Annals of Operations Research 31 (1991) 425–444

APPROXIMATE SCENARIO SOLUTIONS IN THE PROGRESSIVE HEDGING ALGORITHM

A numerical study with an application to fisheries management

Thorkell HELGASON

University of Iceland, Dunhagi 3, IS-107 Reykjavik, Iceland

Stein W. WALLACE

Haugesund Maritime College, Skåregaten 103, N-5500 Haugesund, Norway

This paper describes how the scenario aggregation principle can be combined with approximate solutions of the individual scenario problems, resulting in a computationally efficient algorithm where two individual Lagrangian-based procedures are merged into one. Computational results are given for an example from fisheries management. Numerical experiments indicate that only crude scenario solutions are needed.

Keywords: Stochastic programming, discrete optimal control, scenario aggregation, dynamic optimization, multistage decision making, decomposition, approximation, fisheries models.

1. Introduction

The purpose of this paper is to discuss implementational and algorithmic questions concerning scenario aggregation as defined by Rockafellar and Wets [10]. Our main contribution is to present an efficient way of combining the scenario aggregation method with an approximate solution procedure for the individual scenario problems. Our method is based on the Hamiltonians for each scenario problem, thereby achieving an approach where the inner (local) and outer (global) iterations melt together into one overall procedure.

The method under investigation is applied to a standard problem from fisheries management. Stochasticity is attached to the growth and recruitment of the stock. The understanding of what affects the recruitment in a fish stock is one of the major unsolved questions in fisheries management and therefore it is, in our view, important to use stochastic models so as to capture at least some of the effects of the uncertainty. Stochastic models are not yet standard in this subject area, so this paper is also an attempt to show how results from standard models change as stochasticity is introduced.

Until very recently almost all computational work in stochastic programming was based on the L-shaped decomposition method, as described in Van Slyke and Wets [12]. The most widely spread implementation based on these ideas was

presented by Birge [1] and followed up by Gassmann [2]. Olsen [7] and Louveaux [4,5] have made major contributions to the multi-stage case. For general overviews we refer to Wets [14–17].

Using decomposition techniques based on Van Slyke and Wets [12] is algorithmically very involved unless the randomness is very limited, most likely presented in terms of scenarios. It is therefore natural to turn to scenario aggregation. Not much computational work has been done with this approach, however, but some interesting results can be found in Mulvey and Vladimirou [6]. We hope that our approach will open up a new path for computational studies.

2. Scenarios and control processes

This paper focuses on a time-discrete controllable process in time *stages* $t = 0, \ldots, T$. The *state* of the process at time t is denoted by the variable x_t. The transition from the state at time t to that at time $t+1$ is governed by a *control* variable u_t but is also dependent on an auxiliary variable, the *scenario*.

A scenario is a vector

$$s = (s_0, \ldots, s_t, \ldots, s_T).$$

The set of all scenarios, S, is assumed to be finite. Let s^t be the equivalence class of all scenarios having the first t coordinates, $s_0, \ldots, s_{t-1}$, in common. Note that each s^{T+1} contains only one scenario.

The scenarios can be illustrated with a tree structure, the *scenario tree*, where the nodes correspond to the scenario classes and dependence to the inclusion $s^{t+1} \subseteq s^t$. The root of the tree corresponds to S.

The transition from state to state will be allowed to depend upon the past history of the scenario but will not be affected by its future. Thus we can describe the transition as follows:

$$x_{t+1} = G_t(x_t, u_t, s^t). \tag{2.1}$$

The initial state x_0 is given. Furthermore the controls may be bounded,

$$A_t \leqslant u_t \leqslant B_t. \tag{2.2}$$

The following algorithms will cope with a more general set-up, i.e. where the transition depends on the entire previous history, not only of the scenario, but also of the states and controls. However, for the sake of simplicity, we will not introduce this here. (See Mulvey and Vladimirou [6] for a more general, but somewhat different, problem.)

Probabilities (subjective or objective) are attached to the scenarios. Thus scenarios will be viewed in terms of a stochastic vector $\boldsymbol{s}$ with stochastic components $\mathbf{s}_0, \ldots, \mathbf{s}_t, \ldots, \mathbf{s}_T$. We denote the probability of a particular realization of a scenario with

$$p(s) := \text{prob}(\boldsymbol{s} = s).$$

These probabilities are non-negative numbers and sum to 1. Furthermore, we need probabilities of scenarios conditional upon belonging to a certain equivalence class s^t at time t:

$$p(s \mid s^t) := \text{prob}(\boldsymbol{s} = s \mid \boldsymbol{s} \in s^t) = \frac{p(s)}{p(s^t)}, \tag{2.3}$$

where $p(s^t)$ is the probability mass of all scenarios belonging to the class s^t.

In the ensuing application the scenario components are independently distributed. In this case the conditional probability of $s = (s_0, \ldots, s_t, \ldots, s_T)$ is easily computed as

$$p(s \mid s^t) = \text{prob}(\boldsymbol{s}_t = s_t) \ldots \text{prob}(\boldsymbol{s}_T = s_T).$$

A *policy* is a function assigning to each scenario s a sequence of controls $u(s) = (u_0(s), u_1(s), \ldots, u_t(s), \ldots, u_T(s))$. We write $x_t(s)$ for the corresponding state variables.

A policy is *implementable* if for all $t = 0, \ldots, T$ the tth control is common to all scenarios in the same class s^t, i.e. if $u_t(s) = u_t(s')$ whenever $s^t = s'^t$. This means that the decision made at time t can only be dependent on the history of the process known at that time and not on its future.

A policy is *admissible* if it (always) satisfies constraints (2.2).

For any policy u and scenario class s^t define a decision $\underline{u}_t$ by computing the expected value

$$\underline{u}_t(s^t) := \sum_{s' \in s^t} p(s' \mid s^t) u_t(s').$$

Then define a new policy $\underline{u}$ by setting

$$\underline{u}_t(s') := \underline{u}_t(s^t) \quad \text{for all } s' \in s^t.$$

This new policy $\underline{u}$ is implementable and admissible (see (2.2)). Note that equivalence of u and $\underline{u}$ obviously amounts to implementability:

PROPOSITION

A policy u is implementable if and only if

$$u_t(s) = \sum_{s' \in s^t} p(s' \mid s^t) u_t(s') \quad \text{for all } s \text{ and } t = 0, \ldots, T. \tag{2.4}$$

Now suppose there is some *immediate cost* associated with each period depending on the state, the decision taken and the current scenario. But as with the transition function we will assume that the cost at stage t is not dependent on the future. Thus we can denote the cost function as $F_t(x_t, u_t, s^t)$. Furthermore there may be some terminal cost $Q(x_{T+1})$. In addition we assume that these costs are discounted with the discount factor α, where $0 \leqslant \alpha \leqslant 1$.

