled with the Monte Carlo strategy. Therefore, although there is not sufficient evidence that it is necessary to proceed with the algorithm (since $\rho_N < 1$), the large values of δ indicate that the probability of stopping the algorithm prematurely when ρ_N reaches below 1 might be high.

These results show that it is dangerous to assume convergence in any of the 15 cases, even though the lower bound increases very slowly. There are two possible explanations for such phenomenon. One is that, because of the size of the problem, convergence of the cutting generation algorithm would take more than 30 iterations even if an exact method were used (i.e., if instead of sampling scenarios in the forward pass one could use all scenarios). Another possible explanation is that sampling 200 scenarios in the forward pass of SDDP may not be enough to generate cuts sufficiently fast. Recall that the convergence results in [32] (see also [36]) only guarantee that all necessary cuts will be *eventually* generated—but how fast that happens depends on a number of factors, including the sample size. Unfortunately, due to the size of the

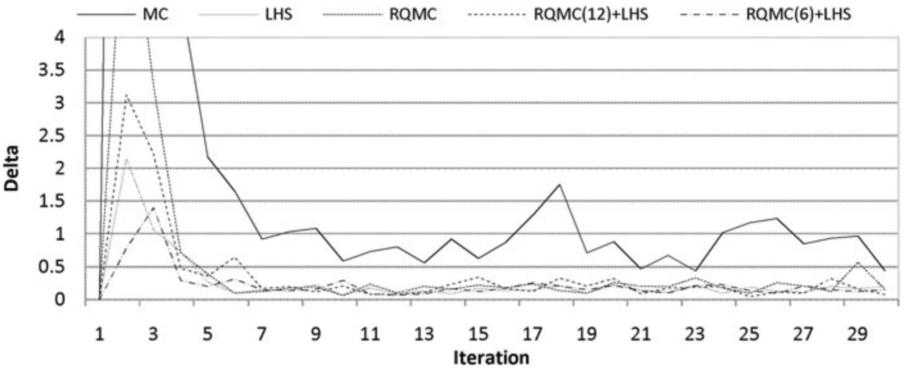


Fig. 10 δ for cases 1 to 5, scenario tree sampled with MC

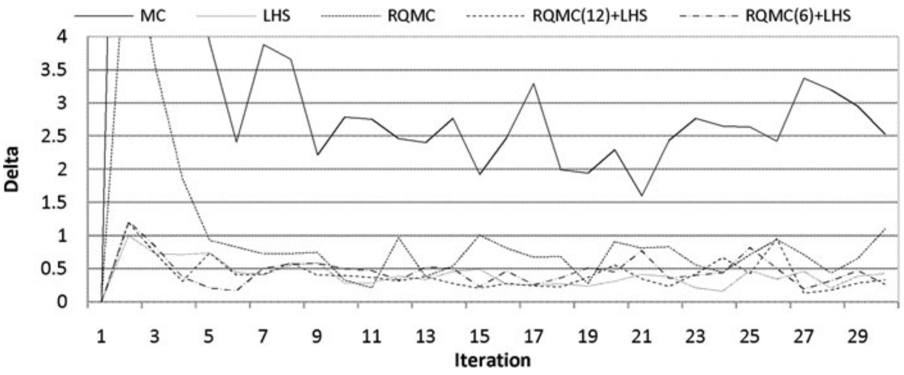


Fig. 11 δ for cases 6 to 10, scenario tree sampled with LHS

problem, increasing the number of sampled scenarios is not practical in terms of computational burden, so consequently the algorithm was stopped after reaching a limit (thirty) on the number of iterations.

