Mathematical Programming 52 (1991) 359-375
North-Holland



# Multi-stage stochastic optimization applied to energy planning

M.V.F. Pereira and L.M.V.G. Pinto

*Electric Engineering Department, Catholic University of Rio de Janeiro, P.O. Box 38063, Gavea, 22452 Rio de Janeiro, RJ, Brazil*

Received October 1988
Revised manuscript received December 1989

This paper presents a methodology for the solution of multistage stochastic optimization problems, based on the approximation of the expected-cost-to-go functions of stochastic dynamic programming by piecewise linear functions. No state discretization is necessary, and the combinatorial "explosion" with the number of states (the well known "curse of dimensionality" of dynamic programming) is avoided. The piecewise functions are obtained from the dual solutions of the optimization problem at each stage and correspond to Benders cuts in a stochastic, multistage decomposition framework. A case study of optimal stochastic scheduling for a 39-reservoir system is presented and discussed.

## 1. Introduction

This technical note describes an algorithm for the solution of multistage stochastic optimization problems. The solution approach, called *stochastic dual dynamic programming* (SDDP), is based on the approximation of the expected-cost-to-go functions of stochastic dynamic programming by piecewise linear functions. These approximate functions are obtained from the dual solutions of the optimization problem at each stage and can be interpreted as Benders cuts in a stochastic, multistage decomposition algorithm. No state discretization is necessary, and the combinatorial "explosion" with the number of states (the well known "curse of dimensionality" of dynamic programming) is avoided. The algorithm is also suitable for implementation in parallel processors. The application of the algorithm is illustrated in a case study of optimal stochastic scheduling for a 39-reservoir system.

## 2. Dual dynamic programming — deterministic case

The concepts of dual dynamic programming will be illustrated with the following linear programming problem:

$$\begin{aligned} &\text{Min} && c_1x_1 + c_2x_2 \\ &\text{subject to} && A_1x_1 \geqslant b_1, \\ & && E_1x_1 + A_2x_2 \geqslant b_2. \end{aligned} \tag{1}$$

Problem (1) can be interpreted as a *two-stage decision process.* In the first stage, we decide on a trial feasible value for $x_1$ (i.e. such that $A_1 x_1 \geq b_1$). Given the trial value, $\hat{x}_1$, we find the optimal solution of the second stage function:

$$\begin{aligned} &\text{Min} && c_2 x_2 \\ &\text{subject to} && A_2 x_2 \geq b_2 - E_1 \hat{x}_1 . \end{aligned} \tag{2}$$

Note that $\hat{x}_1$ is a known value in the second-stage problem (2), and goes to the right-hand side of the constraints. The objective is to minimize the sum of the first-stage and second-stage cost functions. Figure 1 illustrates the decision process.

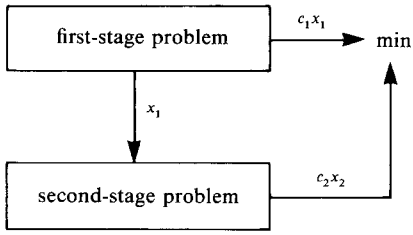


Fig. 1. Two-stage decision process.

Dynamic programming (DP) algorithms can be used to solve sequential decision processes such as problem (1). In the DP approach, a first-stage problem would be defined as

$$\begin{aligned} &\text{Min} && c_1 x_1 + \alpha_1(x_1) \\ &\text{subject to} && A_1 x_1 \geq b_1 . \end{aligned} \tag{3}$$

In DP terminology, $c_1 x_1$ represents the "immediate cost" and $\alpha_1(x_1)$ represents the "future cost" of decision $x_1$, i.e. the consequences of this decision for the second-stage problem. The future cost function $\alpha_1(x_1)$ is defined as

$$\begin{aligned} \alpha_1(x_1) \quad = \quad &\text{Min} && c_2 x_2 \\ &\text{subject to} && A_2 x_2 \geq b_2 - E_1 x_1 . \end{aligned} \tag{4}$$

The future cost function (4) "translates" the second-stage costs as a function of the first-stage decisions $x_1$, also called *state variables.* If this function is available, the two-stage problem (1) can be solved as the one-stage problem (3).

The DP algorithm constructs the future cost function $\alpha_1(x_1)$ by *discretizing* $x_1$ into a set of trial values $\{\hat{x}_{1i},\ i = 1, \ldots, n\}$ and solving problem (4) for each of the trial values. Intermediate values of $\alpha_1(x_1)$ are obtained by interpolation from the neighboring discretized states. Once this function is constructed, the first-stage problem (3) can be solved. Figure 2 illustrates the calculation of the future cost function $\alpha_1(x_1)$ for a set of discretized values.

The DP approach has many attractive features: it can be easily extended to multistage problems; can be extended to stochastic cases; can accommodate discrete