We wish to control the process with an implementable policy to minimize the expected present value of the total cost. This expected value of the immediate costs equals

$$\sum_{t=0}^{T} \alpha^t \sum_{s \in S} p(s) F_t(x_t(s), u_t(s), s^t),$$

which according to (2.3) can be written as

$$\sum_{t=0}^{T} \alpha^t \sum_{s^t} p(s^t) \sum_{s' \in s^t} p(s' \mid s^t) F_t(x_t(s'), u_t(s'), s^t). \tag{2.5}$$

The second sum in (2.5) is over all possible scenario classes at stage t.

The goal is to determine the minimum of (2.5) (plus expected terminal cost) given (2.1) and (2.2), using an implementable policy, i.e. a policy which satisfies (2.4).

This can be summed up as follows:

$$\min \left\{ \sum_{t=0}^{T} \alpha^t \sum_{s^t} p(s^t) \sum_{s' \in s^t} p(s' \mid s^t) F_t(x_t(s'), u_t(s'), s^t) \right.$$

$$\left. + \alpha^{T+1} \sum_{s \in S} p(s) Q(x_{T+1}(s)) \right\} \tag{2.6}$$

subject to

$$x_{t+1}(s) = G_t(x_t(s), u_t(s), s^t), \tag{2.7}$$

$$A_t \leqslant u_t(s) \leqslant B_t \tag{2.8}$$

and

$$u_t(s) = \sum_{s' \in s^t} p(s' \mid s^t) u_t(s') \tag{2.9}$$

for all s, and $t = 0, \ldots, T$. Here $x_0(s)$ is given (and is independent of s).

3. The progressive hedging algorithm

Rockafellar and Wets [10] have developed the so-called *Progressive Hedging Algorithm* (PHA) which can be specialized to solve a stochastic optimal control problem like (2.6)–(2.9). Its fundamental idea is to add the only constraints that tie together the different scenarios to the objective function via Lagrangian multipliers. Denote these multipliers with $W_t(s)$ (but discount them to make the notation more coherent). Thus the Lagrangian term to be added to (2.6) is

$$\sum_{t=0}^{T} \alpha^t \sum_{s^t} p(s^t) \sum_{s' \in s^t} p(s' \mid s^t) W_t(s') \left[u_t(s') - \sum_{s'' \in s''} p(s'' \mid s'') u_t(s'') \right]. \tag{3.1}$$

If the multipliers can be chosen such that

$$\sum_{s' \in s^t} p(s' \mid s^t) W_t(s') = 0 \tag{3.2}$$

for all t and s and using the fact that $s'^t = s^t$ expression (3.1) simplifies to

$$\sum_{t=0}^{T} \alpha^t \sum_{s^t} p(s^t) \sum_{s' \in s^t} p(s' \mid s^t) W_t(s') u_t(s'). \tag{3.3}$$

The objective function (2.6) together with this Lagrangian term becomes

$$\sum_{t=0}^{T} \alpha^t \sum_{s^t} p(s^t) \sum_{s' \in s^t} p(s' \mid s^t) \left[F_t(x_t(s'), u_t(s'), s^t) + W_t(s') u_t(s') \right] + \alpha^{T+1} \sum_{s \in S} p(s) Q(x_{T+1}(s)).$$

Fortunately this Lagrangian function as well as the remaining constraints are separable with respect to the scenarios, so the optimization of this Lagrangian function decomposes. But Rockafellar and Wets go a step further in the direction of augmented Lagrangians (see Hestenes [3] and Powell [9]) by adding penalties for deviating from implementability. These penalty terms can be seen in PHA1 below.

The progressive hedging algorithm, adapted to our formulation, runs as follows:

PHA0. Initialize: Set $W_t(s) = 0$ for all stages t and scenarios s, choose an implementable policy $\underline{u}(s)$ (e.g. by solving (2.6)–(2.9) using average values of the stochastic variables), choose $\rho > 0$.

PHA1. For each $s \in S$ solve the optimization problem in the u's:

$$\min \left\{ \sum_{t=0}^{T} \alpha^t \left[\left(F_t(x_t, u_t, s^t) + W_t(s) u_t + \tfrac{1}{2} \rho \, \| u_t - \underline{u}_t(s) \|^2 \right) \right] + \alpha^{T+1} Q(x_{T+1}) \right\}$$

subject to

$$x_{t+1} = G_t(x_t, u_t, s^t), \quad x_0 \text{ given},$$

and

$$A_t \leqslant u_t \leqslant B_t.$$

Let $u(s) = (u_0(s), \ldots, u_T(s))$ denote the vector of optimal controls, and $x(s) = (x_0(s), \ldots, x_T(s))$ the corresponding (optimal) states.

PHA2. For $t = 0, \ldots, T$ and all scenarios s calculate new average controls

$$\underline{u}_t(s) := \sum_{s' \in s^t} p(s' \mid s^t) u_t(s').$$

and then update the *implementability multipliers*,

$$W_t(s) \leftarrow W_t(s) + \rho\big[u_t(s) - \underline{u}_t(s)\big].$$

Return to PHA1 or terminate.

Rockafellar and Wets [10] measured the error of the algorithm after k steps with an expected value (with respect to the distribution of the scenarios):

$$d^{(k)} := \mathbf{E}\left[\sum_{t=0}^{T} \alpha^t \left\{ \left\| \underline{u}_t(s)^{(k)} - \underline{u}_t(s)^{(k-1)} \right\|^2 + \frac{1}{\rho^2} \left\| W_t(s)^{(k)} - W_t(s)^{(k-1)} \right\|^2 \right\}\right].$$

Here k refers to the iteration number. Rockafellar and Wets prove that $d^{(k)}$ is a monotonically decreasing sequence, given certain regularity conditions and assuming that step PHA1 is solved sufficiently accurately. Hence their termination criterion is

$$d^{(k)} \leqslant \varepsilon. \tag{3.4}$$

We refer to $d^{(k)}$ as the *error sequence*.

According to the theory of augmented Lagrangians the updating step PHA2 is a Newtonian step in an (augmented) dual method for problem (2.6)–(2.9) (see e.g. Pierre and Lowe [8]). All this is explicitly proved by Rockafellar and Wets [10]. They furthermore prove that the algorithm converges even if step PHA1 is not solved exactly.

It is easily seen that the multipliers $W_t(s)$ satisfy eq. (3.2). This is essential for short-cutting the objective function in step PHA1 as indicated by (3.3).