The above discussion demonstrates that the new stopping criterion proposed in this paper is effective in the sense that it correctly prevented the algorithm from stopping prematurely in our case study. However, we would like to analyze whether the criterion is effective in correctly stopping the algorithm as well. To do so, we studied two more cases, which correspond to a smaller version of the Brazilian LTHSP described earlier. In these problems, we reduced the horizon from 36 to 6 months, but we maintained the number of descendent nodes equal to 20 (so, the total number of scenarios in the tree is $20^6 \approx 64$ million). In addition, instead of sampling 200 scenarios for the SDDP forward pass, we sampled a total of 1000 scenarios, which was possible due to the smaller size of the problem. We studied two combinations of sampling methods: one in which both the scenario tree and the SDDP scenarios are

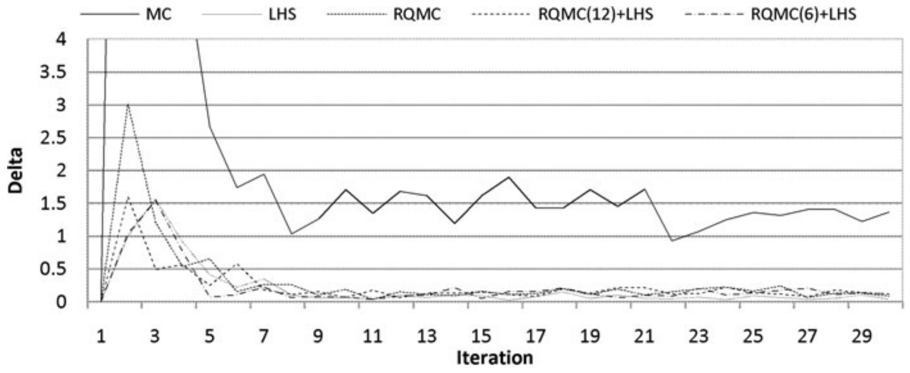


Fig. 12 δ for cases 11 to 15, scenario tree sampled with RQMC

sampled with the MC strategy—which is the standard approach in the literature—and another one where the scenario tree is sampled with the RQMC strategy and the SDDP scenarios are divided into 20 blocks of 50 scenarios each and sampled with the LHS approach. The rationale for choosing the RQMC/LHS combination was that, from the results for the larger problem, these methods seemed to produce the most stable results for scenario generation and SDDP scenario sampling, respectively.[3]

Figure 13 presents the upper and lower bounds for the two cases. It is clear that in the first case (MC/MC) the upper bound estimate is always above the lower bound and gets close to it only twice, whereas in the second case (RQMC/LHS) the upper bound estimate oscillates around the lower bound.

Figure 14 shows the lower limit ratio ρ_N. Here we see that in both cases the lower limit ratio is below 1 in almost all iterations, so there is no evidence that more iterations are necessary. Note also that in the MC/MC case the values of ρ_N are much below 1, in fact they are even negative. This may seem unexpected when comparing with Fig. 13 since the upper bound in the RQMC/LHS case is much closer to the lower bound than in the MC/MC case. The difference, of course, can be explained by the standard deviation, which is much bigger in the MC/MC case; for instance, the standard deviation in the last iteration was around 5.68×10^7 for MC/MC case and 4.30×10^5 for RQMC/LHS.

To check if the $\rho_N < 1$ condition implies convergence, we need to look at the values of δ, which is the tolerance on the relative optimality gap above which we

[3]By "most stable" we mean we mean the methods that yielded the lowest variance across iterations. That the RQMC/LHS combination was the most stable is not clear from the figures but it can be seen from a numerical table with the same information. As mentioned earlier, the choice for a method for the scenario tree must be taken cautiously since this is based on one sampled tree for each method. Nevertheless, other experiments we conducted seem to support that choice.

