Annals of Operations Research 64(1996)39–65

Parallel decomposition of large-scale stochastic nonlinear programs

John R. Birge[1)]
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, USA

Charles H. Rosa[2)]
International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria

Many practical decision problems involve both nonlinear relationships and uncertainties. The resulting stochastic nonlinear programs become quite difficult to solve as the number of possible scenarios increases. In this paper, we provide a decomposition method for problems in which nonlinear constraints appear within periods. We also show how the method extends to lower bounding refinements of the set of scenarios when the random data are independent from period to period. We then apply the method to a stochastic model of the U.S. economy based on the Global 2100 method developed by Manne and Richels.

Keywords: Decomposition, economics, environment, parallel computation, stochastic programming.

1 Introduction

Nonlinearity relationships and uncertainties appear in most models for decision making. These complications can often void the use of methods based on linearization and deterministic data. Standard nonlinear programming approaches do not generally take into consideration special structures that allow for efficient computation. In this paper, we show that stochastic programs with special nonlinear structure can be solved efficiently with decomposition and parallel computation.

[1)] This material is based upon work supported by the National Science Foundation under Award Numbers SES-9211937 and DDM-9215921.
[2)] The research was performed under appointment to the U.S. Department of Energy, Graduate Fellowships for Global Change Program, administered by Oak Ridge Institute for Science and Education.

Our primary motivation for this analysis is to solve problems for economic decision making in which many nonlinearities constrain current resources and decisions but where interperiod constraints can be adequately approximated with a linear relationship. In section 2, we show that a nested decomposition method allows this problem to be divided into separate subproblems that correspond to each individual combination of decisions within a time period and under a given scenario.

In general, stochastic programming models of this type approximate a more complete model by incorporating some form of discretization of a model with a full random variable distribution. If this is the case, then the approximations are gradually refined to obtain increasingly accurate approximations. We show in section 4 that, if the model has a serial independence property, then the result of the decomposition method is generally a lower bound on the complete distribution value. Moreover, we show that all previous subproblems constructed in the algorithm can be maintained when further refinements are included.

The specific example we consider is a generalization of the Global 2100 Model in Manne and Richels [9, 16]. This model fits into the conditions of intraperiod nonlinearities and only linear relationships between periods. We show how our methods perform on this Stochastic Global 2100 model in sections 5 and 6, in particular, showing the results of employing parallel computation.

2 Nested decomposition algorithm for stochastic nonlinear programming

Efficient computation of an approximate solution in a timely fashion is key to the practical implementation of stochastic programming. Efficient solution often depends on taking advantage of the special structure in stochastic programs in which models within a given time period appear quite similar for varying random parameters. Many decomposition methods use this characteristic by operating on a single subproblem for each random outcome. Dantzig and Madansky [6] and Benders [1] originated this idea by treating each stage of the dual formulation of a two-stage linear programming problem separately. Using the dual solutions from the second stage problem, they progressively built up an inner linearization approximation of the recourse function associated with the second stage costs which they then included in the first stage problem to solve the original problem. Van Slyke and Wets [15], working with the primal version of a finite scenario two-stage stochastic linear program, used similar ideas to construct an outer linearization approximation of the expected second stage recourse costs. O'Neill [12] used these outer linearization approximation concepts to solve a deterministic multi-stage nonlinear programming problem, while Birge [2] extended the results of Van Slyke and Wets to the multi-stage stochastic linear programming case.

Noel and Smeers [11] treated the case of the dual formulation of a convex multi-stage stochastic nonlinear program. Pereira and Pinto [13] worked with a multi-stage stochastic programming problem with nonlinear objective function, linear constraints,

and complete recourse. They then advanced the previous work by using sampling to reduce the magnitude of computation required to construct the outer linearization approximation. Our method builds on these approaches by using the outer linearization idea to approximate the recourse function (along with its effective domain) in the primal stochastic nonlinear case with nonlinearities within a period, and linear linking constraints with special random structure. We differ from the preceding methods in our treatment of incomplete recourse. In particular, when we encounter infeasibility, our algorithm solves a series of subproblems of ever greater complexity to ensure a consistent solution across all stages and scenarios of the model. We show that this potentially expensive procedure turns out, in our numerical example, to be quite reasonable.

The general bounded convex nonlinear stochastic program we consider has the following form:

$$
\begin{aligned}
\text{minimize} \quad & f_1(x_1) + E_{\xi_2}(f_2(x_2)) + \cdots + E_{\xi_2,\ldots,\xi_T}(f_T(x_T)) \\
\text{subject to} \quad & A_1 x_1 \leq b_1, \\
& g_1(x_1) \geq 0, \\
& B_1 x_1 + A_2 x_2 \leq (C_2(\xi_2))x_1, \\
& g_2(x_2) \geq 0, \qquad\qquad \text{(CNSP)} \\
& \vdots \\
& B_{T-1} x_{T-1} + A_T x_T \leq (C_T(\xi_T))x_{T-1}, \\
& g_T(x_T) \geq 0, \\
& x_1, \ldots, x_T \geq 0,
\end{aligned}
$$

where A_t is $m_t \times n_t$, B_t is $m_{t+1} \times n_t$, x_t is $n_t \times 1$, $C_t(\xi_t)$ is a random $m_t \times n_{t-1}$ array defined on the probability space (Ξ_t, F_t, F_t) and all constraints hold almost surely.

The following key assumptions are made regarding the structure of the mathematical program:

- The function $f_t : R^{1+} \to R$ is convex.
- The function $g_t : R^{n_t^{*+}} \to R$ $(n_t^* \ll n_t)$ is concave.
- For any $(\xi_2, \ldots, \xi_T) \in (\Xi_2, \ldots, \Xi_T)$ (a particular scenario), there exists an $(x_1, \ldots, x_T)$ which is a strictly interior point.
- The random vectors, ξ_t, are mutually stochastically independent.

Note that the nonlinear objective and constraints in CNSP only appear within a single period. This general stochastic programming formulation with random right-hand sides and transition matrices $(C_t(\xi_t))$ allows for a wide range of applications. In our example (see also Birge and Rosa [3]), this transition corresponds to a return on investment in new technology.

The problem is as follows when we model it as a dynamic program with stages $1,\ldots,T$ and states x_t for $t = 1,\ldots,T-1$. We can find an optimal solution by using backward recursion as follows:

$$z_t(x_{t-1}) = E_{\xi_t}[Z_t(x_{t-1}, \xi_t)],$$

where

$$\begin{aligned} Z_t(x_{t-1}, \xi_t) = \text{minimize } & f_t(x_t) + z_{t+1}(x_t) \\ \text{subject to } & g_t(x_t) \geq 0, \\ & B_{t-1}x_{t-1} + A_t x_t \leq (C_t(\xi_t))x_{t-1}, \\ & x_t \geq 0, \end{aligned} \tag{DP}$$

for $t \geq 2$ and $z_{T+1}(*) = 0$ and

$$\begin{aligned} Z_1 = \text{minimize } & f_1(x_1) + z_2(x_1) \\ \text{subject to } & g_1(x_1) \geq 0, \\ & A_1 x_1 \leq b_1, \\ & x_1 \geq 0. \end{aligned}$$

Note that this characterization uses x_t alone as the state vector, so that future realizations of ξ_t are independent of the past.

To solve this dynamic programming form, we assume that the random variables ξ_t have finite support. In general, this may not be true, but a finite approximation may yield a bounding approximation. In our example, we will in fact show that this is possible and that a refinement approach can use all algorithm information from one period to the next.

Program CNSP then becomes the deterministic equivalent as follows:

$$\text{minimize } f_1(x_1) + \sum_{i=1}^{s(2)} p_2^i * p_1(f_2(x_2^i)) + \cdots +$$

$$\sum_{i=1}^{s(T)} p_T^i * p_{T-1}^{a(i)} * \cdots * p_1(f_T(x_T^i))$$

$$\begin{aligned} \text{subject to } & A_1 x_1 \leq b_1, \\ & g_1(x_1) \geq 0, \\ B_1 x_1 + A_2 x_2^i \leq (C_2(\xi_2^i))x_1, & \quad \forall i = 1,\ldots,s(2), \\ g_2(x_2^i) \geq 0, & \quad \forall i = 1,\ldots,s(2), \\ \vdots & \\ B_{T-1}x_{T-1}^{a(i)} + A_T x_T^i \leq (C_T(\xi_T^i))x_{T-1}^{a(i)}, & \quad \forall i = 1,\ldots,s(T), \\ g_T(x_T) \geq 0, & \quad \forall i = 1,\ldots,s(T), \end{aligned}$$

$$x_1, \{x_2^i,\ i = 1,\ldots,s(2)\},\ldots,\{x_T^i,\ i = 1,\ldots,s(T)\} \geq 0.$$

The matrix A_t is $m_t \times n_t$, B_t is $m_{t+1} \times n_t$, x_t^i is $n_t \times 1$, $C_t(\xi_t^i)$ is the value that $C_t(\xi_t)$ takes in the ith scenario of the tth period. The function f_t with domain = R^{1+} is convex. The function g_t with domain = $R^{n_t^{*+}}$ ($n_t^* \ll n_t$) is concave. The number of scenarios in period t is $s(t)$. The conditional probability that scenario i in period t will occur given that scenario $a(i)$ (the predecessor node of node i in period t) in period $t-1$ has occurred is p_t^i, and $p_1 = 1.0$. The set of descendant nodes of node i is D_t^i.