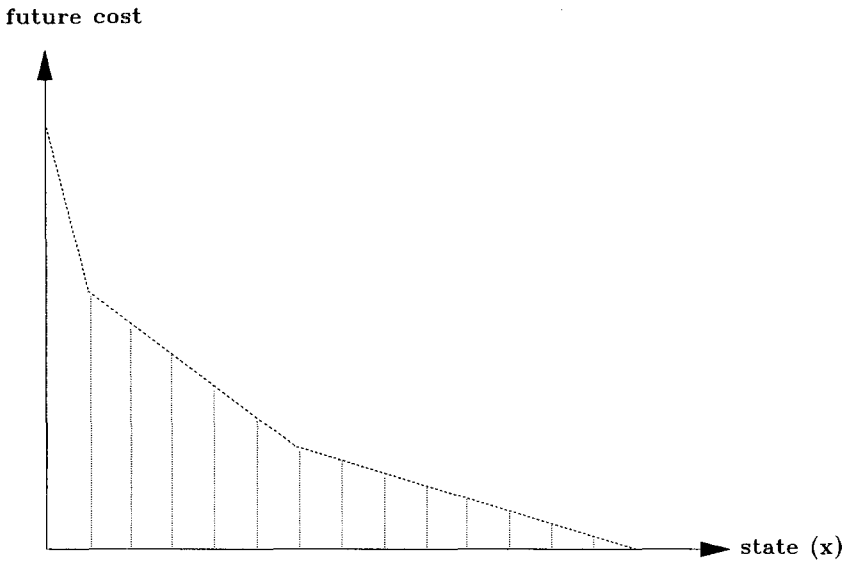


Fig. 2. State discretization in DP.

values, nonlinearities etc. Its main drawback is the need to discretize the decisions (state variables) in order to construct the future cost functions. It is easy to see that this may lead to a very large number of combinations even for a modest number of variables. For example, if the vector $x_1$ has ten components, and if each component is discretized into four values, there will be $4^{10}$, i.e. over one million possible discrete values of $x_1$. This "curse of dimensionality" limits the applicability of DP algorithms to problems with a small number of variables (typically three or four, in the case of stochastic DP problems).

One possible way of avoiding the dimensionality problem is to approximate the future cost function by *analytical* functions rather than a set of discrete values. For example, one could calculate the future cost values $\alpha_1(x_1)$ for a *sample* of states, and then adjust a polynomial (e.g. a quadratic or cubic function of $x_1$) to these values. The polynomial would then be used in the previous stage to supply future cost values for any trial decision $x_1$ (see, for example, Gal, 1989). The algorithm proposed in this note is based on a similar idea. We will show that the future cost function can be represented *exactly* as a *piecewise linear function*, and use a relaxation of this piecewise function as our approximation.

The structure of the future cost function can be characterized by taking the dual of the second-stage problem (4):

$$\begin{aligned} \alpha_1(x_1) \quad = \quad & \text{Max} \qquad \pi(b_2 - E_1 x_1) \\ & \text{subject to} \quad \pi A_2 \leqslant c_2 \end{aligned} \tag{5}$$

where $\pi$ is the (row) vector of dual variables. It is known from LP theory that the optimal solutions of the dual problem (5) and of the original problem (4) coincide.

Therefore, both (4) and (5) represent the future cost function $\alpha_1(x_1)$. Note, however, that the decision variable $x_1$ is in the objective function of (5), and not in the right-hand side of the constraint set as in the original problem (4). This means that the set of possible solutions to problem (5), which correspond to the vertices of the constraint set $\pi A_2 \leq c_2$, can be characterized before knowing the decision $x_1$.

Let $\Pi = \{\pi^1, \pi^2, \ldots, \pi^v\}$ represent the set of all vertices of the constraint set. Because the optimal solution belongs to this set, problem (5) can in principle be solved by *enumeration*:

$$\alpha_1(x_1) = \text{Max}\{\pi^i(b_2 - E_1 x_1) \text{ for all } i = 1, \ldots, v\}. \tag{6}$$

Problem (6) can also be rewritten as a linear programming problem:

$$\begin{aligned} \alpha_1(x_1) = \text{Min} \quad & \alpha \\ \text{subject to} \quad & \alpha \geq \pi^1(b_2 - E_1 x_1), \\ & \vdots \\ & \alpha \geq \pi^v(b_2 - E_1 x_1), \end{aligned} \tag{7}$$

where $\alpha$ is a scalar variable. The equivalence of (7) and (6) can be easily established by observing that $\alpha \geq \pi^i(b_2 - E_1 x_1)$ for all $i = 1, \ldots, v$ in problem (7) implies that $\alpha \geq \text{Max}\{\pi^i(b_2 - E_1 x_1)\}$. Because the objective is to minimize $\alpha$, we can conclude that the constraint will be met in the equality as required in problem (6).

Problem (7) has an interesting geometrical interpretation. It indicates that the future cost function $\alpha_1(x_1)$ is a *piecewise linear function* of the decision variable $x_1$, as illustrated in Figure 3. The components of this piecewise function are the

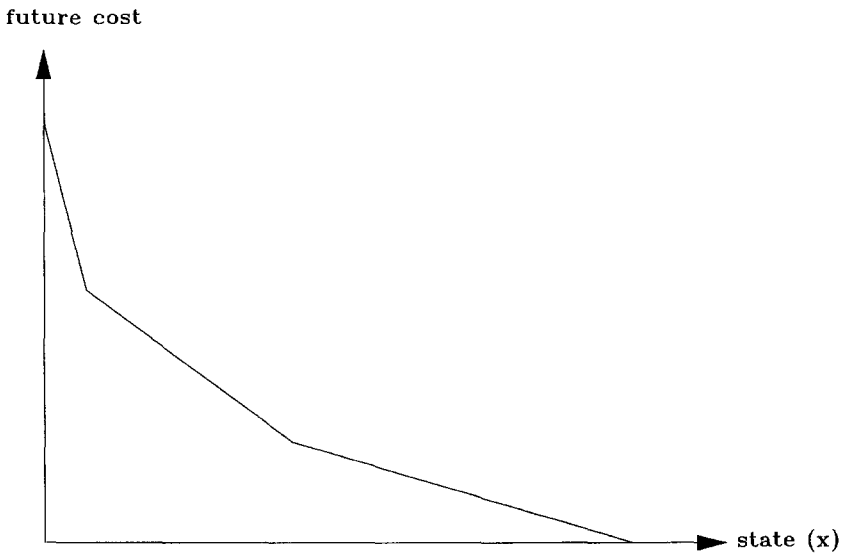


Fig. 3. Piecewise linear approximation.

hyperplanes defined by each $\pi^i(b_2 - E_1 x_1)$. This implies that the future cost function can be characterized without discretizing the values $x_1$; it is sufficient to know the coefficients $\{\pi^i\}$ of the supporting hyperplanes.