4. Solving the subproblems with a Lagrangian step

PHA1 is a standard time-discrete optimal control problem which can again be solved with a Lagrangian method, thus decomposing the problem according to the stages. Let us express the Lagrangian multipliers of the transition constraints at stage t as a product $\alpha^{t+1}\mu_t$, whereby μ_t has the meaning of a *shadow price*. For a fixed scenario s the Lagrangian function for the problem in step PHA1 becomes

$$\sum_{t=0}^{T} \alpha^t \big[F_t(x_t, u_t, s^t) + W_t(s)u_t + \tfrac{1}{2}\rho \| u_t - \underline{u}_t(s) \|^2 + \alpha\mu_t\big(G_t(x_t, u_t, s^t) - x_{t+1}\big)\big] + \alpha^{T+1}Q(x_{T+1}).$$

The Kuhn–Tucker optimality conditions for the state variables yield recursion formulas for the shadow prices. First,

$$\mu_T = \frac{\partial}{\partial x_{T+1}} Q(x_{T+1}) \tag{4.1}$$

and then for $t = T, \ldots, 1$

$$\mu_{t-1} = \frac{\partial}{\partial x_t} F_t(x_t, u_t, s^t) + \alpha \mu_t \frac{\partial}{\partial x_t} G_t(x_t, u_t, s^t). \tag{4.2}$$

Then optimization of the Lagrangian with respect to the control variables decomposes into *Hamiltonian* subproblems, one for each time stage. Thus for all t we have to solve the problem

$$\min\left[F_t(x_t, u_t, s) + W_t(s) u_t + \tfrac{1}{2}\rho \| u_t - \underline{u}_t(s) \|^2 + \alpha \mu_t G_t(x_t, u_t, s^t) \right] \tag{4.3}$$

in u_t subject to $A_t \leqslant u_t \leqslant B_t$. Here the objective has been simplified by leaving out the common factor α^t and omitting the term $-\alpha \mu_t x_{t+1}$ which is independent of the optimizing variable.

Now let us look at conditions (4.1)–(4.3) in connection with step PHA1 of the progressive hedging algorithm. Obviously a first step in solving these scenario subproblems can be obtained by going once through the following two steps.

L1. Compute the shadow prices:

$$\mu_T(s) = \frac{\partial}{\partial x_{T+1}} Q(x_{T+1}(s))$$

and then for $t = T, \ldots, 1$

$$\mu_{t-1}(s) = \frac{\partial}{\partial x_t} F_t(x_t(s), u_t(s), s^t) + \alpha \mu_t(s) \frac{\partial}{\partial x_t} G_t(x_t(s), u_t(s), s^t).$$

L2. Solve the Hamiltonian optimization problems in u_t for $t = 0, \ldots, T$, i.e.

$$\min\left[F_t(x_t(s), u_t, s^t) + W_t(s) u_t + \tfrac{1}{2}\rho \| u_t - \underline{u}_t(s) \|^2 + \alpha \mu_t(s) G_t(x_t(s), u_t, s^t) \right],$$

subject to $A_t \leqslant u_t \leqslant B_t$.

Call the solution $u_t(s)$. Before increasing t update the states x, i.e.

$$x_{t+1}(s) = G_t(x_t(s), u_t(s), s^t), \quad x_0(s) = x_0.$$

Iterating between these two steps amounts to a kind of a dual method for solving the scenario subproblems (see e.g. Pierre and Lowe [8]). Such pure dual methods are generally unstable. Augmenting the Lagrangian with penalties for violating the constraints improves the convergence (and helps finding the dual variables, which is no problem here). This amounts to augmenting each Hamiltonian with a new penalty term. Thus the objective in step L2 becomes

$$\min\Big[F_t(x_t(s), u_t, s^t) + W_t(s) u_t + \tfrac{1}{2}\rho \| u_t - \underline{u}_t(s) \|^2 + \tfrac{1}{2}\eta \big\| x_{t+1}(s) - G_t(x_t(s), u_t, s^t) \big\|^2 + \alpha \mu_t(s) G_t(x_t(s), u_t, s^t) \Big]. \tag{4.4}$$

The progressive hedging algorithm with only one iteration of the above type in the optimization of the subproblem will be called the *Lagrangian Progressive Hedging Algorithm* (LPHA) and runs as follows:

LPHA0. Initialize: Set $W_t(s) = 0$ for all stages t and scenarios s. Choose an implementable policy $\underline{u}(s)$ for all s. This may be chosen independently of s, e.g. by solving (2.6)–(2.9) using average values of the stochastic variables. Calculate initial values for the average states as

$$\underline{x}_{t+1}(s) = G_t\big(\underline{x}_t(s), \underline{u}_t(s), s^t\big), \quad \underline{x}_0(s) = x_0.$$

Choose $\rho > 0$.

LPHA1. Compute the shadow prices for all scenarios s:

$$\mu_T(s) = \frac{\partial}{\partial x_{T+1}} Q\big(\underline{x}_{T+1}(s)\big)$$

and for $t = T, \ldots, 1$

$$\mu_{t-1}(s) = \frac{\partial}{\partial x_t} F_t\big(\underline{x}_t(s), \underline{u}_t(s), s^t\big) + \alpha\mu_t(s)\frac{\partial}{\partial x_t} G_t\big(\underline{x}_t(s), \underline{u}_t(s), s^t\big).$$

LPHA2. For $t = 0, \ldots, T$ and all scenarios s:
Solve the Hamiltonian optimization problems in u_t

$$\min\big[F_t\big(x_t(s), u_t, s^t\big) + W_t(s)u_t + \tfrac{1}{2}\rho \,\|\, u_t - \underline{u}_t(s) \,\|^2 + \alpha\mu_t(s)G_t\big(x_t(s), u_t, s^t\big)\big],$$

subject to $A_t \leqslant u_t \leqslant B_t$.
Call the solution $u_t(s)$. Before increasing t update x:

$$x_{t+1}(s) = G_t\big(x_t(s), u_t(s), s^t\big), \quad x_0(s) = x_0.$$

LPHA3. For $t = 0, \ldots, T$ and all scenarios s calculate new average controls

$$\underline{u}_t(s) := \sum_{s' \in s'} p(s' \mid s^t) u_t(s')$$

and then update the implementability multipliers,

$$W_t(s) \leftarrow W_t(s) + \rho\big[u_t(s) - \underline{u}_t(s)\big].$$

Calculate new average states (as in LPHA0), and return to LPHA1 or terminate.

For details on the implementation see Wallace and Helgason [13]. Note that in each iteration the algorithm goes through the scenario tree only twice; once upward and once downward.

According to the convergence theorems of Rockafellar and Wets, step PHA1 in the progressive hedging algorithm needs only to be solved approximately. They have a criterion for how precise the solution must be to guarantee convergence and, among other things, the monotonicity of the error sequence.

As said before, a more precise solution of the subproblems (step PHA1) calls for several inner (or *local*) iterations between steps LPHA1 and LPHA2. Then, of course, two amendments must be made. First, the shadow prices in step LPHA1 must be recomputed from the policy and corresponding states in step LPHA2. Notationally this means that the underlining should be left out in step LPHA1 (apart from the first time it is applied in any global iteration). Secondly, the latter penalty term (with coefficient η) in (4.4) should be added again into the Hamiltonian in step LPHA2. However, the above mentioned criterion of Rockafellar and Wets has not yet been implemented in the LPHA algorithm so as to control the number of these inner iterations.