We can then rewrite the problem in dynamic programming format. Solve:

$$
\begin{aligned}
Z_1 = \text{minimize } & f_1(x_1) + \sum_{d \in D_1} p_2^d Z_2^d(x_1) \\
\text{subject to } & g_1(x_1) \geq 0, \\
& A_1 x_1 \leq b_1, \\
& x_1 \geq 0,
\end{aligned}
$$

where

$$
\begin{aligned}
Z_t^i(x_{t-1}^{a(i)}) = \text{minimize } & f_t(x_t^i) + \sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i) \\
\text{subject to } & g_t(x_t^i) \geq 0, \\
& B_{t-1} x_{t-1}^{a(i)} + A_t x_t^i \leq (C_t(\xi_t^i)) x_{t-1}^{a(i)}, \\
& x_t^i \geq 0
\end{aligned}
$$

and

$$
\begin{aligned}
Z_T^i(x_{T-1}^{a(i)} = \text{minimize } & f_T(x_T^i) \\
\text{subject to } & g_T(x_T^i) \geq 0, \\
& B_{T-1} x_{T-1}^{a(i)} + A_T x_T^i \leq (C_T(\xi_T^i)) x_{T-1}^{a(i)}, \\
& x_T^i \geq 0.
\end{aligned}
$$

As with the continuously distributed random variable case, $Z_{t+1}^d(x_t^i)$ is convex, which implies that $\sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i)$ is convex.

Our general approach is to use an outer linearization of $z_{t+1}(x_t)$ within each period. This outer linearization is progressively refined by adding *cuts* or linear supports found by solutions of future period subproblems. The basis for the approach is that the cuts are valid lower bounds on the value function z_t and that the algorithm eventually refines the linearization to any degree of accuracy.

To show that the algorithm solves CNSP, we first establish some fundamental properties of the subproblems in the following lemmas.

Lemma 1

$Z_t(x_{t-1}, \xi_t)$ is convex in ξ_t for any x_{t-1}.

Proof
Consider

$$Z_T(x_{T-1}, \xi_T) = \text{minimize } f_T(x_T)$$
$$\text{subject to} \quad g_T(x_T) \geq 0,$$
$$B_{T-1}x_{T-1} + A_T x_T \leq (C_T(\xi_T))x_{T-1},$$
$$x_T \geq 0.$$

Let x_T^1 be optimal when $\xi_T = \xi_T^1$ and x_T^2 be optimal when $\xi_T = \xi_T^2$. Clearly,

$$B_{T-1}x_{T-1} + A_T x_T^1 \leq (C_T(\xi_T^1))x_{T-1} \quad \text{and} \quad B_{T-1}x_{T-1} + A_T x_T^2 \leq (C_T(\xi_T^2))x_{T-1}$$

imply that

$$B_{T-1}x_{T-1} + A_T(\lambda * x_T^1 + (1-\lambda) * x_T^2) \leq (C_T(\lambda * \xi_T^1 + (1-\lambda) * \xi_T^2)x_{T-1}$$

for all $0 \leq \lambda \leq 1$. Also,

$$g_T(x_T^1) \geq 0 \quad \text{and} \quad g_T(x_T^2) \geq 0$$

imply that

$$g_T(\lambda * x_T^1 + (1-\lambda) * x_T^2) \geq 0$$

for all $0 \leq \lambda \leq 1$ as $g_T(\cdot)$ is concave. Then

$$\begin{aligned} Z_T(x_{T-1}, \lambda * \xi_T^1 + (1-\lambda) * \xi_T^2) &\leq f_T(\lambda * x_T^1 + (1-\lambda) * x_T^2 \\ &\leq \lambda * f_T(x_T^1) + (1-\lambda) * f_T(x_T^2) \\ &\leq \lambda * Z_T(x_{T-1}, \xi_T^1) + (1-\lambda) * Z_T(x_{T-1}, \xi_T^2). \end{aligned}$$

Thus, $Z_T(x_{T-1}, \xi_T)$ is convex in ξ_T for any given x_{T-1}.

$$z_T(x_{T-1}) = \int_{\xi_T \in \Xi_T} Z_T(x_{T-1}, \xi_T) dF(\xi_T) d\xi_T$$

is convex as it is a convex combination of convex functions.

$$Z_{T-1}(x_{T-2}, \xi_{T-1}) = \text{minimize } f_{T-1}(x_{T-1}) + z_T(x_{T-1})$$
$$\text{subject to} \quad g_{T-1}(x_{T-1}) \geq 0,$$
$$B_{T-2}x_{T-2} + A_{T-1}x_{T-1} \leq (C_{T-1}(\xi_{T-1}))x_{T-2},$$
$$x_{T-1} \geq 0,$$

is convex in ξ_{T-1} as already established since $f_{T-1}(x_{T-1}) + z_T(x_{T-1})$ is convex, the sum of two convex functions. This implies that $z_{T-1}(x_{T-2})$ is convex as before. This continues recursively until we see that $Z_t(x_{t-1}, \xi_t)$ is convex in ξ_t for all t and for all x_{t-1}. It is easy to show that $Z_t(x_{t-1}, \xi_t)$ is also convex in x_{t-1} for any ξ_t. □

Lemma 2

The mathematical program for which $Z_t^i(x_{t-1}^{a(i)})$ is the optimal value is equivalent to:

$$\begin{aligned}
\text{minimize } & f_t(x_t^i) + \sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i) \\
\text{subject to } & g_t(x_t^i) \geq 0, \\
& B_{t-1}x_{t-1}^{a(i)} + A_t x_t^i \leq (C_t(\xi_t^i))x_{t-1}^{a(i)}, \\
& D_t^{l,i} x_t^i + d_t^{l,i} \geq 0, \quad l = 1,\ldots,L_t^i, \\
& x_t^i \geq 0.
\end{aligned}$$

Proof

In solving the dynamic program (DP), it is clear that a solution exists only if a feasible solution exists. Clearly this means that at optimality and for any node i, x_t^i, the solution to $Z_t^i(x_{t-1}^{a(i)})$, must be feasible $d \in D_t^i$. That is, $Z_{t+1}^d(x_t^i) > -\infty$ for all $d \in D_t^i$. This of course implies that $\sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i) \geq -\infty$. Since we have shown that $\sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i)$ is convex in x_t^i, the set of x_t^i for which this function has a finite value must be convex. As a result, an explicit constraint describing this region that we add to the program will not alter the problem. How will this region be described? Suppose that for a particular value of $x_t^i = \bar{x}_t^i$ there exists $d \in D_t^i$ such that $Z_{t+1}^d(\bar{x}_t^i) = -\infty$. This implies that the constraints

$$\begin{aligned}
g_{t+1}(x_{t+1}^d) &\geq 0, \\
B_t \bar{x}_t^i + A_{t+1}x_{t+1}^d &\leq (C_{t+1}\xi_{t+1}^d)\bar{x}_t^i, \\
x_{t+1}^d &\geq 0
\end{aligned}$$

are inconsistent. This in turn implies that

$$\begin{aligned}
G_{t+1}^d(\bar{x}_t^i) = \text{maximize } & -v^+ - e^t v^- \\
\text{subject to} & \\
\mu_{t+1}^d: \quad & B_t \bar{x}_t^i + A_{t+1}x_{t+1}^d - Iv^- \leq (C_{t+1}\xi_{t+1}^d)\bar{x}_t^i, \\
\lambda_{t+1}^d: \quad & g_{t+1}(x_{t+1}^d) + v^+ \geq 0, \\
& x_{t+1}^d, v^- \in R^{m_{t+1}}, v^+ \in R^1 \geq 0,
\end{aligned}$$

has a strictly negative optimum objective value solution. Thus, the expression $G_{t+1}^d(x_t^i) \geq 0$, where $G_{t+1}^d(x_t^i)$ is a concave function of x_t^i, defines the convex region from which x_t^i may be chosen. Clearly, a finite subset of the supporting hyperplanes of a convex region defines a region of which the convex region is a subset.