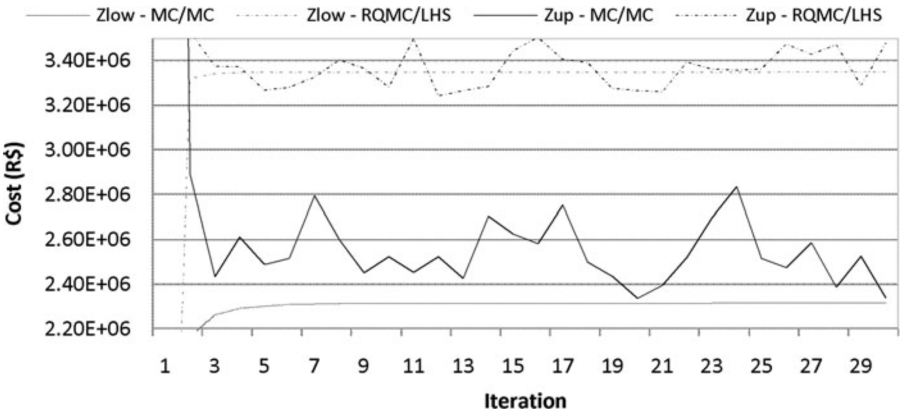


Fig. 13 Upper and lower bounds for the 2 cases with 6 stages

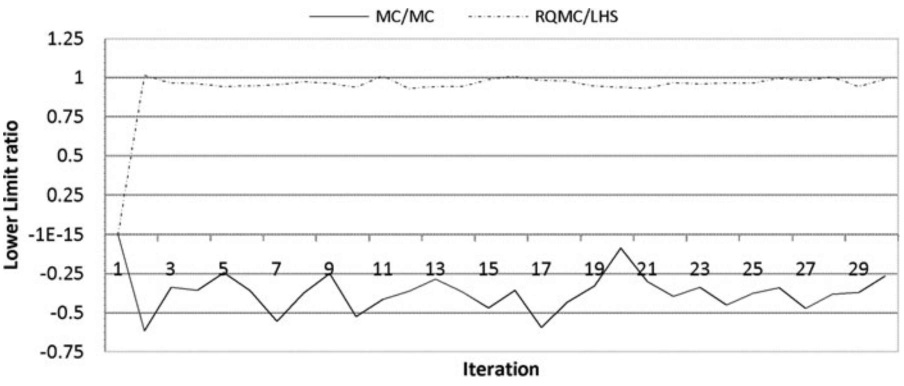


Fig. 14 Lower limit ratio for the 2 cases with 6 stages

can bound the probability of incorrectly claiming convergence. Figure 15 shows the values of δ for both the MC/MC and the RQMC/LHS cases for each iteration. As mentioned earlier, in our case we consider $\delta \leq 0.1$ to be satisfactory. We see that, as in the larger problem, the values of δ for the MC/MC case are quite large, so the probability of stopping the algorithm prematurely might be high. For the RQMC/LHS case, though, we see that $\delta \leq 0.1$ at all iterations. Thus, the stopping criteria described in Sect. 3 would stop the algorithm after just three iterations, since at that point we have $\rho_N < 1$ and $\delta \leq 0.1$. Figure 13 indeed suggests that stopping at that point is the correct decision, since from that point on the upper bound oscillates around the lower bound (for the purpose of exposition, however, we chose not to stop the algorithm so we could see the values of ρ_N and δ in further iterations). These examples demonstrate not only that the proposed stopping criterion is effective but also that the alternative sampling techniques—in particular the RQMC/LHS combination—seem to work very well for SDDP.