We will come back to these questions in the numerical tests in section 6, where provisions are actually made for the use of inner looping between steps LPHA1 and LPHA2.

5. A simple example from fisheries management

Suppose we are harvesting annually some portion u_t of a biomass x_t and that the stock is regenerated according to a version of the Schaefer model (Schaefer [11]):

$$x_{t+1} = x_t + sx_t(1 - x_t/K) - u_t x_t.$$

Here s is called the *growth ratio*. Parameter K is often called the carrying capacity since the stock increases – in the absence of harvesting – until it reaches the level K. For any constant harvesting ratio u with $0 \leqslant u \leqslant s$ this equation has a steady state solution

$$x = K(1 - u/s).$$

The maximal steady state harvesting or *maximum sustainable yield* (MSY) is obtained when $u = s/2$ in which case $x = K/2$.

Now let us assume that the growth ratio in any year t is a stochastic variable, s_t, independently distributed from year to year, taking on finitely many values in the range [0,2]. Let $\underline{s}$ denote the expected value of the growth ratio.

Suppose that our objective is to maximize the expected present value of the yield over a period of T years with a given value of the stock at the end of the planning period, denoted by $Q(x)$.

Thus we wish to solve

$$\max \mathbf{E}\left[\sum_{t=0}^{T} \alpha^t u_t x_t + \alpha^{T+1} Q(x_{T+1})\right],$$

subject to

$$x_{t+1} = x_t + s_t x_t(1 - x_t/K) - u_t x_t, \quad \text{given } x_0, \text{ and } 0 \leqslant u_t \leqslant 1.$$

Here **E** refers to expected value and α is a discount factor.

We use a naïve expression for $Q(x)$ by taking the present value of the infinite sustainable (or steady state) yield from the (fixed) stock x under the assumption of average growth ratio, i.e.

$$Q(x) = \frac{\underline{s}x(1 - x/K)}{1 - \alpha}.$$

This is a proxy for the expected yield from a fixed stock size. Here $\underline{s}$ is, as said before, the average growth ratio.

This model greatly idealizes the real situation in the fisheries. Among other things it does not take into account the age structure of the fish population which, at least for long-lived species like cod, is of major importance. Computationally, though, this problem is complicated enough. Note that it is nonlinear (actually quadratic) both in the objective function and the constraints.

This problem can hardly be solved with methods other than the aggregation principle presented here. Straight-forward dynamic programming would most likely be unusable due to the "curse of dimensionality". On the other hand, it is certainly possible to design a dynamic programming scheme (e.g. by interpolating between discrete state values) which would work. Nevertheless, conceptually at least, the progressive hedging algorithm seems to be more tractable for high-dimensional problems. This holds in particular if considering relatively few or short scenarios suffices in order to get at least a near optimal solution with the algorithm. We come back to this in the conclusion.

Decompositions based on the L-shaped method would probably fail due to the size of the problem. An extremely large nonlinear nested decomposition procedure would be needed. One would have to keep track of, and solve repeatedly, one nonlinear optimization problem for each node in the scenario tree.

6. Numerical experiments

The fisheries model of section 5 was used for numerical experimentation with the LPHA algorithm. The following fixed data were used:

$$K = 10, \quad \alpha = 0.9.$$

In the first part of this section we experiment with a rather modest number of scenarios by setting $T = 5$ and letting the stochastic growth ratio take on only two values. In this case the number of scenarios is $2^6 = 64$. The stochastic variable s can take on the two values 0 and 2 with probabilities 0.8 and 0.2, respectively. These values and probabilities are rather extreme and would hardly occur in real systems. We can consider three examples A, B and C where $x_0 = 2.5$, 5 and 7.5 respectively.

As said earlier, the LPHA algorithm becomes identical to the (original) PHA algorithm if the subproblems expressed by steps LPHA1 and LPHA2 are solved exactly. Furthermore, we should get closer to the solution of the subproblems by adding an inner looping between steps LPHA1 and LPHA2. We have experimented with hybrid algorithms with two or more iterations of this inner loop. We refer in this context to the number of *local* or *inner* iterations and the number of *global* iterations, respectively.

6.1. OPTIMAL SOLUTION

The following three diagrams (figs. 1–3) show the top four stages (corresponding to $t = 0, 1, 2, 3$) and part of the last stage (stage $T = 5$) of the scenario tree for the fisheries problem. The solutions are obtained using the LPHA algorithm with optimal values of the penalty parameter ρ (see subsection 6.4) and with required precision $\varepsilon = 10^{-7}$ (see (3.4)). We believe these results to be very close to the true optimum.

The path to the far left in the trees corresponds to the growth ratio s constantly taking its lower value. The box at the bottom left indicates this extreme at stage 5. The opposite holds for the far right of the tree. The upper number in each box is the value of the biomass or the state variable x and the

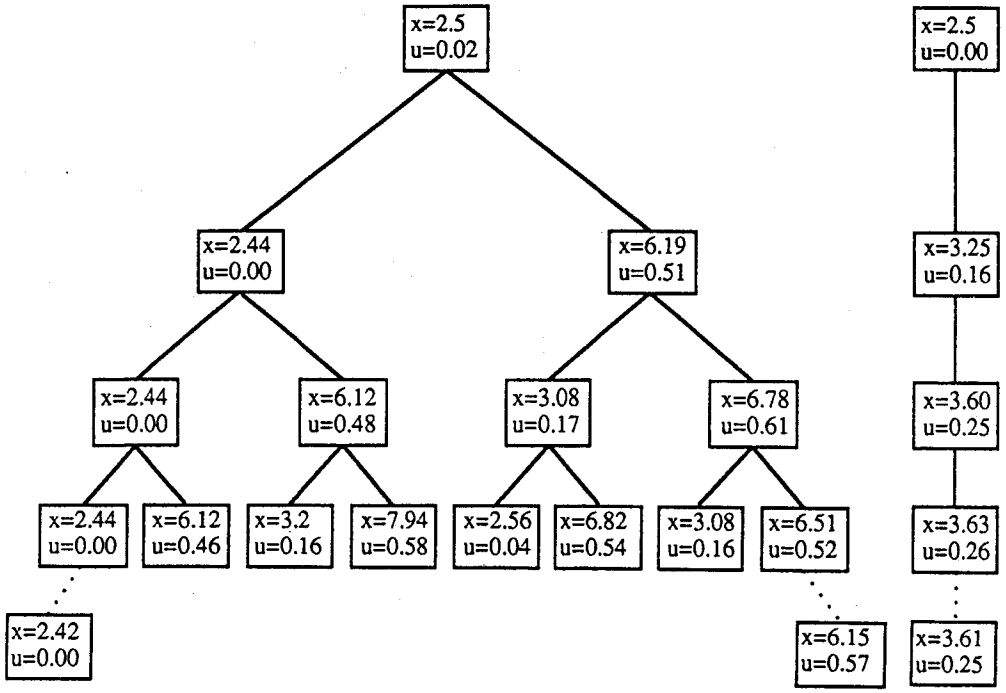

Fig. 1. Scenario tree for example A with optimal solutions for the stochastic case on the left and the deterministic case on the right.