A supporting hyperplane is guaranteed to exist at every point x_t^i by our assumption about the existence of a feasible interior point and the well known Slater condition

given our assumption of a strict interior point. Proceeding in the usual manner, we denote a finite number of supports as:

$$D_t^{l,i} x_t^i + d_t^{l,i} \geq 0, \quad l = 1, \ldots, L_t^i,$$

where

$$D_t^{l,i} = \mu_{t+1}^{d^l}[(C_{t+1}(\xi_{t+1}^d)) - B_t]$$

and

$$d_t^{l,i} = -v^{+^l} - e^t v^{-^l} - \mu_{t+1}^{d^l} * (A_{t+1} x_{t+1}^{d^l} - I v^{-^l}) + \lambda_{t+1}^{d^l} * (g_{t+1}(x_{t+1}^{d^l}) + v^{+^l}).$$

The program can, thus, be written:

$$\begin{aligned}
Z_t^i(x_{t-1}^{a(i)}) = \ & \text{minimize } f_t\ (x_t^i) + \theta_t^i \\
& \text{subject to} \\
\sigma_t^i: \quad & g_t(x_t^i) \geq 0, \\
\pi_t^i: \quad & B_{t-1} x_{t-1}^{a(i)} + A_t x_t^i \leq (C_t(\xi_t^i)) x_{t-1}^{a(i)}, \\
\mu_t^i: \quad & \theta_t^i \leq \textstyle\sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i), \\
\lambda_t^{i,l}: \quad & D_t^{l,i} x_t^i + d_t^{l,i} \geq 0, \quad l = 1, \ldots, L_t^i, \\
& x_t^i \geq 0.
\end{aligned}$$

We have already established that $\sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i)$ is convex in x_t^i. Since it is convex, it is clear that we can develop a lower bound on the function with a set of supporting hyperplanes. With a lower bound on $\sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i)$ for all $t = 2, \ldots, T$ and $i = 1, \ldots, s(t)$, we have obtained a lower bound for Z_1. □

Lemma 3

A lower bound can be obtained for $Z_t^i(x_{t-1}^{a(i)})$ by solving the following program:

$$\begin{aligned}
\bar{Z}_t^i(x_{t-1}^{a(i)} = \ & \text{minimize } f_t\ (x_t^i) + \theta_t^i \\
& \text{subject to} \\
\sigma_t^i: \quad & g_t(x_t^i) \geq 0, \\
\pi_t^i: \quad & B_{t-1} x_{t-1}^{a(i)} + A_t x_t^i \leq (C_t(\xi_t^i)) x_{t-1}^{a(i)}, \\
\mu_t^i: \quad & E_t^{l,i} x_t^i + e_t^{l,i} \leq \theta_t^i, \quad l = 1, \ldots, K_t^i, \\
\lambda_t^{i,l}: \quad & D_t^{l,i} x_t^i + d_t^{l,i} \geq 0, \quad l = 1, \ldots, L_t^i, \\
& x_t^i \geq 0,
\end{aligned}$$

where K_t^i and L_t^i are finite numbers.

Proof
Consider

$$\begin{aligned}
\overline{Z}^i_{T-1}(x^{a(i)}_{T-2}) = \ & \text{minimize } f_{T-1}(x^i_{T-1}) + \theta^i_{T-1} \\
& \text{subject to} \\
\sigma^i_{T-1}: \quad & g_{T-1}(x^i_{T-1}) \geq 0, \\
\pi^i_{T-1}: \quad & B_{T-2}x^{a(i)}_{T-2} + A_{T.-1}x^i_{T-1} \leq (C_{T-1}(\xi^i_{T-1}))x^{a(i)}_{T-2}, \\
\lambda^{i,l}_{T-1}: \quad & D^{l,i}_{T-1}x^i_{T-1} + d^{l,i}_{T-1} \geq 0, \quad l = 1, \ldots, L^i_{T-1}, \\
& x^i_{T-1} \geq 0.
\end{aligned}$$

This program will give a lower bound on the value of $Z^i_{T-1}(x^{a(i)}_{T-2})$ since θ^i_t is unbounded below. Clearly, a better bound exists for $\sum_{d \in D^i_T} Z^d_T(x^i_{T-1})$ than $\theta^i_t = -\infty$. Consider $Z^d_T(x^i_{T-1})$ for some $d \in D^i_T$.

$$\begin{aligned}
Z^d_T(x^i_{T-1}) = \ & \text{minimize } f_T(x^d_T) \\
& \text{subject to} \\
\sigma^d_T: \quad & g_T(x^d_T) \geq 0, \\
\pi^d_T: \quad & B_{T-1}x^i_{T-1} + A_T x^d_T \leq (C_T(\xi^d_T))x^i_{T-1}, \\
& x^d_T \geq 0.
\end{aligned}$$

Proceeding just as we did in the previous lemma, we can develop the dual of this program and from this and strong duality, a lower supporting hyperplane of $Z^d_T(x^i_{T-1})$. This has the following form:

$$\begin{aligned}
\theta^i_{T-1} &\geq \sum_{d \in D^i_{T-1}} p^d_T Z^d_T(x^i_{T-1}) \\
&\geq \sum_{d \in D^i_{T-1}} p^d_T \{\pi^{d*}_T(-B_{T-1} + (C_T(\xi^d_T)))x^i_{T-1} \\
&\qquad f_T(x^{d*}_T) + \sigma^{d*}_T(g_T(x^{d*}_T)) - \pi^{d*}_T(A_T x^{d*}_T)\} \\
&= E^{K^i_{T-1},i}_{T-1} x^i_{T-1} + e^{K^i_{T-1},i}_{T-1}.
\end{aligned}$$

This same argument proceeds recursively so that we can easily construct a lower bound on $\sum_{d \in D^i_t} p^d_{t+1}\overline{Z}^d_{t+1}(x^i_t)$ as before.

$$\theta^i_t \geq \sum_{d \in D^i_t} p^d_{t+1} Z^d_{t+1}(x^i_t)$$

$$
\begin{aligned}
&\geq \sum_{d \in D_t^i} p_{t+1}^d \{\pi_{t+1}^{d*}(-B_t + (C_{t+1}(\xi_{t+1}^d)))x_t^i \\
&\quad f_{t+1}(x_{t+1}^{d*}) + \sigma_{t+1}^{d*}(g_{t+1}(x_{t+1}^{d*})) - \pi_{t+1}^{d*}(A_{t+1}x_{t+1}^{d*}) \\
&\quad + \sum_{l=1}^{K_{t+1}^d} \mu_{t+1}^{d*,l}[E_{t+1}^{l,d}x_{t+1}^{d*} + e_{t+1}^{l,d}] \\
&\quad + \sum_{l=1}^{L_{t+1}^d} \lambda_{t+1}^{d*,l}[D_{t+1}^{l,d}x_{t+1}^{d*} + d_{t+1}^{l,d}]\} \\
&= E_t^{K_t^i,i}x_t^i + e_t^{K_t^i,i}.
\end{aligned}
$$

The lemma is proven. □

Thus, we have shown that a new DP formulation can be constructed from the old one that will return a lower bound on the optimal solution.

3 Solution procedure

In practice, rather than constructing cuts for each node problem ahead of time and then solving the modified DP to obtain the lower bound, we will use Birge's [2] NDSPA algorithm (modified for the nonlinear case as in [12]) to construct the cuts dynamically and solve the DP at the same time. We show that any level of precision can be obtained.

The problem constructed for every node in the stochastic tree is the following:

$$\overline{Z}_t^i(x_{t-1}^{a(i)}) = \text{minimize } f_t(x_t^i) + \theta_t^i \tag{1.1}$$

subject to

$$\sigma_t^i: \quad g_t(x_t^i) \geq 0, \tag{1.2}$$

$$\pi_t^i: \quad B_{t-1}x_{t-1}^{a(i)} + A_t x_t^i \leq (C_t(\xi_t^i))x_{t-1}^{a(i)}, \tag{1.3}$$

$$\mu_t^{i,l}: \quad E_t^{l,i}x_t^i + e_t^{l,i} \leq \theta_t^i, \quad l = 1, \ldots, K_t^i, \tag{1.4}$$

$$\lambda_t^{i,l}: \quad D_t^{l,i}x_t^i + d_t^{l,i} \geq 0, \quad l = 1, \ldots, L_t^i, \tag{1.5}$$

$$x_t^i \geq 0, \tag{1.6}$$

where the problem describing $\overline{Z}_T^i(x_T^{a(i)})$ includes no θ_T^i and the K_t^i optimality cuts and L_t^i feasibility cuts are added during the course of the algorithm.

Nested Decomposition for Stochastic Nonlinear Programming Algorithm (NDSNPA)
Choose $\varepsilon > 0$ (the optimality cut stopping criterion) and $\delta > 0$ (the feasibility cut stopping criterion).

Step 1 Solve (1) for $t = 1$ where $\theta_1 = 0$, $K_1 = 0$, $L_1 = 0$, and (1.3) is replaced by

$$A_1 x_1 \leq b_1. \tag{2}$$

(The scenario index i has been dropped for the period 1 problem.) Set $\theta_t^i = 0$ and $K_t^i = L_t^i = 0$ in problem (1) for all t and scenarios i at t. (The K_t^i and L_t^i indices are updated whenever a constraint (1.4) or (1.5) is added to (1).) $k_1 = k_2^1 = \cdots = k_2^{s(2)} = \cdots = k_T^1 = \cdots = k_T^{s(T)} = 0$. $(k_1, k_2^1, \ldots, k_2^{s(2)}, \ldots, k_T^1, \ldots, k_T^{s(T)}$ are iterated whenever $x_1, x_2^1, \ldots, x_2^{s(2)}, \ldots, x_T^1, \ldots, x_T^{s(T)}$ are updated.)