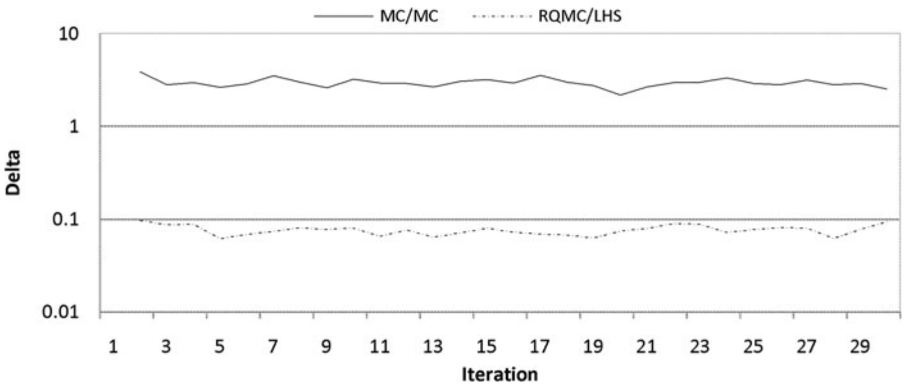


Fig. 15 δ for the 2 cases with 6 stages

6 Conclusion

This paper focuses on studying sampling strategies for a Multi-Stage Stochastic Programming (MSSP) problem and the stopping criterion used in the SDDP (Stochastic Dual Dynamic Programming) algorithm to solve the Long-term Hydrothermal Scheduling Problem (LTHSP). Initially, we presented a brief review of the MSSP theory and the SDDP algorithm, to emphasize the importance of the sampling strategies and of the stopping criterion. Because the MSSPs we are interested in are very large problems, it is necessary to incorporate sampling into the optimization algorithm, and the Monte Carlo sampling strategy is typically used. However, there are some variance reduction sampling strategies, such as randomized quasi-Monte Carlo (RQMC) and Latin hypercube sampling (LHS), which can produce more representative samples, thus enhancing the quality of the solution. We discussed the application of these alternative sampling strategies. We also proposed a novel stopping criterion based on statistical hypothesis testing. Our formulation of the SDDP stopping criterion as a hypothesis test allowed us to prove that the stopping criterion originally proposed for the algorithm is not sufficient to evaluate whether or not the convergence was achieved with high confidence. However, we showed that it is possible to improve the stopping criterion by adding a new criterion, which guarantees bounds on the probability of stopping the algorithm either too soon or too late. Although we presented the discussion in the context of the LTHSP framework, the methods we propose can be useful in other practical problems.

We evaluated our methods on a model previously proposed in the literature (the SMERA model) and we used the data for the Brazilian LTHSP, with a three-year planning horizon. The results showed that the RQMC and LHS strategies were more consistent producing an operation policy with a much smaller variance than the MC. In our view, the combination of using RQMC for building the scenario tree, and LHS for the SDDP sampling scenarios was the best configuration, although no definitive statements can be made based only on this experiment.

The proposed stopping criterion proved to be effective in avoiding stopping the algorithm prematurely (unlike the criterion originally proposed for SDDP). However, it was not possible to analyze whether the stopping criterion was effective in preventing the algorithm from running more iterations than necessary. This happened because, in all cases we analyzed, convergence did not occur with a pre-specified limit on the number of iterations. Therefore, in order to evaluate the new stopping criterion we reduced the planning horizon to six months, which enabled us to sample more scenarios within the SDDP. The results indicate that the additional stopping criterion is essential for the SDDP framework, because without it we would accept the convergence even though there was not much evidence that the operation policy was a good one. With the new criterion, the solution obtained upon stopping was indeed a good policy.

Finally, our recommendations can be summarized as follows. We propose to evaluate the quality of the solution produced by the algorithm (for a given scenario tree) by means of a test of the ratio of upper and lower bounds that takes into account a tolerance for the estimation error. We propose the application of LHS methods for the generation of scenarios from a given tree. RQMC methods can be used for that purpose as well if either the number of stages is small or an RQMC method that can handle well relatively high dimensional vectors is used. Finally, RQMC appears to work well for the generation of scenario trees if only low-dimensional vectors are required as it was the case in our problem, where the vectors were 4-dimensional.

It would be interesting also to evaluate solutions with respect to the problem corresponding to the original input process—the current and proposed methods evaluate solutions with respect to the problem corresponding to a scenario tree sampled from the original process, so an extension of such methods is in order. We are conducting some investigation with this focus. Also, it may be possible to tailor the QMC methods to exploit the structure of the problem, in particular to determine the proper components to which QMC techniques should be applied.