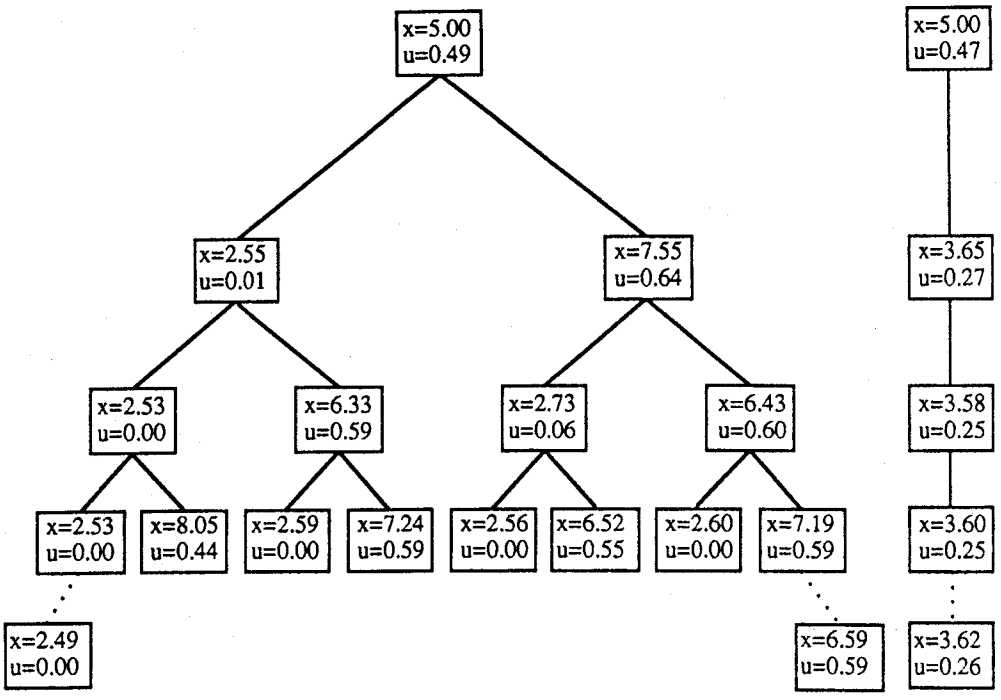

Fig. 2. Scenario tree for example B with optimal solutions for the stochastic case on the left and the deterministic case on the right.

lower number is the corresponding optimal (implementable) control u, i.e. the harvesting ratio. The vertical paths to the right in the diagrams show the solution of the optimal control problem when the stochastic variables s_t are replaced with their expected value $\underline{s} = 0.4$.

In these examples the difference between the first decisions (which for practical management are the only decisions that matter) in the stochastic and the deterministic models is marginal. However, we tested different values of the discount factor α. For a value of α equal to 0.95 the difference was greater and actually in the opposite direction. For an initial biomass equal to 5 units (cf. example B) the stochastic solution calls for a lower catch in the first year than does the deterministic one. This may give us food for thought about fisheries management in that it is generally based on deterministic models. Evidently stochasticity does matter but it depends upon the circumstances whether a decision based on a deterministic model leads to under- or overfishing.

6.2. RATE OF CONVERGENCE

Figure 4 shows the rate of convergence as measured by the error term d and by the actual error in the objective value compared to the optimal one for example B and two different values of the global penalty parameter ρ.

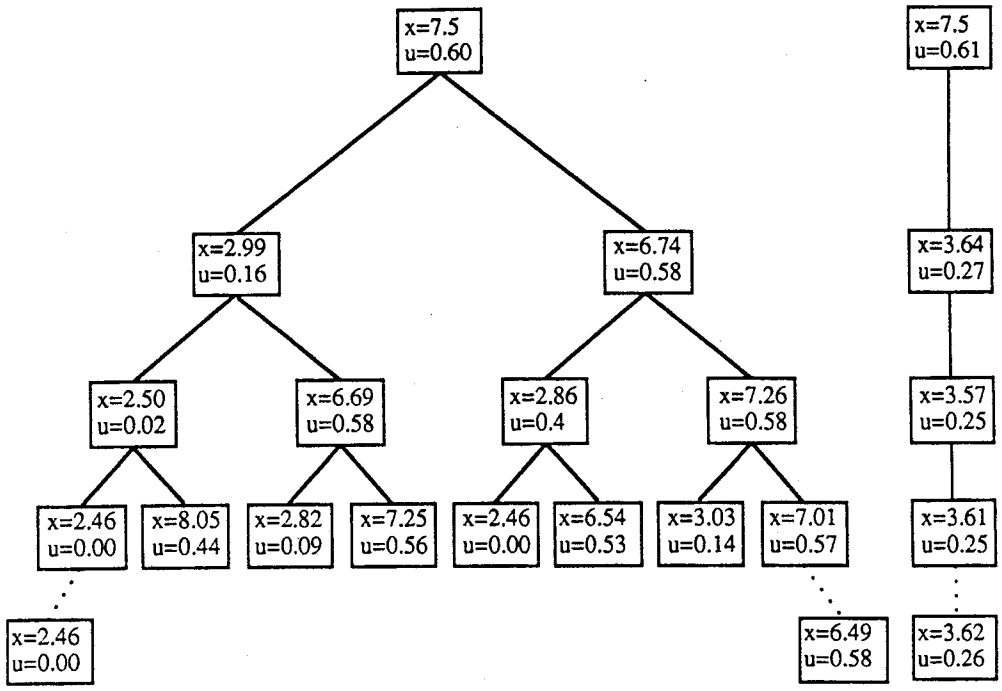

Fig. 3. Scenario tree for example C with optimal solutions for the stochastic case on the left and the deterministic case on the right.