Step 2 If the period 1 problem (1) is infeasible, STOP. The problem is infeasible. Otherwise, $k_1 = k_1 + 1$ and let $x_1^{k_1}$ be the current optimal solution of (1) for $t = 1$. Use $x_1^{k_1}$ as an input in (1.3) for $t = 2$ and all ξ_2^i, $i = 1, \ldots, s(2)$.

If any period 2 problem is infeasible and

$$0.0 > D_1^l (x_1)^{k_1} + d_1^l + \delta, \tag{3}$$

then add a feasibility constraint (1.5) to (1) for $t = 1$, set $t = 1$, and return to step 1. Else if any period 2 problem is infeasible and

$$0.0 > D_1^l (x_1)^{k_1} + d_1^l \tag{4}$$

then solve the following program and, if it is feasible, obtain a consistent $(x_1, x_2^1, \ldots, x_2^{s(2)})$ vector, let $t = 2$ and go to step 3. If it is infeasible, STOP. The problem is infeasible.

$$\begin{aligned}
\text{minimize} \quad & f_1(x_1) + \theta_1 \\
\text{subject to} \quad & g_1(x_1) \geq 0, \\
& A_1 x_1 \leq b_1, \\
& E_1^l x_1 + e_1^l \leq \theta_1, && l = 1, \ldots, K_1, \\
& D_1^l x_1 + d_1^l \geq 0, && l = 1, \ldots, L_1, \\
& x_1 \geq 0, \\
& g_2(x_2^i) \geq 0, && \forall i \in D_1, \\
& B_1 x_1 + A_2 x_2^i \leq (C_2(\xi_2^i)) x_1, && \forall i \in D_1, \\
& E_2^{l,i} x_2^i + e_2^{l,i} \leq \theta_2^i, && \forall i \in D_1, l = 1, \ldots, K_2^i, \\
& D_2^{l,i} x_2^i + d_2^{l,i} \geq 0, && \forall i \in D_1, l = 1, \ldots, L_2^i, \\
& x_t^i \geq 0,
\end{aligned} \tag{5}$$

Otherwise, let $t = 2$ and go to step 3.

Step 3 (a) Let the current period t optimal solutions be $(x_t^i)^{k_t^i}$ for $i = 1,\ldots,s(t)$. Solve (1) for $t+1$ and all $j = 1,\ldots,s(t+1)$ using the appropriate ancestor solution $(x_t^i)^{k_t^i}$ in (1.3).

(b) If any period $t+1$ problem is infeasible and

$$0.0 > D_t^{l,i}(x_t^i)^{k_t^i} + d_t^{l,i} + \delta, \tag{6}$$

add a feasibility constraint to the corresponding ancestor period t problem and let $t = t-1$.

If $t = 1$, go to step 1.

Otherwise, return to step 3(a).

Otherwise, if any period $t+1$ problem is infeasible and

$$0.0 > D_t^{l,i}(x_t^i)^{k_t^i} + d_t^{l,i}, \tag{7}$$

then iteratively solve a finite sequence of subproblems (8) for $n = 1,\ldots,t$ until one is feasible. If a problem is feasible, then one has a new set of values for each stage that ensure all periods from period 1 through the $(t+1)$st period are feasible (in particular, $(\{x_t^1, x_{t+1}^i\}_{i\in D_1}, \{x_t^2, x_{t+1}^i\}_{i\in D_2}, \ldots, \{x_t^{s(t)}, x_{t+1}^i\}_{i\in D_{s(t)}})$ are consistent).

minimize

$$p_{t-(n-1)}^{a^{n-1}(i)}(f_{t-(n-1)}(x_{t-(n-1)}^{a^{n-1}(i)}) + \theta_{t-(n-1)}^{a^{n-1}(i)}) +$$

$$\sum_{d\in D_{t-(n-1)}^{a^{n-1}(i)}} p_{t-(n-2)}^{d}(f_{t-(n-2)}(x_{t-(n-2)}^{d}) + \theta_{t-(n-2)}^{d}) + \ldots +$$

$$\sum_{i\in S(a_i^{n-1},t)} p_t^i(f_t(x_t^i) + \theta_t^i)$$

subject to

$$g_{t-(n-1)}(x_{t-(n-1)}^{a^{n-1}(i)}) \geq 0,$$

$$B_{t-n}x_{t-n}^{a^n(i)} + A_{t-(n-1)}x_{t-(n-1)}^{a^{n-1}(i)} \leq (C_{t-(n-1)}(\xi_{t-(n-1)}^{a^{n-1}(i)}))x_{t-n}^{a^n(i)},$$

$$E_{t-(n-1)}^{l}x_{t-(n-1)}^{a^{n-1}(i)} + e_{t-(n-1)}^{l} \leq \theta_{t-(n-1)}^{a^{n-1}(i)}, \quad l = 1,\ldots,K_{t-(n-1)}^{a^{n-1}(i)},$$

$$D_{t-(n-1)}^{l}x_{t-(n-1)}^{a^{n-1}(i)} + d_{t-(n-1)}^{l} \geq 0, \quad l = 1,\ldots,L_{t-(n-1)}^{a^{n-1}(i)},$$

$$x_{t-(n-1)}^{a^{n-1}(i)} \geq 0,$$

$$
\begin{aligned}
g_{t+1}(x_{t+1}^d) &\geq 0, && \forall d \in D_t^i,\ i \in S(a_i^{n-1}, t),\\
B_t x_t^i + A_{t+1} x_{t+1}^d &\leq (C_{t+1}(\xi_{t+1}^d)) x_t^i, && \forall d \in D_t^i,\ i \in S(a_i^{n-1}, t),\\
E_{t+1}^{l,d} x_{t+1}^d + e_{t+1}^{l,d} &\leq \theta_{t+1}^d, && \forall d \in D_t^i,\ l = 1, \ldots, K_{t+1}^d,\\
D_{t+1}^{l,d} x_{t+1}^d + d_{t+1}^{l,d} &\geq 0, && \forall d \in D_t^i,\ l = 1, \ldots, L_{t+1}^d,\\
x_t^i, x_{t+1}^d &\geq 0, && \forall d \in D_t^i,\ i \in S(a_i^{n-1}, t), \qquad (8)
\end{aligned}
$$

where a_i^{n-1} is the $(n-1)$st ancestor of node i ($a_i^0 = i$) and $S(a_i^{n-1}, t)$ is the set of successors of node a_i^{n-1} in the tth period.

If $t \leq T-2$, let $t = t+1$ and return to step 3(a).

Otherwise ($t = T-1$), remove any remaining $\theta_t^i = 0$ restrictions for all period t and scenarios i at t and, for each of these, let the current value of $(\theta_t^i)^{k_t^i} = \infty$. Go to step 4.

Otherwise, if all the subproblems (8) are infeasible, STOP. The entire problem is infeasible.

Step 4 (a) Find $E_t^{l,i}$ and $e_t^{l,i}$ for a new constraint (1.4) at each scenario t problem (1) using the current period $t+1$ solutions.

(b) If some i satisfies

$$(\theta_t^i)^{k_t^i} < E_t^{l,i}(x_t^i)^{k_t^i} - e_t^{l,i} + \varepsilon/(T-1), \qquad (9)$$

then add the new constraint (1.4) for which the above equation holds. Solve each period t problem (1). If a problem is infeasible, let $t = t-1$ and return to step 3(a). Otherwise, use the resulting $((x_t^i)^{k_t^i}, (\theta_t^i)^{k_t^i})$ to form (1.3) for the corresponding descendant period $t+1$ problems (1) and re-solve each period $t+1$ problem (1).

If $t < T-1$, let $t = t+1$ and go to step 3(a).

Otherwise, return to step 4(a).

Otherwise,

$$(\theta_t^i)^{k_t^i} \geq E_t^{l,i}(x_t^i)^{k_t^i} + e_t^{l,i} + \varepsilon/(T-1) \qquad (10)$$

for all scenarios i at t.

If $t > 1$, let $t = t-1$ and return to step 4(a).

Otherwise, STOP. The current solutions $(x_t^i)^{k_t^i}$, $t = 1, \ldots, T$ form an ε lower bound of (1).

We briefly discuss the issue of feasibility before proceeding to a discussion of the convergence behavior of our algorithm. We ensure that when moving from step 3 to

step 4 of our algorithm we always have a feasible solution. We do this iteratively using feasibility cuts and subproblem (8). We use feasibility cuts to construct an outer linearization approximation of our feasible region. This gives us δ feasibility. Then, if a period $t = \tau$ problem is δ feasible but not feasible to within zero tolerance (typically much smaller than δ), then we solve a sequence of subproblems (8).

These subproblems extend backwards (as n increases from 1 to $\tau - 1$) towards the root of our tree and spread outwards to encompass the subtree containing the current root node and all its descendent nodes through period τ. The first feasible solution we find ensures that we have a consistent solution from the root of the tree up through period $t = \tau$.