Naturally, the calculation of all vertices $\{\pi^i\}$ in set $\Pi$ may be a very difficult task. Our basic approach will be to calculate a *subset* of these vertices and to construct an approximation to the future cost function. We will initially show that these vertices can be calculated as dual variables of the second-stage problem (4), reproduced below:

$$\begin{array}{rcll} \alpha_1(\hat{x}_{1i}) & = & \text{Min} & c_2 x_2 & \text{dual} \\ & & \text{subject to} & A_2 x_2 \geq b_2 - E_1 \hat{x}_{1i} & \text{var } \pi^i, \end{array} \tag{8}$$

where $\hat{x}_{1i}$ is a trial value. Let $\pi^i$ be the *simplex multiplier vector* associated to the constraints of problem (8). It is known from LP theory that this vector is one of the vertices of the solution set $\Pi$ in the dual problem. Therefore, it can be used to construct one of the supporting hyperplanes of the future cost function $\alpha_1(x_1)$.

In other words, given a set of $n$ trial decisions $\{x_{1i}, i = 1, \ldots, n\}$, we can calculate the set of associated multipliers $\{\pi^i, i = 1, \ldots, n\}$ by solving problem (8) for each of the trial values. An *approximation* to the future cost function can then be constructed as:

$$\begin{array}{rcll} \hat{\alpha}_1(x_1) & = & \text{Min} & \alpha \\ & & \text{subject to} & \alpha \geq \pi^i(b_2 - E_1 x_1) \text{ for } i = 1, \ldots, n. \end{array} \tag{9}$$

It is easy to see that the approximate function $\hat{\alpha}_1(x_1)$ is a *lower bound* to the future cost function $\alpha_1(x_1)$, because problem (9) has only a subset of the constraints of problem (7). The approximate future cost function can then be used to solve the first-stage problem as in the DP formulation:

$$\begin{array}{rcll} z & = & \text{Min} & c_1 x_1 + \hat{\alpha}_1(x_1) \\ & & \text{subject to} & A_1 x_1 \geq b_1. \end{array} \tag{10}$$

Note that problem (10) is actually a LP problem. Substituting (9) into (10), we get:

$$\begin{array}{rcll} z & = & \text{Min} & c_1 x_1 + \alpha \\ & & \text{subject to} & A_1 x_1 \geq b_1, \\ & & & \alpha - \pi^i(b_2 - E_1 x_1) \geq 0 \text{ for } i = 1, \ldots, n. \end{array} \tag{11}$$

Because $\hat{\alpha}_1(x_1)$ is an approximation to the future cost function, we cannot guarantee that the solution of problem (10) is the optimal solution of the two-stage problem (1). However, because $\hat{\alpha}_1(x_1)$ is a lower bound to the future cost function, we know that the optimal solution value of (10) is a *lower bound* $\underline{z}$ to the true optimal cost. In other words,

$$\underline{z} = c_1 \hat{x}_1 + \hat{\alpha} \tag{12}$$

where $\hat{x}_1$ and $\hat{\alpha}$ are the optimal solutions of the approximate problem (10). In turn, an *upper bound* $\bar{z}$ can be obtained by solving the second-stage problem (4) for $\hat{x}_1$:

$$\bar{z} = c_1\hat{x}_1 + \alpha_1(\hat{x}_1). \tag{13}$$

The difference between upper and lower bounds $\bar{z} - \underline{z}$ can be used to verify the accuracy of the approximate future cost function. Initially, we note that the term $c_1\hat{x}_1$ is cancelled, because it belongs to both upper and lower bound expressions. Thus, we can see that $\bar{z} - \underline{z}$ measures the difference between the *predicted* future cost (given by $\hat{\alpha}$) and the *actual* future cost (given by $\alpha_1(\hat{x}_1)$) of the current optimal solution $\hat{x}_1$. If this difference is greater than a given tolerance, the problem is solved. Otherwise, a new set of trial decisions must be used to determine additional vertices $\pi^i$. The process described so far can be summarized in the following algorithm:

*Step (a).* Select a set of $n$ trial decisions $\{\hat{x}_{1i},\ i = 1, \ldots, n\}$.

*Step (b).* For each trial decision, solve the second-stage problem, and calculate the associated multipliers $\pi^i$ as in (8).

*Step (c).* Use the multipliers $\{\pi^i\}$ to construct the approximate future cost function as in (9); solve the approximate first-stage problem (10).

*Step (d).* Calculate upper and lower bounds as in (12) and (13); if $\bar{z} - \underline{z}$ are within a given tolerance $\varepsilon$, stop; otherwise, go to Step (a).

One important point remains to be discussed, which is the selection of trial decisions $\{\hat{x}_{1i}\}$ in Step (a). At each iteration of the algorithm, we will use as an additional trial decision the optimal solution $\hat{x}_1$ of Step (b) (the approximate first stage problem) at the *previous* iteration. This ensures that we will be building approximations to the future cost function in the neighborhood of "interesting" points, i.e. points which are good candidates to the optimal solution.