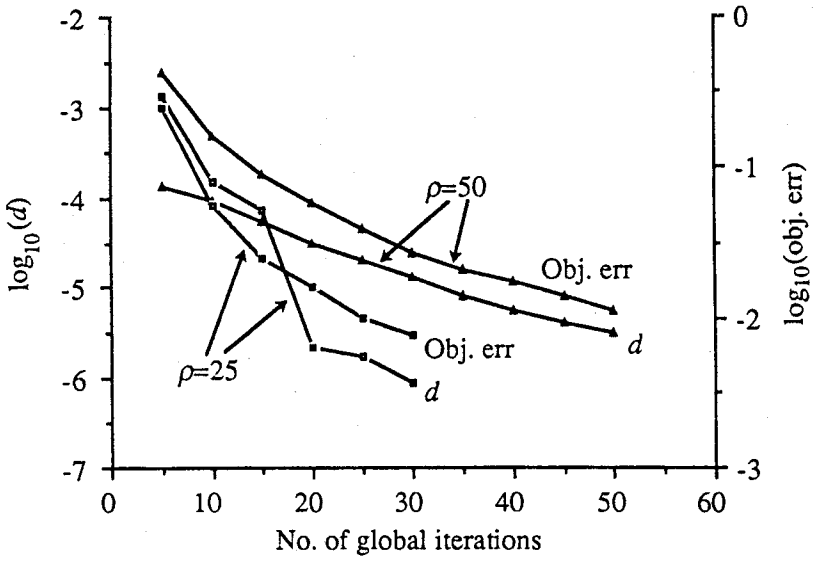

Fig. 4. Convergence properties of example B. The error term d and the objective error are shown as functions of the number of global iterations for two values of the penalty parameter ρ. By "objective error" is meant the difference between the achieved and the true optimum.

These sequences are monotonically decreasing although the scenario subproblems are only solved with one inner iteration, thus supporting the assumption (made later) that one iteration is enough.

The program was written in Pascal and run on a SUN 3/50. For these examples the CPU time per global iteration is 1.2 seconds.

After about 10 iterations the error term d is of the order 10^{-4} and an (implementable) objective value is obtained which is less than 0.5% under the true optimum. On the other hand, the error in the first decision, $\underline{u}_0$, is substantially larger or some 30% off the optimal value. This discrepancy is due to a very flat objective function. For the practical decision maker this means that he can choose from a large selection of solutions as regards the initial decision. These solutions are all nearly optimal. Unfortunately, managers tend to interpret such results to allow them to overfish in the current year as this can always be compensated for with lower catches in the succeeding years. However, managers tend to postpone corrective measures indefinitely.

6.3. CHOICE OF THE PENALTY PARAMETER

The choice of the global penalty parameter ρ seems to be of vital importance. This is demonstrated in table 1, which shows the number of (global) iterations needed to achieve a certain precision as measured by d and the corresponding deviation of the (implementable) objective value from the optimum one for different values of the penalty parameter. The smallest value shown is $\rho = 25$. For smaller values such as $\rho = 15$ the algorithm did not converge. In example C, even $\rho = 25$ is too low.

We observe from table 1 that for a given level of the error term d, the actual error in the objective value varies considerably. This is due to the fact that ρ affects d directly. Therefore it can be misleading to judge the convergence properties just by looking at this measure. This is even more serious if the scenario subproblems are solved only approximately, as is the case in table 1. On the other hand, in an actual setting where the optimal objective value is not known, d is the only available measure of convergence. It is furthermore useful in the sense that lack of monotonicity in d is an indication of no convergence. In that case ρ should be increased.

Figure 5 shows the progress of the objective value in example B for three different values of the penalty. Here one of the penalty values, $\rho = 15$, does not ensure convergence.

The only conclusion to be drawn from table 1 and fig. 5 is that the penalty should be as small as possible, provided it is large enough to guarantee convergence. This is in agreement with the findings of Mulvey and Vladimirou [6] and consistent with the general theory of augmented Lagrangian methods (see Pierre and Lowe [8]).

Table 1
Number of global iterations needed to reach a certain precision d for different penalty parameters ρ. "Obj. err." shows the difference between the achieved and the true optimum objective values

Example A					
		$d = 0.001$	$d = 0.0001$	$d = 0.00001$	$d = 0.000001$
$\rho = 25$	Itr.	5	14	23	28
	Obj.err.	0.40	0.022	0.0063	0.00044
$\rho = 50$	Itr.	3	13	31	52
	Obj.err.	1.1	0.20	0.015	0.0040
$\rho = 100$	Itr.	2	10	39	83
	Obj.err.	1.5	0.76	0.063	0.0072
Example B					
		$d = 0.001$	$d = 0.0001$	$d = 0.00001$	$d = 0.000001$
$\rho = 25$	Itr.	6	14	29	52
	Obj.err.	0.14	0.029	0.0078	0.0024
$\rho = 50$	Itr.	3	10	33	66
	Obj.err.	0.62	0.16	0.24	0.0068
$\rho = 100$	Itr.	3	7	34	92
	Obj.err.	0.87	0.54	0.075	0.013
Example C					
		$d = 0.001$	$d = 0.0001$	$d = 0.00001$	$d = 0.000001$
$\rho = 25$	Itr.	No convergence			
	Obj.err.				
$\rho = 50$	Itr.	5	14	28	54
	Obj.err.	0.40	0.10	0.025	0.0063
$\rho = 100$	Itr.	4	15	39	70
	Obj.err.	0.87	0.27	0.066	0.019

Experiments show that increasing the discount factor calls for an increase in the penalty parameter. If the algorithm is to be applicable to general problems, some automatic way of choosing and adjusting the penalty parameter must be implemented. Experience with the current examples – also supported by the theory – indicates that a sufficiently large penalty coincides with monotone decrease in the error term d. Thus an increase in d should immediately lead to some increment in the penalty parameter.

6.4. SOLVING THE SCENARIO SUBPROBLEMS EXACTLY

The basic theory of Rockafellar and Wets assumes that the individual scenario subproblems are solved exactly. This is not done in our LPHA algorithm. As we mention in section 4 the subproblems can be solved with any desired precision by looping between steps LPHA1 and LPHA2 after addition of an inner penalty term with an appropriate parameter η (see (4.4)) which in our case turned out to be for $\eta = 1$.

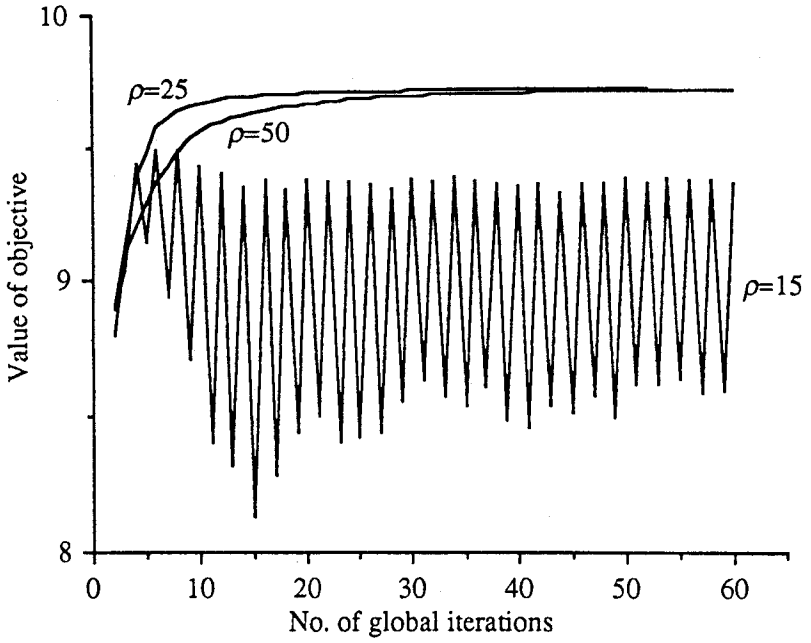

Fig. 5. Convergence of the objective value for different choices of ρ in example B.