We note that, in general, it may be necessary to solve all problems $n = 1,\ldots,\tau - 1$ in order to find a consistent solution. It might be better to simply solve subproblem (8) once for $n = \tau - 1$. This would give us a consistent solution up through period $t = \tau$ after having solved only one subproblem [11]. We choose not to do this because we may find a consistent solution long before having to solve all $\tau - 1$ problems.

Solving several of the subproblems (8) when n is small is preferable to solving even one larger problem when $n = \tau - 1$. In fact, while experimenting with the class of stochastic energy economic models that we investigated for this paper, we never solved subproblem (8) more than once (i.e., $n = 1$). This indicates that we reaped significant savings by following the strategy that we did.

Finally, we note that in practice, since feasibility cuts can give any desired level of feasibility, one may want to skip entirely the step we have included to ensure certain feasibility. By doing this, subproblems will always be small (i.e., no bigger than one period) and, thus, will not grow unmanageable as the overall problem size increases.

We now wish to show that our algorithm terminates finitely with an approximate solution for any $\varepsilon > 0$ and $\delta > 0$. We first establish four more lemmas about the progress of the algorithm. Lemma 4 shows that whenever an optimality cut is added to the program, the approximate optimal value will increase. We note that this is true also for feasibility cuts. Lemmas 5 and 6 establish two properties of the algorithm at the point at which the algorithm proceeds from step 3 to step 4. Finally, lemma 7 derives an expression for $(\theta_t^i)^{k_t^i}$ that holds throughout the algorithm. After all of these lemmas have been introduced and proven, a general proof of convergence of the algorithm will be presented.

Lemma 4

If, for any t, i, $(\theta_t^i)^{k_t^i} < E_t^{l,i}(x_t^i)^{k_t^i} + e_t^{l,i} - \varepsilon/(T-1)$, then the algorithm adds the constraint, $\theta_t^i \geq E_t^{l,i}x_t^i + e_t^{l,i}$, to program (1). If program (1) has a unique optimum, this row becomes active in $\overline{Z}_t^i(x_{t-1}^{a(i)})$ and the optimal value of $\overline{Z}_t^i(x_{t-1}^{a(i)})$ increases.

Proof

The optimal solution of $(\overline{Z}_t^i)^{k_t^i}(x_{t-1}^{a(i)})$, $((x_t^i)^{k_t^i}, (\theta_t^i)^{k_t^i})$, is not feasible after the addition of this constraint. Since $\overline{Z}_t^i(x_{t-1}^{a(i)})$ is a convex program which obeys the Slater

condition, this constraint must be active. If $(\bar{Z}_t^i)^{k_t^i}(x_{t-1}^{a(i)})$ does not have alternate optima, then $(\bar{Z}_t^i)^{k_t^i+1}(x_{t-1}^{a(i)}) > (\bar{Z}_t^i)^{k_t^i}(x_{t-1}^{a(i)})$.

A similar proof shows an identical result for the addition of a feasibility cut. □

Lemma 5

Each time the algorithm proceeds from step 3 to step 4, the decision vector $(x_1^{k_1}, (x_2^1)^{k_2^1}, \ldots, (x_2^{s(2)})^{k_2^{s(2)}}, \ldots, (x_T^1)^{k_T^1}, \ldots, (x_T^{s(T)})^{k_T^{s(T)}})$ is a feasible solution to the multistage stochastic convex program.

Proof

At this point in the algorithm, for all t, i,

$$g_t((x_t^i)^{k_t^i}) \geq 0,$$

$$B_{t-1}(x_{t-1}^{a(i)})^{k_{t-1}^{a(i)}} + A_t(x_t^i)^{k_t^i} \leq (C_t(\xi_t^i))(x_{t-1}^{a(i)})^{k_{t-1}^{a(i)}},$$

$$(x_t^i) \geq 0.$$

Thus, each $(x_t^i)^{k_t^i}$ is feasible. □

Lemma 6

At each point at which the algorithm proceeds from step 3 to step 4, the following inequalities hold:

$$\sum_{t=1}^{T} \sum_{i=1}^{s(t)} p_t^i * p_{t-1}^{a(i)} * p_{t-2}^{a(a(i))} * \cdots * p_1 * f_t((x_t^i)^{k_t^i}) \geq Z_1 \geq \bar{Z}_1. \tag{11}$$

Proof

As already established in the preceding lemma, $(x_1^{k_1}, (x_2^1)^{k_2^1}, \ldots, (x_2^{s(2)})^{k_2^{s(2)}}, \ldots, (x_T^1)^{k_T^1}, \ldots, (x_T^{s(T)})^{k_T^{s(T)}})$ is feasible at this point in the algorithm. As a result, the objective function value corresponding to it is greater than or equal to the optimal solution to the problem. Also, as we have shown already, the expression $\bar{Z}_1$ is always less than or equal to the optimal objective value solution to the problem. □

Lemma 7

For $t = 1, \ldots, T-1$ and for all i, there exist dual multipliers, $(\mu_t^{i,l})^{k_t^i}$, such that

$$(\theta_t^i)^{k_t^i} = \sum_{l=1}^{K_t^i} (\mu_t^{i,l})^{k_t^i} [E_t^{l,i}(x_t^i)^{k_t^i} + e_t^{l,i}].$$

Proof
The result follows directly from the sufficient optimality condition in (1) and noting that some cut l must be active to achieve a finite optimum. As an alternative, consider the following LP which returns the value of $(\theta_t^i)^{k_t^i}$ for a given value of $(x_t^i)^{k_t^i}$:

$$\begin{aligned} &\text{minimize} && \theta_t^i \\ &\text{subject to} && \mu_t^{l,i}: E_t^{l,i}(x_t^i)^{k_t^i} + e_t^{l,i} \le \theta_t^i, \quad l = 1, \ldots, K_t^i. \end{aligned}$$

$(\theta_t^i)^{k_t^i}$ is the optimal solution to the above problem with the dual variables $(\mu_t^{i,l})^{k_t^i}$. Duality implies that

$$(\theta_t^i)^{k_t^i} = \sum_{l=1}^{K_t^i} (\mu_t^{i,l})^{k_t^i} [E_t^{l,i}(x_t^i)^{k_t^i} + e_t^{l,i}].$$

Recall that the convex stochastic nonlinear program that we are bounding below is itself a bounded program. That is, each variable in the problem can be considered to lie within a compact set:

$$x_1 \in S_1, x_2^1 \in S_2^1, \ldots, x_2^{s(2)} \in S_2^{s(2)}, \ldots, x_T^1 \in S_T^1, \ldots, x_T^{s(T)} \in S_T^{s(T)}.$$

Also recall that the stochastic ETA-MACRO model has a feasible interior point solution. □

Theorem 1
Under the assumptions given above, for any choice of $\varepsilon > 0$ and $\delta > 0$, the NDSNPA algorithm terminates in a finite number of iterations.

Proof
For the algorithm not to terminate, there must exist some stage and scenario for which the condition

$$D_t^{k_t^i,i}(x_t^i)^{k_t^i} + d_t^{k_t^i,i} \ge -\delta \tag{12}$$

never holds (i.e., the forward pass does not terminate in a finite number of iterations) or there exist some stage and scenario for which the condition

$$(\theta_t^i)^{k_t^i} \ge E_t^{k_t^i,i}(x_t^i)^{k_t^i} - e_t^{k_t^i,i} + \varepsilon/(T-1) \tag{13}$$

never holds (i.e., the backward pass does not terminate finitely) or both. We first assume that the algorithm does not terminate finitely because the forward pass never terminates for some node i in stage t. Suppose for some $d \in D_t^i$, we have

$$G_{t+1}^d((x_t^i)^k) = \text{maximize} \quad -v^+ - e^t v^-$$

subject to

$$\pi^d_{t+1}: \quad B_t(x^i_t)^k + A_{t+1}x^d_{t+1} - Iv^- \le (C_{t+1}(\xi^d_{t+1}))(x^i_t)^k,$$

$$\sigma^d_{t+1}: \quad g_{t+1}(x^d_{t+1}) + v^+ \ge 0,$$

$$\mu^{d,l}_{t+1}: \quad E^{l,d}_{t+1}x^d_{t+1} + e^{l,d}_{t+1} \le \theta^d_{t+1}, \quad l = 1,\dots,K^d_{t+1},$$

$$\lambda^{d,l}_{t+1}: \quad D^{l,d}_{t+1}x^d_{t+1} + d^{l,d}_{t+1} \ge 0, \quad l = 1,\dots,L^d_{t+1},$$

$$x^d_{t+1}, v^- \in R^{m_{t+1}}, v^+ \in R^1 \ge 0,$$

and $G^d_{t+1}((x^i_t)^k) < 0$ for the kth value of x^i_t derived in the node i and stage t optimization problem and sent to the node d problem in stage $t+1$. $k-1$ feasibility cuts have already been sent from node d in stage $t+1$ to node i in stage t. We assume that this process never stops.