The two-stage *dual dynamic programming* (DDP) algorithm is then composed of the following steps:

*Step (a).* Initialize: approximate future cost function $\hat{\alpha}_1(x_1) = 0$; upper bound $\bar{z} = \infty$; number of vertices $n = 0$.

*Step (b).* Solve the approximate first stage problem (10); let $\hat{x}_1$ be the optimal solution.

*Step (c).* Calculate the lower bound $\underline{z}$ as in (12); if $\bar{z} - \underline{z} \leqslant \varepsilon$, stop; otherwise, go to Step (d).

*Step (d).* Solve the second stage problem (8), i.e. calculate $\alpha(\hat{x}_1)$; update $\bar{z}$ as in (13).

*Step (e).* Increment the number of vertices $n \leftarrow n + 1$; let the multiplier associated to the optimal solution of Step (d) be $\pi^n$; construct the approximate cost function $\hat{\alpha}_1(x_1)$ as in (9), using the $n$ vertices.

*Step (f).* Go to Step (b).

The DDP algorithm described above has several attractive features: no state discretization is required; upper and lower bounds are provided at each iteration; the previous optimal solution of the approximate optimization problem (10) of step (b) can be used as an initial solution in the next iteration (note that the only difference between two successive problems is one additional linear constraint associated to the new vertex $\pi^n$).

It should be observed that the two-stage version of the DDP algorithm outlined above is equivalent to a Benders decomposition algorithm (Benders, 1962), in which both master and subproblem are LP problems. The extensions of the DDP algorithm to the multistage and stochastic cases can also be interpreted as extensions of the Benders decomposition scheme (see, for example, Birge, 1980; Wets, 1988).

The multistage DDP algorithm is composed of the following steps:

*Step (a).* Let $T$ be the planning horizon; initialize $\hat{\alpha}_t(x_t) = 0$ for $t = 1, \ldots, T$; $\bar{z} = \infty$.

*Step (b).* Solve the approximate first stage problem (10); let $\hat{x}_1$ be the optimal solution.

*Step (c).* Calculate the lower bound $\underline{z}$ as in (12); if $\bar{z} - \underline{z} \leq \varepsilon$, stop; otherwise, go to Step (d).

*Step (d).* Repeat for $t = 2, \ldots, T$ (*forward simulation*).

Solve the optimization problem for stage $t$, trial decision $\hat{x}_{t-1}$:

$$\begin{aligned} &\text{Min} && c_t x_t + \hat{\alpha}_t(x_t) \\ &\text{subject to} && A_t x_t \geq b_t - E_{t-1}\hat{x}_{t-1}. \end{aligned} \tag{15}$$

Store the optimal solution as $\hat{x}_t$.

*Step (e).* Calculate the upper bound

$$\bar{z} = \sum_{t=1}^{T} c_t \hat{x}_t. \tag{16}$$

*Step (f).* Repeat for $t = T, T-1, \ldots, 2$ (*backward recursion*).

Solve the optimization problem for stage $t$, trial decision $\hat{x}_{t-1}$:

$$\begin{aligned} &\text{Min} && c_t x_t + \hat{\alpha}_t(x_t) \\ &\text{subject to} && A_t x_t \geq b_t - E_{t-1}\hat{x}_{t-1}. \end{aligned} \tag{17}$$

Let $\pi_{t-1}$ be the multiplier associated to the constraints of problem (17) at the optimal solution; use this multiplier to construct an additional supporting hyperplane for the approximate future cost function in the *previous* stage, $\hat{\alpha}_{t-1}(x_{t-1})$.

*Step (g).* Go to Step (b).

## 3. Stochastic dual dynamic programming

As mentioned previously, one attractive feature of DP algorithms is their capability of handling stochastic problems. The DDP approach described in the previous section can also be extended to the stochastic case. This will be illustrated with the following two-stage problem:

$$\begin{aligned} \text{Min} \quad & c_1x_1 + p_1c_2x_{21} + p_2c_2x_{22} + \cdots + p_mc_2x_{2m} \\ \text{subject to} \quad & A_1x_1 \geqslant b_1, \\ & E_1x_1 + A_2x_{21} \geqslant b_{21}, \\ & E_1x_1 \qquad + A_2x_{22} \geqslant b_{22}, \\ & \qquad\qquad\qquad\qquad \ddots \\ & E_1x_1 \qquad\qquad\qquad\quad + A_2x_{2m} \geqslant b_{2m}. \end{aligned} \tag{18}$$

Problem (18) can be interpreted as follows: in the first stage, a decision $x_1$ is taken; given the trial decision $x_1$, there will be $m$ second stage subproblems:

$$\begin{aligned} \alpha_{1j}(x_1) \quad = \quad & \text{Min} \qquad c_2x_{2j} \\ & \text{subject to} \quad A_2x_{2j} \leqslant b_{2j} - E_1x_1 \end{aligned} \tag{19}$$

for all $j = 1, \ldots, m$. The objective is to minimize the sum of the first-stage costs $c_1x_1$ plus the expected value of the second-stage costs ($\sum_{j=1}^{m} p_jc_2x_{2j}$), where $p_j$ represents the probability of each subproblem (naturally, $\sum_{j=1}^{m} p_j = 1$). Figure 4 illustrates the decision process.