Experiments with the three versions of the fisheries problem for a moderate number of stages ($T \leqslant 7$) showed that there is nothing to be gained by solving the subproblems with more precision than that already obtained with one inner iteration. Solving the subproblems more precisely reduces the number of global iterations only marginally but the CPU time depends heavily on the number of inner iterations. The CPU time is entirely based on the number of local and global iterations. For a SUN 3/50 we have estimated this formula to be as follows:

$$\text{CPU} = (0.37 + 0.85 I_L) I_G.$$

Here CPU time is measured in seconds and I_L is the number of local iterations and I_G the number of global iterations. (For a SUN 3/60 these numbers can be multiplied by 0.63.) Thus the total CPU time actually increases if any effort is spent on solving the subproblems more precisely than already done with one local iteration.

Furthermore, if the subproblems are solved exactly, the global penalty parameter has to be increased to ensure global convergence. As said in the previous subsection convergence is secured for $\rho \geqslant 25$ with the basic LPHA algorithm for these examples, but if the subproblems are solved exactly a much larger penalty is needed ($\rho \geqslant 200$).

What happens is that although the total objective increases monotonically from iteration to iteration in the progressive hedging algorithm this need not be the

case if the objective function is broken down into individual scenarios. Solving the subproblems exactly amplifies these oscillations of the individual scenarios, which then again have to be damped with a larger penalty. But increasing the penalty, on the other hand, slows down the speed of the algorithm, as we already see from table 1.

These results strongly justify our suggestion of solving the subproblems of the progressive hedging algorithm (step PHA1) only in a very approximate way. Whether this also holds for larger more realistic problems with more stages needs to be tested. Also other methods for solving the scenario subproblems more accurately than the one used here might be faster and thus change the picture, although we doubt they will.

6.5. LARGE SCENARIO TREES

The examples cited so far are of modest size with only 64 scenarios. In this subsection we report about tests with thousands of scenarios. The example is still based on the fisheries management model of section 5 with $K = 10$, and $x_0 = 7.5$ but now with $\alpha = 0.95$. The main difference is that we have increased the number of time stages and/or the number of children in the scenario trees.

In the first of these examples the number of possible outcomes for the growth ratio s_t has been increased from two to three, namely to the values 0, 1, and 2

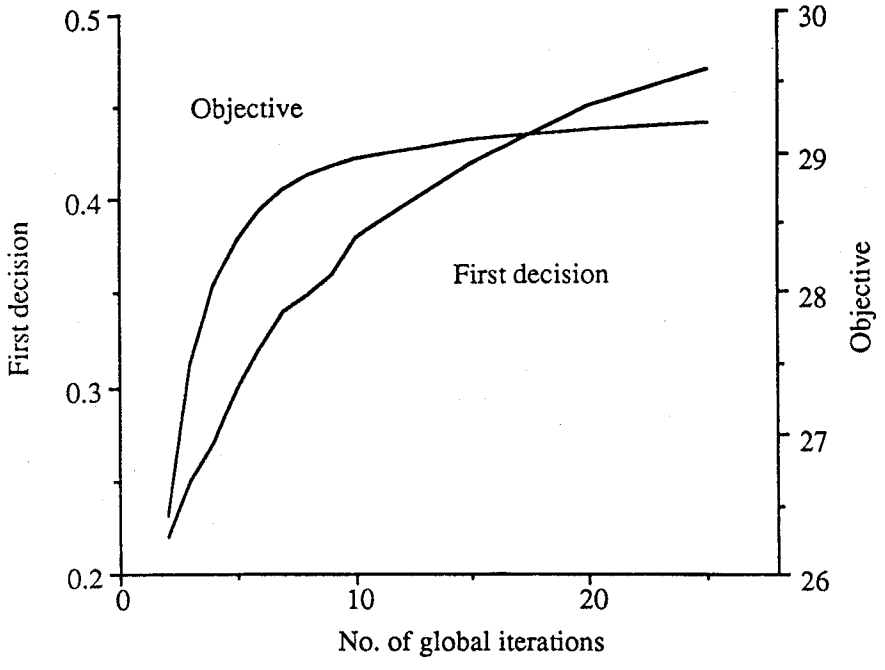

Fig. 6. Convergence of the objective value and the first decision for an example with $3^8 = 6591$ scenarios (see further description in subsection 6.5). There is one inner iteration and $\rho = 100$.

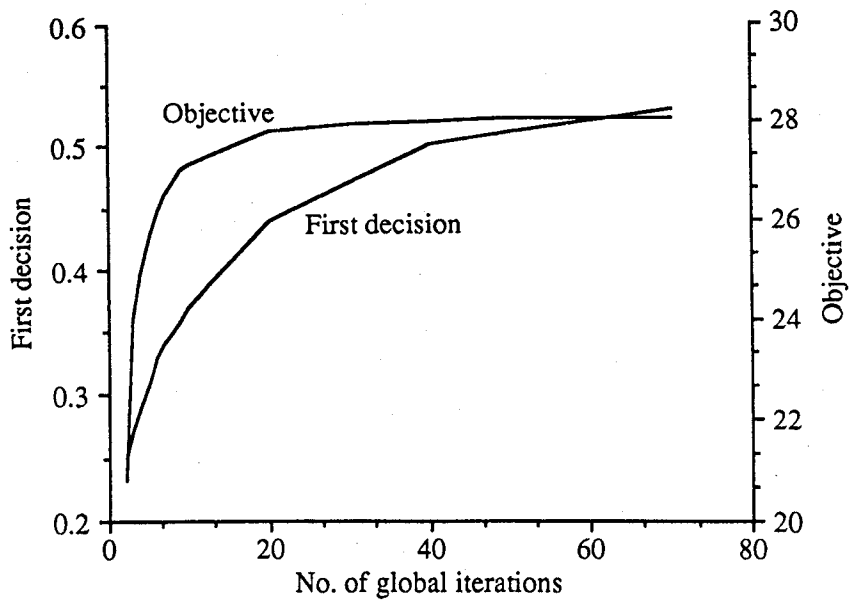

Fig. 7. Convergence of the objective value and the first decision for an example with $2^{15} = 32768$ scenarios (see further description in main text). There is one inner iteration and $\rho = 100$.

with probabilities 0.55, 0.30, and 0.15, respectively. The expected value is now 0.6; therefore in the deterministic case the maximum sustainable yield is achieved with a harvesting ratio u of 0.3. There are 8 periods ($T = 7$), so the number of scenarios is $3^8 = 6591$. We used only one inner iteration. Again tests showed that nothing is gained by solving the subproblems with higher precision.