The algorithm proceeds choosing points $(x^i_t)^k$ and $(x^d_{t+1})^k$ from the compact sets S^i_t and S^d_{t+1}. Let $((x^i_t)^k, (x^d_{t+1})^k) = (x^k) \in S^i_t \times S^d_{t+1}$. Note that $S^i_t \times S^d_{t+1}$ is also compact as the product of two compact sets.

Since $S^i_t \times S^d_{t+1}$ is compact, (x^k) must have a cluster point. From the continuity of problem functions and the Slater conditions, $(\pi^d_{t+1})^k$, $(\sigma^d_{t+1})^k$, $(\mu^{d,l}_{t+1})^k$, $(\lambda^{d,l}_{t+1})^k$ are contained in compact sets and have cluster points. Let these cluster points be x^∞, $(\pi^d_{t+1})^\infty$, $(\sigma^d_{t+1})^\infty$, $(\mu^{d,l}_{t+1})^\infty$, $(\lambda^{d,l}_{t+1})^\infty$.

We know from a previous lemma that

$$(D^i_t)^k((\pi^d_{t+1})^k, (\sigma^d_{t+1})^k, (\mu^{d,l}_{t+1})^k, (\lambda^{d,l}_{t+1})^k)(x^i_t)^k$$
$$+ (d^i_t)^k((\pi^d_{t+1})^k, (\sigma^d_{t+1})^k, (\mu^{d,l}_{t+1})^k, (\lambda^{d,l}_{t+1})^k) = 0$$

since $(D^i_t)^k x^i_t + (d^i_t)^k \ge 0$ is the last feasibility constraint to have been added to the $\bar{Z}^i_t$ program. Since $G^d_{t+1}((x^i_t)^k) < 0$, this implies that

$$0 = (D^i_t)^k(x^i_t)^k + (d^i_t)^k \ge G^d_{t+1}((x^i_t)^k) = (D^i_t)^{k+1}(x^i_t)^k + (d^i_t)^{k+1},$$

which implies that

$$0 \ge G^d_{t+1}((x^i_t)^k) = ((D^i_t)^{k+1}(x^i_t)^k + (d^i_t)^{k+1}) - ((D^i_t)^k(x^i_t)^k + (d^i_t)^k)$$
$$= ((D^i_t)^{k+1} - (D^i_t)^k)(x^i_t)^k + ((d^i_t)^{k+1} - (d^i_t)^k).$$

Passing to the limit on k, the term on the right-hand side vanishes, implying that x^∞ is feasible for node d. This also implies that there exists some finite k such that

$$((D_t^i)^{k+1}(x_t^i)^k + (d_t^i)^{k+1}) - ((D_t^i)^k(x_t^i)^k + (d_t^i)^k) \geq -\delta,$$

$$((D_t^i)^{k+1}(x_t^i)^k + (d_t^i)^{k+1}) \geq -\delta,$$

$$\delta + ((D_t^i)^{k+1}(x_t^i)^k + (d_t^i)^{k+1}) \geq 0.$$

Thus, we have shown that, in fact, this condition is met after a finite number of iterations. Hence, our assumption that this never occurs is contradicted. Thus, the forward pass of the algorithm will terminate in a finite number of iterations.

We must now show that the backward pass of this algorithm, likewise, terminates in a finite number of iterations. Assume first that for some node i in stage $T-1$, the condition for termination,

$$(\theta_{T-1}^i)^{k_i^i} \geq E_{T-1}^{l,i}(x_{T-1}^i)^{k_{T-1}^i} - e_{T-1}^{l,i} + \varepsilon/(T-1),$$

is never met in the $\overline{Z}_{T-1}^i(x_{T-2}^{a(i)})$ problem for any $x_{T-2}^{a(i)}$.

We know from a previous theorem that

$$\sum_{t=1}^{T}\sum_{i=1}^{s(t)} p_t^i * p_{t-1}^{a(i)} * p_{t-2}^{a(a(i))} * \cdots * p_1 * f_t((x_t^i)^{k_t^i}) \geq Z_1 \geq \overline{Z}_1, \tag{14}$$

and thus, that

$$\begin{aligned} f_{T-1}((x_{T-1}^i)^{k_{T-1}^i}) + \sum_{d\in D_{T-1}^i} p_T^d f_T((x_T^d)^{k_T^d}) &\geq Z_{T-1}^i(x_{T-2}^{a(i)}) \\ &\geq \overline{Z}_{T-1}^i(x_{T-2}^{a(i)}) \\ &= f_{T-1}((x_{T-1}^i)^{k_{T-1}^i}) + (\theta_{T-1}^i)^{k_{T-1}^i}. \end{aligned}$$

As before, since the iteration points $\{x^k\} = \{(x_{T-1}^i)^k, \{(x_T^d)^k, d \in D_{T-1}^i\}\} \in S_{T-1}^i \times \prod_{d\in D_{T-1}^i} S_T^d$, compact set, $\{x^k\}$ has a cluster point.

The continuity of the problem functions and Slater conditions imply that

$$\{(\sigma_{T-1}^i)^k\},\ \{(\pi_{T-1}^i)^k\},\ \{(\mu_{T-1}^{i,l})^k\},\ \{(\lambda_{T-1}^{i,l})^k\},$$

$$\{(\sigma_T^d)^k, d \in D_{T-1}^i\},\ \{(\pi_T^d)^k, d \in D_{T-1}^i\},$$

$$\{(\mu_T^{d,l})^k, d \in D_{T-1}^i\},\ \{(\lambda_T^{d,l})^k, d \in D_{T-1}^1\}$$

are contained in compact sets and have cluster points. Let these cluster points be

$$(\sigma_{T-1}^i)^\infty,\ (\pi_{T-1}^i)^\infty,\ (\mu_{T-1}^{i,l})^\infty,\ (\lambda_{T-1}^{i,l})^\infty,$$

$$(\sigma_T^d)^\infty, d \in D_{T-1}^i\},\ (\pi_T^d)^\infty, d \in D_{T-1}^i,$$

$$(\mu_T^{d,l})^\infty, d \in D_{T-1}^i,\ (\lambda_T^{d,l})^\infty, d \in D_{T-1}^1.$$

We then have

$$f_{T-1}((x^i_{T-1})^k) + \sum_{d \in D^i_{T-1}} p^d_T f_T((x^d_T)^k) \geq Z^i_{T-1}(x^{a(i)}_{T-2})$$

$$\geq f_{T-1}((x^i_{T-1})^k) + (\theta^i_{T-1})^k$$

$$= f_{T-1}((x^i_{T-1})^k) + \sum_{l=1}^{K^i_{T-1}} (\mu^{i,l}_{T-1})^k [E^{l,i}_{T-1}(x^i_{T-1})^k + e^{l,i}_{T-1}]$$

$$= f_{T-1}((x^i_{T-1})^k) + \sum_{(\mu^{i,l}_{T-1})^k > 0} (\mu^{i,l}_{T-1})^k [E^{l,i}_{T-1}(x^i_{T-1})^k + e^{l,i}_{T-1}]$$

$$= f_{T-1}((x^i_{T-1})^k) + [E^{l,i}_{T-1}(x^i_{T-1})^k + e^{l,i}_{T-1}]$$

$$= f_{T-1}((x^i_{T-1})^k) + [E^{k,i}_{T-1}(x^i_{T-1})^k + e^{k,i}_{T-1}]$$

(since the last cut added must be active by the previous lemma)

$$= f_{T-1}((x^i_{T-1})^k) + \sum_{d \in D^i_{T-1}} p^d_T f_T((x^d_T)^k) + [E^{k,i}_{T-1}(x^i_{T-1})^k + e^{k,i}_{T-1}]$$

$$- [E^{k+1,i}_{T-1}(x^i_{T-1})^k + e^{k+1,i}_{T-1}]$$

$$= f_{T-1}((x^i_{T-1})^k) + \sum_{d \in D^i_{T-1}} p^d_T f_T((x^d_T)^k)$$

$$+ [(E^{k,i}_{T-1} - E^{k+1,i}_{T-1})(x^i_{T-1})^k + (e^{k,i}_{T-1} - e^{k+1,i}_{T-1})].$$

Passing to the limit on k, the last two terms vanish, implying that x^∞ is optimal for $Z^i_{T-1}(x^{a(i)}_{T-2})$. This also implies that after a finite number of iterations,

$$[E^{k,i}_{T-1}(x^i_{T-1})^k + e^{k,i}_{T-1}] - [E^{k+1,i}_{T-1}(x^i_{T-1})^k + e^{k+1,i}_{T-1}] \geq -\varepsilon/(T-1) \rightarrow$$

$$(\theta^i_{T-1})^k - [E^{k+1,i}_{T-1}(x^i_{T-1})^k + e^{k+1,i}_{T-1}] \geq -\varepsilon/(T-1) \rightarrow$$

$$(\theta^i_{T-1})^k \geq [E^{k+1,i}_{T-1}(x^i_{T-1})^k + e^{k+1,i}_{T-1}] - \varepsilon/(T-1).$$

As a result, our assumption about not meeting the condition for termination is false. This portion of the algorithm will terminate finitely. Proceeding recursively as before, we can see that for all i in each stage $t < T-1$ the algorithm reaches a point where

$$(\theta^i_t)^k \geq [E^{k+1,i}_t (x^i_t)^k + e^{k+1,i}_t] - \varepsilon/(T-1)$$

in a finite number of iterations for all $x^{a(i)}_{t-1}$.