As in the deterministic case of Section 2, the two-stage problem (18) can in principle be solved by a stochastic DP recursion. Given a trial decision $x_1$, one can build the expected future cost function $\bar{\alpha}_1(x_1)$ as

$$\bar{\alpha}_1(x_1) = \sum_{j=1}^{m} p_j\alpha_{1j}(x_1) \tag{20}$$

where $\alpha_{1j}(x_1)$ is defined in (19).

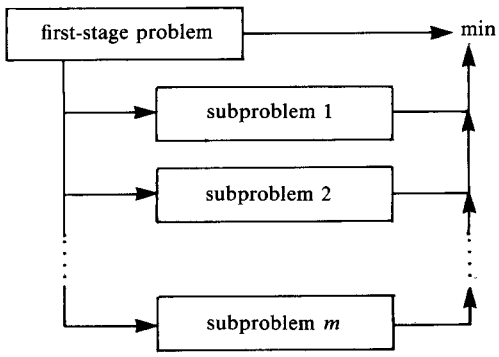


Fig. 4. Two-stage decision process (stochastic case).

The first-stage problem in the DP recursion becomes

$$\begin{aligned} &\text{Min} && c_1 x_1 + \bar{\alpha}_1(x_1) \\ &\text{subject to} && A_1 x_1 \geq b_1 \end{aligned} \tag{21}$$

where $c_1 x_1$ represents the immediate cost, and $\bar{\alpha}_1(x_1)$ represents the future (expected) consequences of decision $x_1$. The derivations leading to the dual DP algorithm are similar to the deterministic case (the future cost function is piecewise etc.).

In order to present the multistage stochastic DDP algorithm we assume, without loss of generality, that the right-hand side vectors $\{b_t,\ t=1,\ldots,T\}$ are independent random variables, and that each $b_t$ is discretized into $m$ values, or *scenarios* $\{b_{tj}, j=1,\ldots,m\}$ with probabilities $\{p_{tj}, j=1,\ldots,m\}$. The algorithm is implemented as follows:

*Step (a).* Define a set of trial decisions $\{\hat{x}_{ti}$ for $i=1,\ldots,n,\ t=1,\ldots,T\}$.

*Step (b).* Repeat for $t=T,\ T-1,\ldots,2$ (backward recursion).

Repeat for each trial decision $\hat{x}_{ti},\ i=1,\ldots,n$.

Repeat for each scenario $b_{tj},\ j=1,\ldots,m$.

Solve the optimization problem for $t$, $\hat{x}_{t-1i}$, and $b_{tj}$:

$$\begin{aligned} &\text{Min} && c_t x_t + \bar{\alpha}_t(x_t) \\ &\text{subject to} && A_t x_t \geq b_{tj} - E_{t-1}\hat{x}_{t-1i}. \end{aligned} \tag{22}$$

Let $\pi_{t-1ij}$ be the multiplier associated to the constraints of problem (22) at the optimal solution.

Calculate the expected vertex value $\bar{\pi}_{t-1i} = \sum_{j=1}^{m} p_{tj}\pi_{t-1ij}$, and construct one supporting hyperplane of the approximate expected future cost function for stage $t-1$, $\bar{\alpha}_{t-1}(x_{t-1})$.

*Step (c).* Go to Step (a).

As in the deterministic case, one important aspect in the algorithm is the determination of the trial decisions $\{\hat{x}_{ti}\}$. Ideally, we should carry out a forward simulation (see Step (d) of the multistage deterministic DDP algorithm) for every combination of scenarios $\{b_{tj}\}$. Note, however, that the combinations increase exponentially with the number of stages. Our proposal is to carry out a *Monte Carlo* forward simulation for a *sample* of the scenarios as follows:

*Step (a).* Solve the first-stage problem (21); let $\hat{x}_1$ be the optimal solution; initialize $x_{1i} = \hat{x}_1$ for $i=1,\ldots,n$.

*Step (b).* Repeat for $t = 2, \ldots, T$.
Repeat for $i = 1, \ldots, n$.
*Sample* a vector $b_{ti}$ from the set $\{b_{tj}, j = 1, \ldots, m\}$.
Solve the optimization problem for stage $t$, sample $i$:

$$\begin{aligned} &\text{Min} && c_t x_t + \bar{\alpha}_t(x_t) \\ &\text{subject to} && A_t x_t \geq b_{ti} - E_{t-1}\hat{x}_{t-1i}. \end{aligned} \tag{23}$$

Store the optimal solution as $\hat{x}_{ti}$.

As in the deterministic case, the objective of the above simulation is to determine "good" trial decisions at each stage, i.e. around which one should try to approximate the future cost function.

One aspect remains to be discussed, which is the calculation of lower and upper bounds. It can be seen that the lower bound $\underline{z}$ is obtained from the solution of the first stage problem (21), as in the deterministic case. The upper bound $\bar{z}$, in turn, is *estimated* from the Monte Carlo simulation results for all stages and scenarios, that is,

$$\bar{z} = c_1 x_1 + \frac{1}{n} \sum_{i=1}^{n} z_i \tag{24}$$

where $z_i$ is the total cost for one Monte Carlo run:

$$z_i = \sum_{t=1}^{T} c_t \hat{x}_{ti}. \tag{25}$$

The *uncertainty* around the estimate of $\bar{z}$ in expression (24) can be measured by the standard deviation of the estimator:

$$\sigma_z = \sqrt{\frac{1}{n^2} \sum_{i=1}^{n} (\bar{z} - z_i)^2}. \tag{26}$$

For example, the 95% confidence interval for the "true" (population) value of $\bar{z}$ is given by

$$[\bar{z} - 2\sigma_z, \bar{z} + 2\sigma_z]. \tag{27}$$

It is interesting to observe that the uncertainty around the upper bound estimate can be used as a convergence criterion: for example, if the lower bound $\underline{z}$ is in the interval (27), the algorithm is stopped. This criterion introduces a relationship between the acceptable accuracy of the simulation (given by the sample size $n$) and the accuracy of the optimal policy calculated by the stochastic DDP algorithm.