Acknowledgements The authors thank two anonymous referees for their valuable comments. This research was supported by Projeto de Pesquisa & Desenvolvimento Tractebel Energia GDF Suez—P&D ANEEL (0403-017 2007).

Appendix

In this Appendix we present the MSSP model for the LTHSP used in our analysis. The model is discussed in detail in [5], but we include it here for completeness.

The LTHSP formulation in the SMERA Model

The SDDP is a decomposition technique that solves each node, of the scenarios tree, separately; each one is modeled as a LP problem. In this sense, we present the formulation of the LTHSP for one particular node. The formulation of LTHS problem in the Brazilian case must take into account the EER representation and the load levels[1].

$$\omega_t = \min \sum_{j=1}^{NUT}\left(ct_j \cdot \sum_{q=1}^{NP} gt_{jqt}\right) + \sum_{k=1}^{NS}\left[\sum_{h=1}^{NDEF}\left(cd_{hk}\cdot\sum_{q=1}^{NP} d_{hkqt}\right)\right] + \frac{1}{(1+\beta/100)}\omega_{t+1}, \tag{1}$$

s.t.:

$$v_{r,t+1} + \sum_{q=1}^{NP} gh_{rqt} + s_{rt} = fv_{rt}v_{rt} - q_{rt}^{\min} + qc_{rt} - vm_{rt} - ev_{rt} + fc_{rt}a_r y_{rt}, \tag{2}$$

$$\begin{aligned}\sum_{r\in NR_k} gh_{rqt} + \sum_{j\in NUT_k} gt_{jqt} + \sum_{n\in\Omega_k}\left(f_{nkqt} - f_{knqt}\right) + \sum_{h=1}^{NDEF} d_{khqt} + exc_{qkt} = L_{kt}LF_{kqt}FP_{kqt} - \sum_{r\in NR_k}\left(q_{rt}^{\min} + pu_{rt}\right)FP_{kqt} \\ - \sum_{r\in NR_k}\left[\left(1-a_r\right)y_{rt} + qf_{rt}\right]FP_{kqt},\end{aligned} \tag{3}$$

$$gt_{jt}^{\min} FP_{k_j qt} \le gt_{jqt} \le gt_{jt}^{\max} FP_{k_j qt}, \tag{4}$$

$$0 \le d_{khqt} \le d_{kht}^{\max} FP_{kqt}, \tag{5}$$

$$0 \le f_{nkqt} \le f_{nkt}^{\max} FP_{kqt}, \tag{6}$$

$$0 \le f_{knqt} \le f_{knt}^{\max} FP_{kqt}, \tag{7}$$

$$v_{rt}^{\min} \le fv_{r,t+1}v_{r,t+1} \le v_{rt}^{\max}, \tag{8}$$

$$0 \le gh_{rqt} \le gh_{rt}^{\max} FP_{k_r qt} - \left[\left(1-a_r\right)y_{rt}\right]FP_{k_r qt} + qf_{rt}FP_{k_r qt} - q_{rt}^{\min}FP_{k_r qt}, \tag{9}$$

Where:

NUT, NP, NS, NDEF, NR	number of thermal plants, load levels, electrical subsystems, energy deficit levels and EERs;
j, q, k, n, h, r	thermal plants, load levels, electrical subsystems, energy deficit levels, EERs' indexes;
ct_j	thermal plants' *j* incremental costs (R$/MWmonth);
gt_{jqt}	thermal plants' *j* generations in load level *q* and stage *t* (MWmonth);
cd_{hk}	energy deficit levels' *h* incremental costs in subsystem *k* (R$/MWmonth);
d_{hkqt}	energy deficit levels' *h* generation in subsystem *k*, load level *q* and stage *t* (MWmonth);
gh_{rqt}	EERs' *r* generation in load level *q* and stage *t* (MWmonth);
a_r	coefficient to calculate the controllable part of the energy inflow *y*;
f_{nkqt}	energy interchange from electrical subsystems *n* to *k* in load level *q* and stage *t* (MWmonth);
L_{kt}	load of subsystem *k* in stage *t* (MWmonth);
LF_{kqt}	consumption factor[2] of subsystem *k* in load level *q* and stage *t* (%);
FP_{kqt}	load level factor of subsystem *k* in load level *q* and stage *t* (%);
exc_{kqt}	excess of energy of subsystem *k* in load level *q* and stage *t*;
$gt^{min}{}_{jt}$	minimum generation of thermal plant *j* in stage *t* (MWmonth);
$gt^{max}{}_{jt}$	maximum generation of thermal plant *j* in stage *t* (MWmonth);
$d^{max}{}_{kht}$	maximum energy assumed by deficit level *h* of subsystem *k* in stage *t* (MWmonth);
$f^{max}{}_{nkt}$	maximum interchange from electrical subsystems *n* to *k* in stage *t* (MWmonth);
ω_{t+1}	expected future cost of stage *t*+1 (R$);
β	discount rate (%).

Equation (1) represents the objective function, which is subject to the energy reservoir balance constraint for each EER (2), the load supply constraint for each electrical subsystem and load levels (3). The variables' limits are imposed by (4) to (9).

Table 1 presents all EER attributes, with their notation and a very brief description. The equations used in the model are presented in [1-2]. Basically, the EER attributes are calculated by the energy that could be produced

[1] The load curve is modeled into three levels: Light (low consumption period), Valley (average consumption period) and Peak (peak load period).

[2] The consumption factor is a value for each load level and electrical subsystem that multiplies the demand in order to have the energy to be supplied in the light, average and heavy periods of consumption. The sum of the three load level factors multiplied by its consumption factors must be equal to 100%, in each electrical subsystem.

by using the water associated with each one of the attributes presented in Table 1. In order to find the equivalent energy, the volume of water is multiplied by the sum of the average productivity of all hydro plants in the cascade through which the water flows.

Table 1 - EER attributes[3].

Name	Notation	Description (Energy that can be generated by…)
Stored Energy	v	the volume stored in the reservoirs
Maximum Stored Energy	v^{max}	the maximum volume of the reservoirs
Minimum Stored Energy	v^{min}	the minimum volume of the reservoirs
Energy Inflow	y	all hydro plants inflows
Evapored Energy	ev	the water evaporated from the reservoirs
Minimum Discharge Energy	q^{min}	the reservoirs' minimum water discharge
Production Before the Full Commitment	pu	the hydro plants that have just been constructed, but it does not have all units committed
Fulfilling Minimum Stored Energy	vm	the water necessary to fulfill the new hydro plants' reservoir up to its minimum volume
Energy Controllable Inflow Deviation	qc	the water deviated just before the reservoirs
Energy Run-of-river Inflow Deviation	qf	the water deviated just before the run-of-river plants
Maximum Energy Generation	gh^{max}	the reservoirs' maximum water discharge.
Spillage Energy	s	the water spilled
Stored Energy Correction	fv	Correction of the stored energy due to new reservoirs
Energy Inflow Correction	fc	Correction of the energy inflow depending on the stored energy

We summarize in Table 2 the roles of the components of the formulation above, i.e., if they are decision variables, state variables, random variables, input data depending on the previous node (called internal input data) or external input data.

Table 2 - Roles of the components in the LTHSP formulation.

Description	Components
Decision Variables	gt, d, gh, f, exc, s
State Variables	v
Random Variables	y
Internal input data	$gh^{max}, q^{min}, ev, fc$
External input data	$ct, cd, a, L, LF, FP, gt^{min}, gt^{max}, d^{max}, f^{max}, \beta, v^{max}, v^{min}, pu, qc, qf, fv, vm$

[1] CEPEL. Manual de Referência do Modelo NEWAVE. Rio de Janeiro: CEPEL; 2001.