Thus, both the forward and backward portions of the algorithm terminate in a finite number of iterations implying that the entire algorithm terminates finitely. □

4 A lower bounding approximation

In section 3, we gave an algorithm for a finite realization version of CNSP that yields a solution with given ε and δ within a finite number of iterations. In general, the finite realization is checked for accuracy in comparison to an upper bounding approximation (see Birge and Wets [4]). When the gap between the two approximations is too large, the lower bounding approximation is improved and the solution process is repeated.

Since the cuts generated in one iteration of NDSNPA are all valid lower bounds on the value functions, these cuts are lower bounds on any further refining approximation as long as the refinements are nested (e.g., $z_t^k \leq z_t^{k+1}$ from refinement k to $k+1$). In fact, this is generally the case. With this, we can simply start with all previous subproblem cuts when progressing from one refinement to the next.

In the following theorem, we show that a particular lower bounding scheme of taking conditional expectations over regions of a continuous random vector does indeed yield a lower bound in this example. We then achieve the proper nested bounding results by simply ensuring that all current regions for conditional expectations are maintained in the next refinement.

Theorem 2
A discretization of the distribution over the conditional means of a partition of Ξ_t where each conditional mean is weighted by the conditional probability of each partition results in a lower bound on the optimal value of CNSP.

Proof
For all t,

$$z_t(x_{t-1}) = \int_{\xi_t \in \Xi_t} Z_t(x_{t-1}, \xi_t) dF(\xi_t) d\xi_t$$

$$= \sum_{i=1}^{K} \int_{\xi_t \in \Xi_t^i} Z_t(x_{t-1}, \xi_t) dF(\xi_t) d\xi_t$$

where Ξ_t is partitioned into $\{\Xi_t^1, \ldots, \Xi_t^K\}$,

$$= \sum_{i=1}^{K} \int_{\xi_t \in \Xi_t^i} Z_t(x_{t-1}, \xi_t) dF(\xi_t) d\xi_t \int_{\xi_t \in \Xi_t^i} dF(\xi_t) d\xi_t \Big/ \int_{\xi_t \in \Xi_t^i} dF(\xi_t) d\xi_t$$

$$\geq \sum_{i=1}^{K} Z_t(x_{t-1}, \int_{\xi_t \in \Xi_t^i} \xi_t dF(\xi_t) d\xi_t) P(\xi_t \in \Xi_t^i),$$

as $Z_t(x_{t-1}, \xi_t)$ is convex in ξ_t

$$= \sum_{i=1}^{K} Z_t(x_{t-1}, E_\xi[\xi_t | \xi_t \in \Xi_t^i]) P(\xi_t \in \Xi_t^i).$$

The result follows if $\sum_{d \in D_t^i} p_{t+1}^d Z_{t+1}^d(x_t^i) \leq Z_{t+1}(x_t^i)$ for all t, i and x_t^i where p_t^i is the conditional probability that scenario i in period t will occur given that scenario $a(i)$ (the predecessor node of node i in period t) in period $t-1$ has occurred and D_t^i is the set of descendant nodes of node i. The result holds for $t = T-1$ as in lemma 1. Using induction on t, assume the result for $t > t'-1$. To show the result for t', note that $\sum_{d \in D_t^i} p_{t'+1}^d Z_{t'+1}^d(x_{t'}^i) \leq Z_{t'+1}(x_{t'}^i)$, which is independent of i by assumption.

With this result and an upper bound [17, 14], we then have a procedure to achieve any desired accuracy in finite time even for continuous random variables. □

5 ETA-MACRO decomposition

Our motivation for this study is an extension of the Manne–Richels Global 2100 model. This model builds on the energy-economic model developed by Manne [8] called ETA-MACRO. The key stochastic parameter that we introduced into this model was a stochastic return on investment in various energy technologies and resources. The model with our changes highlighted as dashed lines appears in figure 1.

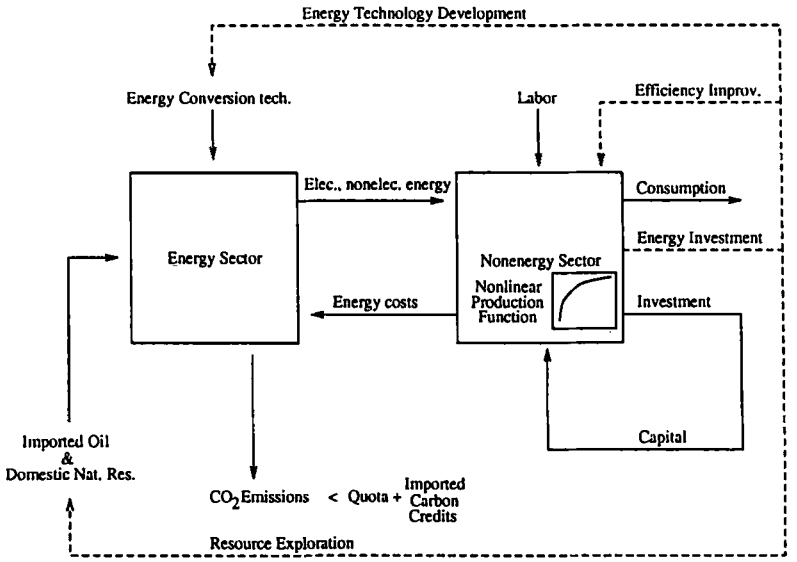

Figure 1. Stochastic Global 2100 model.

Notice in figure 1 that this model fits into the framework of CNSP because all the nonlinearity occurs within each time step and does not directly connect decision variables in one period with results in the next. Because of this, the decomposition of the problem follows directly from what was derived in the previous section.

We should note that the addition of stochasticity to the Global 2100 model did result in fundamental changes to the predicted path of economic development over the next century in the modeled region. We have chosen not to include any of those results in this paper, though, in order to devote more space to the description of the algorithm. A forthcoming paper [3] contains a more lengthy description of the model and added stochasticity, and an analysis of the results from the study.

In the next section we will explore some of the computational results we obtained when we implemented our decomposition algorithm in parallel on a network of workstations.

6 Parallel lower bounding

Parallelization of decomposition algorithms allows modelers to solve problems whose size made them previously unsolvable. In this section, we first investigate how the parallel lower bounding NDSNPA algorithm is implemented. We then present results comparing solution time requirements for four differently sized versions of the stochastic economic model when solved by the parallel cutting plane algorithm and the projected augmented Lagrangian algorithm (the solution algorithm used by MINOS [10]). This will give an indication of the value of the algorithm and when it should be used.

6.1 *Parallel implementation*

The original Global 2100 deterministic model is a large nonlinear program with a total of 493 variables (99 nonlinear) and 390 constraints (23 nonlinear) spread out over the eleven periods of the study horizon. The original model was provided by the Electric Power Research Institute. The model is written in GAMS [5] and runs in conjunction with the solver MINOS. Because AMPL [7] is the modeling language available on the University of Michigan Computer Aided Engineering Network (CAEN), we translated the GAMS file into AMPL. The translation was validated by ensuring that output from both the GAMS and AMPL models was identical.

To make the model stochastic, we introduced three new types of investment whose return affects the period just subsequent to the period of expenditure. The rates of return on these investment expenditures are uncertain. The energy technology investment expenditures affect the growth in availability of the two generic electrical technologies (ADV-HC, ADV-LC) and the three nonelectrical technologies (Renewable and Synthetic fuels and the nonelectric backstop fuel). The resource exploration expenditures add to the stock of two types of fuels (oil and gas), each having two

grades (high cost and low cost). Finally, energy efficiency investment expenditures contribute to increasing the overall efficiency with which energy resources are converted from input to output in the Cobb–Douglas production function. The rates of return for each scenario of the four scenario case of the model are included in table 1.[3)]

Table 1

Returns on investment.

Technology name	Scen 1	Scen 2	Scen 3	Scen 4
ADV-HC	0.094	0.33	3.3	10.328
ADV-LC	0.115	0.4	4.0	12.6492
OIL-LC	7	10	25	48
GAS-LC	7	10	25	48
OIL-HC	7	10	25	48
GAS-HC	7	10	25	48
RNEW	0.89	3.1	30.9	97.98
SYNF	2.36	8.17	81.7	258.2
NE-BAK	0.632	2.2	21.9	69.3
EFFICIENCY	0.06	0.12	0.15	0.17

After having developed the model, we developed the computer code and data structures necessary to run the decomposition algorithm. To ensure flexibility in the model, we decided to use AMPL as a modeling language wherever possible. AMPL is not only flexible but, like GAMS, has the benefit of modeling nonlinearity so that less programming effort is required (i.e., writing special FORTRAN code for the calculation of gradient and Hessian information).