## 4. Computational aspects and other extensions of the algorithm

### *4.1. Solution strategies*

The solution strategy presented in the previous sections (forward simulation of the sequential decisions process; backward recursion to update the future cost functions)

allows many variations. For example, it is possible to iterate between the forward and backward steps in two or more stages without necessarily going back to the first stage.

### *4.2. Parallel processing*

The forward simulation of the sequential decisions process (23) can be carried out independently for each sample. It is also possible to calculate the backward recursions (22) as a set of independent problems. This allows the implementation of the algorithm in computers with several processors operating concurrently. We conjecture that the convergence process itself can be asynchronous (some processors involved in the forward simulation; others calculating the backward recursion results in various stages etc.).

### *4.3. Nonlinear functions*

The algorithm can be extended to the nonlinear case. Natually, convergence to the global optimum can only be ensured under the usual convexity conditions for nonlinear problems.

## 5. Applications

The application of the algorithm will be illustrated for the problem of optimal scheduling of a hydrothermal generating system.

### *5.1. Introduction*

The objective of the optimal operation of a hydrothermal system is to determine an operation strategy which, for each stage of the planning period, given the system state, produces generation targets for each plant. This strategy should minimize the expected value of the operation cost along the period, composed of fuel costs plus penalties for failure in load supply. The availability of limited amounts of hydroelectric energy, in the form of stored water in the system reservoirs, makes the optimal operation problem very complex, because it creates a *link* between an operating decision in a given stage and the future consequences of this decision. In other words, if we deplete the stocks of hydroelectric energy, and low inflow volumes occur, it may be necessary to use very expensive thermal generation in the future or even fail to supply the load. On the other hand, if we keep the reservoir levels high through a more intensive use of thermal generation and high inflow volumes occur, there may be a spillage in the system, which means a waste of energy and, consequently, higher operating costs.

Because it is impossible to have perfect forecasts of the future inflow sequences, the operation problem is essentially *stochastic*. The existence of multiple interconnected reservoirs and the need for multiperiod optimization characterize the problem as *large scale*. The objective function is *nonlinear*, due not only to nonlinear thermal

costs, but also to the product of outflow and head in the expression of hydroelectric production. Finally, because the worth of energy generated in a hydro plant cannot be measured directly as a function of the plant state alone, but rather in terms of expected fuel savings from avoided thermal generation, the objective function is also *nonseparable.*

### 5.2. Problem formulation (Pereira and Pinto, 1985)

If the inflow volumes are known at the beginning of each stage, the operation planning problem can be represented as a stochastic dynamic programming (SDP) recursion:

$$\alpha_t(x_t) = \underset{a_t|x_t}{E}\left\{\underset{u_t}{\text{Min}}\, c_t(u_t) + \beta\alpha_{t+1}(x_{t+1})\right\} \tag{28a}$$

subject to

$$x_{t+1} = f_t(x_t, a_t, u_t), \tag{28b}$$

$$g_{t+1}(x_{t+1}) \geqslant 0, \tag{28c}$$

$$h_t(u_t) \geqslant 0, \tag{28d}$$

$\forall t = T, T-1, \ldots, 1, \forall x_t$, where:

$t$: indexes the stages ($T$ planning horizon),
$x_t$: state vector at the beginning of stage $t$,
$\alpha_t(x_t)$: expected value of operation cost from state $x_t$,
$a_t|x_t$: probability distribution of inflow vector $a_t$ conditioned by state $x_t$,
$E\{\cdot\}$: expected value,
$u_t$: decision vector for the stage,
$c_t(\cdot)$: immediate cost associated to decision $u_t$,
$\beta$: discount factor,
(28b): state transition equation,
(28c): constraints on the state vector,
(28d): constraints on the decision vector.

### 5.3. State variables

The system state $x_t$ should represent all information which can affect future operation costs. In the case of hydroelectric systems, at least two classes of state variables should be included: the reservoir storage level $v_t$ and information about the hydrologic trend in the system. This information can be given, for example, by the lateral inflow volumes during the previous stages. Therefore,

$$x_t = [v_t, a_{t-1}, a_{t-2}, \ldots]. \tag{29}$$

### 5.4. Decision variables

The decision vector $u_t$ comprises the turbined outflow volumes $q_t$ and the spilled volumes $s_t$. In vector terms,

$$u_t = [q_t, s_t]. \tag{30}$$

### 5.5. Immediate costs

The immediate operation cost associated to decision vector $u_t = [q_t, s_t]$ in (28a) is given by the thermal generation cost required to complement the energy supply. Penalties for failure in load supply are represented as dummy thermal plants.

$$c_t(q_t, s_t) = \text{Min} \sum_{k \in K} \sum_{j \in T_k} c_j(g_{tj}) \tag{31a}$$

subject to

$$\sum_{i \in H_k} \rho_i q_{ti} + \sum_{j \in T_k} g_{tj} + \sum_{l \in \Omega_k} (f_{tlk} - f_{tkl}) = d_{tk} \quad \forall k \in K, \tag{31b}$$

$$g_t \leq \bar{g}, \tag{31c}$$

$$|f_t| \leq \bar{f}, \tag{31d}$$

where:

$k$: indexes subsystems, or regions ($K$ is the set of subsystems),
$j$: indexes thermal plants ($T_k$ is the set of thermal plants in subsystem $k$),
$i$: indexes hydro plants ($H_k$ is the set of plants in subsystem $k$),
$g_{tj}$: generation of thermal plant $j$ in stage $t$,
$c_j$: generation cost function of plant $j$,
$\rho_i$: production coefficient of plant $i$,
$d_{tk}$: energy demand in subsystem $k$,
$\bar{g}$: vector of thermal generation capacities,
$f_{tkl}$: energy interchange from system $k$ to system $l$,
$\bar{f}$: vector of interchange capacities,
$\Omega_k$: set of subsystems directly connected to system $k$.