[2] de Matos VL, Finardi EC, Silva ELd. Comparison between the Energy Equivalent Reservoir per Subsystem and per Cascade in the Long-Term Operation Planning in Brazil. EngOpt - International Conference on Engineering Optimization. Rio de Janeiro2008.

[3] The unit of all attributes is MWmonth, which is the equivalent to a constant generation or consumption through the whole month.

References

1. Bailey, T.G., Jensen, P., Morton, D.: Response surface analysis of two-stage stochastic linear programming with recourse. Nav. Res. Logist. **46**, 753–778 (1999)
2. Bayraksan, G., Morton, D.P.: Assessing solution quality in stochastic programs. Math. Program. **108**, 495–514 (2006)
3. Chen, Z.L., Powell, W.B.: Convergent cutting plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. J. Optim. Theory Appl. **102**, 497–524 (1999)
4. Chiralaksanakul, A., Morton, D.P.: Assessing policy quality in multi-stage stochastic programming. Manuscript, The University of Texas at Austin, available at SPEPS. http://edoc.hu-berlin.de/browsing/speps/ (2004)
5. de Matos, V.L., Finardi, E.C.: A stochastic optimization model for long term hydrothermal scheduling. Manuscript, Laboratório de Planejamento de Sistemas de Energia Elétrica, Federal University of Santa Catarina, Brazil (2010)
6. de Matos, V.L., Finardi, E.C., da Silva, E.L.: Comparison between the energy equivalent reservoir per subsystem and per cascade in the long-term operation planning in Brazil. In: EngOpt 2008-International Conference on Engineering Optimization, Rio de Janeiro (2008)
7. Donohue, C., Birge, J.R.: The abridged nested decomposition method for multistage stochastic programming. Algorithmic Oper. Res. **1**(1), 20–30 (2006)
8. dos Santos, M.L., da Silva, E.L., Finardi, E.C., Goncalves, R.E.: Practical aspects in solving the medium-term operation planning problem of hydrothermal power systems by using the progressive hedging method. Int. J. Electr. Power Energy Syst. **31**(9), 546–552 (2009)
9. Drew, S.S., Homem-de-Mello, T.: Quasi-Monte Carlo strategies for stochastic optimization. In: Perrone, L.F., Wieland, F.P., Liu, J., Lawson, B.G., Nicol, D.M., Fujimoto, R.M. (eds.) Proceedings of the 2006 Winter Simulation Conference, pp. 774–782. IEEE Press, New York (2006)
10. Finardi, E.C., da Silva, E.L.: Solving the hydro unit commitment problem via dual decomposition and sequential quadratic programming. IEEE Trans. Power Syst. **21**(2), 835–844 (2006)
11. Friedel, I., Keller, A.: Fast generation of randomized low-discrepancy point sets. In: Monte Carlo and Quasi-Monte Carlo Methods, 2000, Hong Kong, pp. 257–273. Springer, Berlin (2002). Software available at http://www.multires.caltech.edu/software/libseq/
12. Heitsch, H., Römisch, W.: Scenario tree modeling for multistage stochastic programs. Math. Program. **118**, 371–406 (2009)
13. Hilli, P., Pennanen, T.: Numerical study of discretizations of multistage stochastic programs. Kybernetika **44**(2), 185–204 (2008)
14. Hindsberger, M., Philpott, A.B.: ReSa: A method for solving multi-stage stochastic linear programs. In: SPIX Stochastic Programming Symposium, Berlin (2001)
15. Homem-de-Mello, T.: On rates of convergence for stochastic optimization problems under non-i.i.d. sampling. SIAM J. Optim. **19**(2), 524–551 (2008)
16. Infanger, G., Morton, D.P.: Cut sharing for multistage stochastic linear programs with interstage dependency. Math. Program. **75**, 241–256 (1996)
17. Kalagnanam, J., Diwekar, U.: An efficient sampling technique for off-line quality control. Technometrics **39**(3), 308–319 (1997)
18. Koivu, M.: Variance reduction in sample approximations of stochastic programs. Math. Program. **103**(3), 463–485 (2005)
19. L'Ecuyer, P., Lemieux, C.: Recent advances in randomized quasi-Monte Carlo methods. In: Dror, M., L'Ecuyer, P., Szidarovszky, F. (eds.) Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications, pp. 419–474. Kluwer Academic, Boston (2002)
20. Linderoth, J.T., Shapiro, A., Wright, S.J.: The empirical behavior of sampling methods for stochastic programming. Ann. Oper. Res. **142**(1), 215–241 (2006)
21. Loh, W.: On Latin hypercube sampling. Ann. Stat. **24**(5), 2058–2080 (1996)
22. Mak, W.K., Morton, D.P., Wood, R.K.: Monte Carlo bounding techniques for determining solution quality in stochastic programs. Oper. Res. Lett. **24**, 47–56 (1999)
23. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**, 239–245 (1979)
24. Morton, D.P.: Stopping rules for a class of sampling-based stochastic programming algorithms. Oper. Res. **46**(5), 710–718 (1998)
25. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia (1992)