From the AMPL model file that describes the original deterministic problem with multiple periods, we developed a generic single period AMPL model file. A C shell program uses this generic AMPL model file as a template to create a model file for every node in the stochastic tree. These model files for each node are used by the decomposition code.

We wrote the decomposition algorithm itself in C. We used the package of C subroutines, PVM, to run the algorithm in parallel on a network of IBM RISC 6000 workstations. We designed the decomposition algorithm so that each processor solves a subtree of the entire stochastic tree as in the four scenario case illustrated in figure 2. The PVM subroutines make the passing of data between subtrees on separate processors possible.

[3)] Nonelectrical technologies: EXAJ/10^{12} \$. Electrical technologies: TKWH/10^{12} \$. Efficiency: (Percentage reduction in energy intensity)/10^{12} \$.

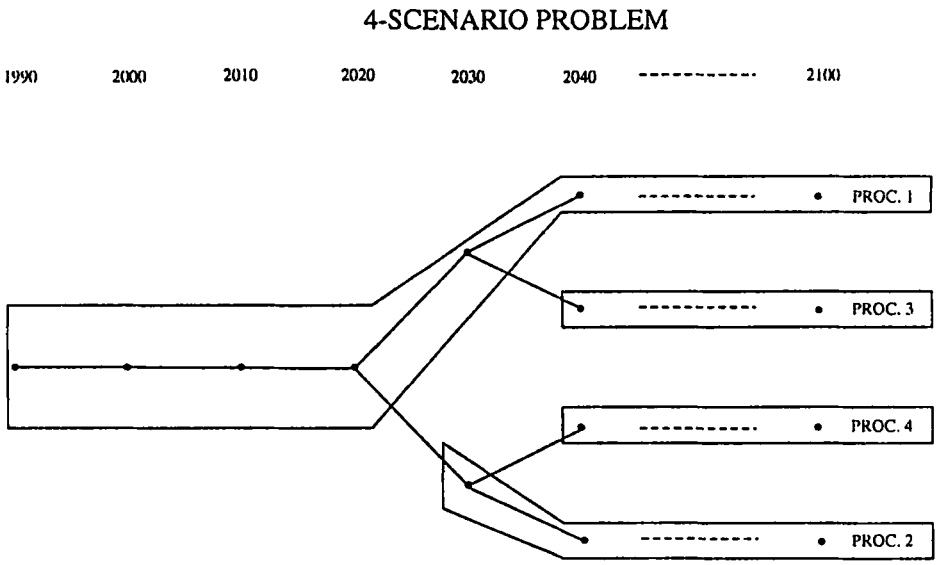

Figure 2. Processor assignment of nodes.

Each subtree processor maintains an array of strings containing the entire model file of each node problem in its subtree. It reads these in at the beginning of the algorithm. Also at the beginning of the algorithm as each subtree processor begins, the C program starts AMPL as a forked child process and then creates communication pipes between the parent process and AMPL. In this way, the parent process can send AMPL the appropriate node model (in the form of a large string) to send to a solver. AMPL is then able to pass back the resulting variable and cut information to the parent process. The parent process can then pass this on to which ever node or nodes have need of it at this particular place in the algorithm and pass a message back to AMPL resetting it for the next model. Results of the implementation follow in the next section.

6.2 *Parallel computational results*

In the original continuous problem, each return on investment, starting with 2030, is modeled as a uniformly distributed random variable between the upper and lower bounds of the first and fourth scenarios from the discretized problem. That is, the values of the payoffs from these two scenarios provide lower and upper bounds for each payoff. We chose the uniform distribution to reflect the great uncertainty regarding investment returns in the future.

Using the parallel implementation of the lower bounding decomposition method, we solved the stochastic program for varying numbers of scenarios. The results of these experiments are listed in table 2.[4)]

[4)] Stopping criterion: $\varepsilon = 0.01$.

Table 2

Parallel performance.

	32 scenarios	16 scenarios	8 scenarios	4 scenarios
Nodes	161	97	57	33
Rows	54946	27474	13738	6870
Nonlinear constraints	1409	705	353	177
Columns	23809	11905	5953	2977
Nonlinear variables	4224	2112	1056	528
Matrix elements	112714	54250	26010	12386
Projected Augmented Lagrangian Algorithm	21219 sec	5521	1436	423 sec
Parallel computation	11314 sec	6180	3526	2726 sec
Number of processors	8	8	8	4
Solution	58744.38	58744.934	58745.100	58745.605

We analyzed four problems with 4, 8, 16 and 32 scenarios respectively. This corresponds to 33, 57, 97 and 161 nodes in the stochastic tree. With regard to the relative sizes of the problems, each doubling in the number of scenarios is accompanied by an approximate doubling in most of the key characteristics of the problems (e.g., number of rows, matrix elements).

The quality of the bound varies little between the different problems. In fact, the largest lower bound differs from the smallest lower bound by only 0.002%. This seems to be due to the fact that, as we increased the number of scenarios, we did not substantially increase the coverage of the random space. We only added detail to regions of the space that were already well approximated by the existing scenarios.

Although for the first three problems the Projected Augmented Lagrangian Algorithm (MINOS) solves the problem more quickly than the parallel implementation of the NDSNPA algorithm, the rate at which the solution times increase for the parallel method is slower than that for MINOS. Once the number of scenarios is increased to 32, MINOS solves the problem more slowly than the parallel code. For nonlinear convex problems of this size and larger, it appears that the NDSNPA algorithm, preferably implemented in parallel, is the method of choice.

The computational results from this section indicate that parallel algorithms based on decomposition techniques hold great promise as a means for solving previously unsolvable models in a timely fashion. Continued work with specialized parallel systems having faster network speed than the network linking the RISC 6000 processors used in this study should make this even more apparent.

7 Conclusion

This paper developed a decomposition method for a stochastic nonlinear program in which nonlinearities appear within a period only and in which the random variables affect period to period transitions linearly. We showed how this method achieves any desired level of accuracy in a finite number of steps. We then showed how increasingly accurate refinements of the random variables could be efficiently incorporated into the method without complete restart of the optimization.

Our application to a practical economic modeling problem demonstrated significant savings over general purpose methods. Parallel computation also yields high efficiencies due to the separation among all subproblems. The method appears quite applicable to numerous other applications. More general nonlinear forms would, however, require additional consideration to achieve the lower supporting cuts that form the basis of this method.

Acknowledgements

The comments and suggestions of Professor Stein W. Wallace (University of Trondheim) and three anonymous referees are gratefully acknowledged.

References

[1] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerical Mathematics 4, 1962, 238–252.

[2] J.R. Birge, Decomposition and partitioning methods for multi-stage stochastic linear programs, Operations Research 33, 1985, 989–1007.

[3] J.R. Birge and C.H. Rosa, Modeling investment uncertainty in the costs of global CO_2 emission policy, European Journal of Operations Research, to be published.

[4] J.R. Birge and R. J-B Wets, Designing approximation schemes for stochastic optimization problems, in particular, for stochastic programs with recourse, Mathematical Programming Study 27, 1986, 54–102.

[5] A. Brooke, D. Kendrick and A. Meeraus, *GAMS: A User's Guide*, The Scientific Press, San Francisco, CA, 1992.

[6] G. B. Dantzig and A. Madansky, On the solution of two-stage linear programs under uncertainty, in *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*, J. Neyman, ed., University of California Press, Berkeley, 1961,pp. 165–176.

[7] R. Fourer, M.G. Gay and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, San Francisco, CA, 1993.

[8] A.S. Manne, ETA-MACRO: a user's guide, EPRI Interim Report, Project 1014, 1981.

[9] A.S. Manne and R.G. Richels, Global CO_2 emission reductions – the impacts of rising energy costs, The Energy Journal 12, 1991, 87–107.
[10] B.A. Murtagh and M.A. Saunders, MINOS 5.1 user's guide, report SOL-83-20R, Systems Optimization Library, Stanford, CA, 1987.
[11] M.C. Noël and Y. Smeers, Nested decomposition of multistage nonlinear programs with recourse, Mathematical Programming 37, 1987, 131–152.
[12] R.P. O'Neill, Nested decomposition of multistage convex programs, SIAM Journal on Control and Optimization 14, 1976, 409–418.
[13] M.V.E. Pereira and L.M.V.G. Pinto, Multi-stage stochastic optimization applied to energy planning, Mathematical Programming 52, 1991, 359–375.
[14] C.H. Rosa, Modeling investment uncertainty in the costs of global CO_2 emission policy, Department of Industrial and Operations Engineering, University of Michigan, Ph.D. Thesis, Sept. 1993.
[15] R. Van Slyke and R.J-B Wets, L-shaped linear programs with application to optimal control and stochastic programming, SIAM Journal on Applied Mathematics 17, 1969, 638–663.
[16] S.A. Vejtasa and B.L. Schumann, Technology data for carbon dioxide emission model: Global 2100, SFA Pacific Inc., Mountain View, CA, 1989.
[17] S.W. Wallace and T.C. Yan, Bounding multi-stage stochastic programs from above, Mathematical Programming 61, 1993, 111–129.