It should be noted that the production coefficient of a hydro plant is in fact a function of the initial volume, end volume and outflow:

$$\rho_i = \rho_i(v_{ti}, v_{t+1i}, q_{ti}, s_{ti}). \tag{32}$$

For ease of presentation, $\rho$ will be assumed to be constant in the derivations that follow.

### 5.6. Transition equation

The transition equation (28b) corresponds to the reservoir water balance equations:

$$v_{t+1} = v_t + a_t + M(q_t + s_t) \tag{33}$$

where $M$ is the incidence matrix of hydro plants.

### 5.7. Constraints on the state vector and decision variables

The constraints on state vector (28c) correspond to bounds on storage:

$$v_{t+1} \leq \bar{v} \tag{34}$$

where $\bar{v}$ is the vector of reservoir storage capacities.

The constraints on decision variables (28d) usually correspond to limits on turbined outflow and lower bounds on total outflow:

$$q_t \leq \bar{q}, \tag{35}$$

$$q_t + s_t \geq \underline{q}, \tag{36}$$

where $\bar{q}$ and $\underline{q}$ represent respectively the upper and lower bounds on outflow.

### 5.8. *The curse of dimensionality*

As discussed previously, the solution of SDP recursions usually requires the discretization of the state space, which leads to an exponential increase of the computational effort with the number of state variables. As an illustration, let $x_t = [v_t, a_{t-1}]$ be the state vector for a system with $n$ reservoirs, in which the hydrologic trend is represented by the lateral inflow in the previous month. If each component of the state vector is discretized into $m$ intervals, there will be $m^{2n}$ discretized states in each stage. Supposing $m = 20$ intervals of discretization:

1 reservoir $\Rightarrow$ $20^2 = 400$ states,
2 reservoirs $\Rightarrow$ $20^4 = 160$ thousand states,
3 reservoirs $\Rightarrow$ $20^6 = 64$ million states,
4 reservoirs $\Rightarrow$ $20^8 = 25$ billion states,
5 reservoirs $\Rightarrow$ $20^{10} = 10$ trillion states.

Therefore, it becomes necessary to develop methods able to approximate the optimal operating policy at reasonable computational cost. The application of stochastic dual dynamic programming (SDDP) to the scheduling problem will be discussed next.

### 5.9. *Application of the SDDP algorithm*

The SDDP algorithm described in Section 3 was applied to a system composed of 39 hydroelectric plants (22 with reservoirs and 17 run-of-the-river), derived from the southern-southeastern Brazilian power pool. The system characteristics can be found in Pereira and Pinto (1985). The initial stored volumes in the system reservoirs were set as 50% of the storage capacities. The other generation resource is an aggregate thermal unit, with maximum generation capacity of 10 000 MW. Thermal generation cost was set at a reference value of \$1/MW.month (i.e. the energy corresponding to a continuous generation of 1 MW along one month). The cost of load curtailment was set at \$10/MW.month. The energy demand for each stage was 29 300 MW.month. The number of stages is 10.

Inflows were represented as independent random vectors, with two realizations per stage. The inflow vector in the first stage is assumed to be known. There are therefore $2^9 = 512$ possible inflow sequences.

Each one-stage subproblem has 102 variables and 40 constraints (excluding upper and lower bounds on variables, and the linear constraints representing the future

cost function). Because there are 512 inflow sequences and 10 stages, the total number of one-stage subproblems would be 5120. The complete problem would then have 522 240 variables and 204 800 constraints. By avoiding duplications, we can reduce the total number of subproblems to 1023, corresponding to a complete problem with 104 346 variables / 40 920 constraints.

The one-stage subproblem (22) corresponds to a linear programming problem. Instead of a standard LP solution package, we have used a customized algorithm that takes advantage of the *network flow* characteristics of the hydrothermal scheduling problem (Kennington and Helgason, 1984). Looking at equations (33)-(36) (flow conservation, bounds on storage and bounds on outflow) we see that can be modelled as flows in a network, which allows the use of efficient solution techniques. The non-network-flow equations (power balance and the linear constraints corresponding to the future cost functions) are modelled as additional constraints to the network problem. The problem is then solved by a basis partitioning algorithm (Kennington and Helgason, 1984).

As shown in Section 3, we use a *sample* of the set of possible inflow sequences to calculate the new trial states $\{\hat{x}_{ti}\}$ at each iteration (forward simulation step). The forward simulation is also used to obtain an upper bound to the optimal cost (see equation (20)). Finally, the uncertainty around the upper bound estimate, given by the standard deviation of the estimator (see equation (26)) is used as a convergence criterion. In this case study, a sample of 50 inflow sequences was used.