26. Noakes, D.J., McLeod, A.I., Hipel, K.W.: Forecasting monthly riverflow time series. Int. J. Forecast. **1**(2), 179–190 (1985)
27. Owen, A.B.: Monte Carlo variance of scrambled net quadrature. SIAM J. Numer. Anal. **34**(5), 1884–1910 (1997)
28. Owen, A.B.: Latin supercube sampling for very high-dimensional simulations. ACM Trans. Model. Comput. Simul. **8**, 71–102 (1998)
29. Pennanen, T.: Epi-convergent discretizations of multistage stochastic programs. Math. Oper. Res. **30**, 245–256 (2005)
30. Pennanen, T., Koivu, M.: Epi-convergent discretizations of stochastic programs via integration quadratures. Numer. Math. **100**, 141–163 (2005)
31. Pereira, M.V.F., Pinto, L.M.V.G.: Multi-stage stochastic optimization applied to energy planning. Math. Program. **52**, 359–375 (1991)
32. Philpott, A., Guan, Z.: On the convergence of stochastic dual dynamic programming and related methods. Oper. Res. Lett. **36**, 450–455 (2008)
33. Ruszczyński, A.: Decomposition methods. In: Ruszczyński, A., Shapiro, A. (eds.) Handbook of Stochastic Optimization. Elsevier, Amsterdam (2003)
34. Shapiro, A.: Monte Carlo sampling methods. In: Ruszczyński, A., Shapiro, A. (eds.) Handbook of Stochastic Optimization. Elsevier, Amsterdam (2003a)
35. Shapiro, A.: Inference of statistical bounds for multistage stochastic programming problems. Math. Methods Oper. Res. **58**, 57–68 (2003b)
36. Shapiro, A.: Analysis of stochastic dual dynamic programming method. Eur. J. Oper. Res. **209**, 63–72 (2011)
37. Shapiro, A., Homem-de-Mello, T.: A simulation-based approach to two-stage stochastic programming with recourse. Math. Program. **81**, 301–325 (1998)
38. Shapiro, A., Homem-de-Mello, T., Kim, J.C.: Conditioning of convex piecewise linear stochastic programs. Math. Program. **94**, 1–19 (2002)
39. Spanier, J.: Quasi-Monte Carlo methods for particle transport problems. In: Niederreiter, H., Shiue, P. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, pp. 121–148. Springer, New York (1995)
40. Stein, M.L.: Large sample properties of simulations using Latin hypercube sampling. Technometrics **29**, 143–151 (1987)
41. Wasserman, L.: All of Statistics: A Concise Course in Statistical Inference. Springer, Berlin (2004)