Figure 6 shows results from this run with penalty $\rho = 100$. The first decision and the objective value are shown as functions of the number of global iterations.

The CPU time per iteration on a SUN 3/50 for this example is about 6.9 minutes. It is interesting that most of this time, or 4.6 minutes, is needed for the calculation of averages, i.e. calculating implementable decisions (step LPHA3). The addition of one more time stage ($T = 8$) did not change the above solution or its behaviour. Now the number of scenarios is three times larger or $3^9 = 19683$. The CPU increased almost proportionally or up to 23.2 minutes per iteration.

A still larger example was tested. This time the growth ratio s_t again has only two values, 0 with probability 0.7 and 2 with probability 0.3. The expected value is still 0.6. On the other hand $T = 14$, which means that the number of scenarios is $2^{15} = 32768$. Now the convergence is rather slow. Even after 70 iterations, it is difficult to see whether the optimum has been reached or not (see fig. 7).

The penalty was chosen as 100, which was probably too low. Now the iteration time was 63 minutes, which means that the 70 iterations took more than 3 CPU days!

6.6. ALTERNATIVE STARTING VALUES

We have earlier discussed the optimal number of local iterations in LPHA, that is, the optimal number of iterations between LPHA1 and LPHA2 before we continue to LPHA3. We found the optimal number to be 1 for our examples. However, we might increase the benefit of this one iteration with a different starting value; step LPHA1 need not be initialized with an implementable control, as we are doing; any admissible control will do. As an alternative to using the latest implementable controls $\underline{u}_t(s)$, we have tested the effect of using the scenario solutions $u_t(s)$ from the previous global iteration. Let us refer to these approaches as the "original" and "modified" method, respectively.

A reason for testing the modified method is that one may expect $u_t(s)$ to be closer to the optimal scenario solution than $\underline{u}_t(s)$. On the other hand, the $\underline{u}_t(s)$ are implementable, while that is not the case for $u_t(s)$ (until convergence has been achieved). It is difficult to say which of these two approaches is best. Our computational test does not answer the question in a satisfactory way. To ensure convergence the penalty parameter ρ had to be larger in the modified method than in the original one. This in turn slows down the rate of convergence.

It is difficult to draw any conclusions from the experiments reported in this subsection, but since it may inspire those who try to implement the scenario aggregation algorithm it is included in the paper.

7. Conclusions and further research

We have shown how it is possible to implement the scenario aggregation procedure of Rockafellar and Wets in a simplified version by solving the individual scenario problems only in an approximate way, using an integrated application of a Lagrangian approach. Our computational results show that there are substantial savings in using approximate scenario solutions. In no cases have we observed that increasing the number of the inner iterations – needed to solve the subproblems more accurately – has a desirable effect on CPU time. We therefore believe that the approach tested here on a rather simple, but nonlinear, problem can be used in most contexts, i.e. exact solutions of the scenario problems are rarely needed.

Many aspects of the procedures need further investigations. In particular, we need a method for adjusting the penalties to ensure fast convergence. For any realistic problem the number of scenarios will be formidable. In order to make the progressive hedging algorithm universally applicable some scenario-saving improvement is called for. Obviously the scenarios do not all contribute equally to the only interesting solution, namely the optimal first decision. Hence it is tempting to develop a version of the algorithm whereby scenarios are generated gradually, the most "important" first, and then use the solution to generate the next scenario(s) and so on. This will be the subject of our next paper in this area.

Acknowledgements

We are indebted to the two referees who suggested many useful changes in the paper, in particular regarding the terminology. We would like to thank Kurt M. Alonso for his assistance with coding and running the program and for generating the graphics. Finally, we are grateful for partial support from the Science Institute of the University of Iceland.

References

[1] J.R. Birge, Decomposition and partitioning methods for multistage stochastic linear programs, Oper. Res. 33 (1985) 989–1007.

[2] H. Gassmann, Multi-period stochastic programming, Ph.D. Thesis, University of British Columbia, Vancouver, Canada (1987).

[3] M.R. Hestenes, Multiplier and gradient methods, J. Opt. Theory Appl. 4 (1969) 303–320.

[4] F. Louveaux, A solution method for multi-stage stochastic programs with recourse with applications to an energy investment problem, Oper. Res. 28 (1980) 889–902.

[5] F. Louveaux, Multistage stochastic programs with block-separable recourse, Math. Progr. Study 28 (1986) 48–62.

[6] J. Mulvey and H. Vladimirou, Solving multistage stochastic networks: An application of scenario analysis, Report SOR-88-1, Dept. of Civil Engineering and Operations Research, Princeton University (1988).

[7] P. Olsen, Multistage stochastic program with recourse: The equivalent deterministic problem, SIAM J. Control Optim. 14 (1976) 518–527.

[8] D.A. Pierre and M.J. Lowe, *Mathematical Programming Via Augmented Lagrangians: An Introduction with Computer Programs. Applied Mathematics and Computation 9* (Addison–Wesley, Reading, 1975).

[9] M.J.D. Powell, A method for nonlinear constraints in minimization problems, in: *Optimization*, ed. R. Fletcher (Academic Press, New York, 1969) pp. 283–298.

[10] R.T. Rockafellar and R.J.-B. Wets, The principle of scenario aggregation in optimization under uncertainty, Working paper WP-87-119, IIASA, Austria (1987), to appear in Math. Oper. Res.

[11] M.B. Schaefer, Some aspects of the dynamics of populations important to to the management of the commercial marine fisheries, Inter-Am. Trop. Tuna Comm. Bull. 1 (1954) 27–56.

[12] R. Van Slyke and R.J.-B. Wets, L-shaped linear programs with applications to optimal control and stochastic programming, SIAM J. Appl. Math. 17 (1969) 638–663.

[13] S.W. Wallace and T. Helgason, Structural properties of the progressive hedging algorithm, this volume.

[14] R.J.-B. Wets, Stochastic programming: Solution techniques and approximation schemes, in: *Mathematical Programming. The State of the Art*, eds. A. Bachem, M. Grötschel and B. Korte (Springer-Verlag, Berlin, 1983) pp. 566–603.

[15] R.J.-B. Wets, Large scale linear programming techniques, in: *Numerical Techniques in Stochastic Optimization*, eds. Y. Ermoliev and R.J.-B. Wets (Springer, Berlin, 1988) pp. 65–94.

[16] R.J.-B. Wets, The aggregation in scenario analysis and stochastic optimization, in: *Algorithms and Model Formulations in Mathematical Programming*, ed. S.W. Wallace (Springer, Berlin, 1989) pp. 91–113.

[17] R.J.-B. Wets, Stochastic programming, in: *Handbooks in Operations Research and Management Science*, vol. 1: *Optimization*, eds. G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd (North-Holland, Amsterdam, 1989) pp. 573–629.