The optimal solution in this case was obtained in five iterations (one iteration consists of a forward simulation and a backward recursion). Figure 5 illustrates the

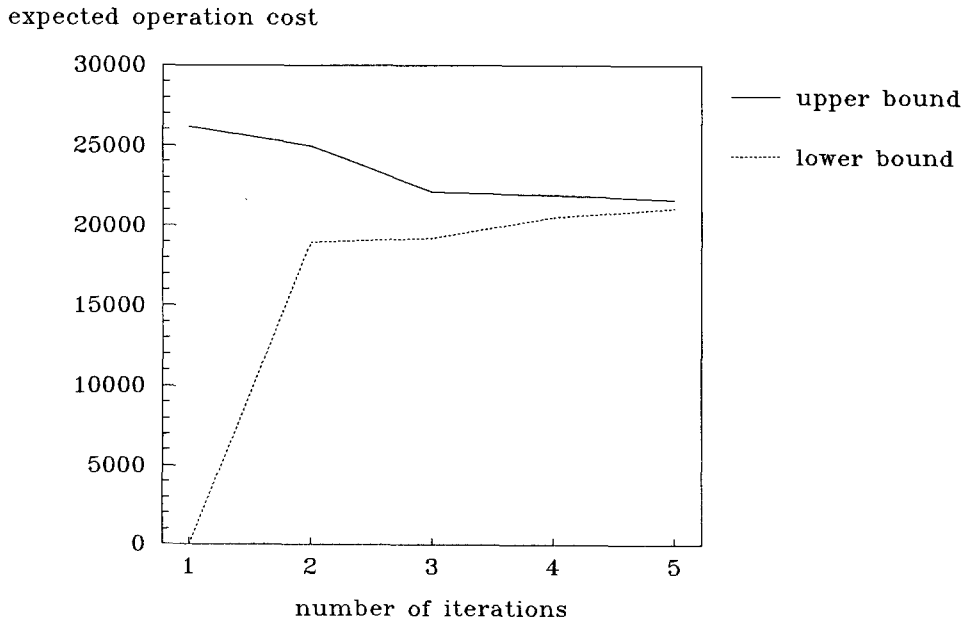


Fig. 5. Convergence of SDDP algorithm.

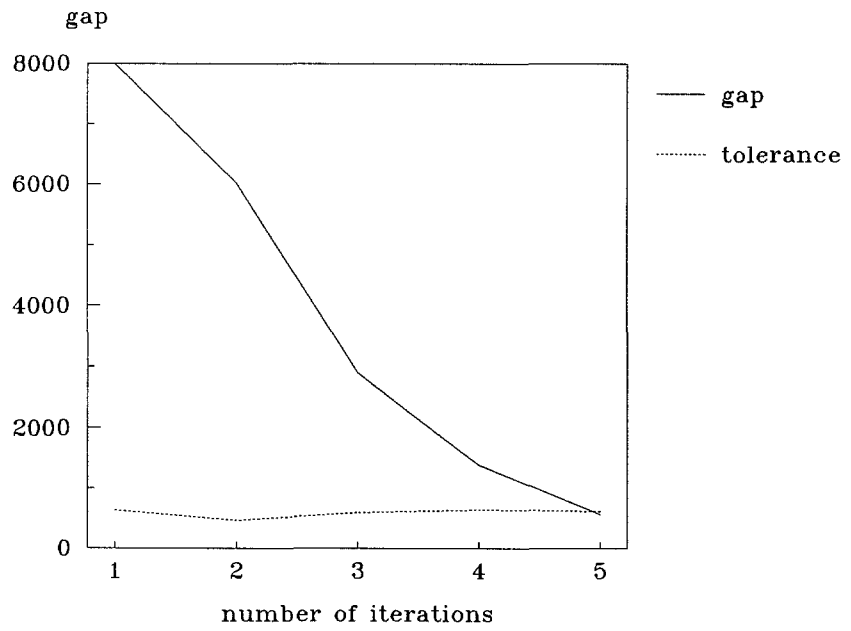


Fig. 6. Convergence criterion.

evolution of lower and upper bounds. Figure 6 shows the evolution of the gap (difference between upper and lower bound) and the standard deviation of the upper bound estimate, which served as a convergence criterion. Total CPU time was 37 minutes in a VAX 11/780 (a 1 MIP computer, comparable to a 20 MHz 386-based personal computer with floating point coprocessor). This is roughly equivalent to 1.5 CPU minutes on an IBM 3090 (25 MIPS).

## Acknowledgements

The work described in this paper was carried out under contracts with the Generation Department and the Operations Department of Eletrobras, the Brazilian holding company for power. The contributions of B.G. Gorenstin, J.P. Costa and N.M. Campodonico of Cepel, the Brazilian electric power research center, have been essential for the development of methodology and related software. The comments and suggestions of J. Stedinger (Cornell University) and of two anonymous referees are gratefully acknowledged.

## References

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

J.R. Birge, "Solution methods for stochastic dynamic linear programs," Report 80/29, Systems Optimization Laboratory, Department of Operations Research, Stanford University (Stanford, CA, 1980).

S. Gal, "Parameter iteration dynamic programming", *Management Science* (1989).

J.L. Kennington and R.V. Helgason, *Algorithms for Network Programming* (Wiley, New York, 1984).

M.V.F. Pereira and L.M.V.G. Pinto, "Stochastic optimization of a multireservoir hydroelectric system — a decomposition approach", *Water Resources Research* 21(6) (1985).

R. J.-B. Wets, "Large scale linear programming techniques," in: Y. Ermoliev and R. Wets, eds., *Numerical Methods for Stochastic Optimization* (Springer, Berlin, 1988) Chapter 3.