SOLVING MULTISTAGE STOCHASTIC PROGRAMS WITH TREE DISSECTION

ADAM BERGER, JOHN MULVEY, ED ROTHBERG, AND ROBERT VANDERBEI

ABSTRACT. One component of every multi-stage stochastic program is a filtration that determines the notion of which random events are observable at each stage of the evolution. Within the context of interior-point methods, we describe an efficient pre-ordering technique, called filtered dissection, that takes advantage of the filtration's structure to dramatically reduce fill-in in the factorization as compared with methods such as the default methods employed by CPLEX-barrier and LOQO. We have implemented this technique as a minor modification to LOQO, and it produces a roughly 200-fold performance improvement. In particular, we have solved a previously-unsolvable, real-world, 6-stage financial investment problem having 800K equations and 1,200K variables (and 8,192 points in its sample space) using a single processor SGI workstation.
The filtered dissection algorithm applies in a natural manner to generic (linear and convex) multi-stage stochastic programs. The approach promises to eliminate the need for decomposition algorithms for these classes of applications.

1. INTRODUCTION.

We are interested in multistage nonlinear stochastic programs. There are several components to the definition of such problems. First, we assume the existence of a *sample space* Ω on which is defined a *filtration* $\mathcal{F} = \{\mathcal{F}_t : t = 0, 1, \ldots, T\}$. That is, $\mathcal{F}$ consists of an increasing sequence of σ-algebras on Ω. Each σ-algebra $\mathcal{F}_t$ consists of those events that are "known" at time t. Since we wish to consider stochastic programs for which practical algorithms can be developed, we shall assume that Ω is a finite set and that the *time horizon* T is finite. In such settings, every σ-algebra is finitely generated. That is, for each σ-algebra there exists a finite partition of Ω such that every set in the σ-algebra is a union of sets in this finite partition.

1991 *Mathematics Subject Classification.* Primary ??, Secondary ??
Second and Third authors' research supported by AFOSR through grant AFOSR-91-0359.

A random variable, say Z, is simply a function defined on Ω: $Z = \{Z(\omega) : \omega \in \Omega\}$. If the function takes its values in a vector space, then it is called a *random vector*. Similarly, if it takes its values in a space of matrices, then it is called a *random matrix*. As is customary, given a random variable Z and a σ-algebra, say $\mathcal{F}_t$, we write $Z \in \mathcal{F}_t$ to indicate that Z is $\mathcal{F}_t$-measurable. Since $\mathcal{F}_t$ is finitely generated, Z is $\mathcal{F}_t$-measurable if and only if it is constant on each of the sets in the partition that generates $\mathcal{F}$.

Since Ω is finite, it is natural to introduce the "master" σ-algebra consisting of the collection of all subsets of Ω (generated by the partition consisting of the individual points of Ω). We assume that the sample space together with this master σ-algebra is also endowed with a probability measure which we denote by $\mathbf{P}$. The corresponding expectation operator shall be denoted, as usual, by $\mathbf{E}$. For each $\omega \in \Omega$, we let $p(\omega)$ denote the probability of the one-point set consisting of ω alone:

$$p(\omega) = \mathbf{P}(\{\omega\}).$$

The function p so defined is called the *probability mass function* associated with $\mathbf{P}$. Expectations are easily written in terms of this function:

$$\mathbf{E}Z = \sum_{\omega \in \Omega} Z(\omega)p(\omega).$$

Without loss of generality, we shall always assume that the probability mass function is strictly positive on Ω (otherwise points from Ω could simply be deleted).

Next, we fix an integer m and, for each $t = 0, 1, \ldots, T$, we assume that we are given a random matrix A_t having m rows and n_t columns. Also given is a random m-vector b.[1] The set of *decision variables* for the stochastic program consists of a sequence of random vectors of the appropriate dimension:

$$x = \{x_t \in \mathbb{R}^{n_t} : t = 0, 1, \ldots, T\}.$$

Let $n = n_0 + n_1 + \cdots + n_T$ and let f be a concave function defined on $\mathbb{R}^n$. The function f shall be used to define our *objective function*. It is taken to be concave to allow for the modeling of risk averse behavior. Our *multistage nonlinear stochastic program* can now be defined as:

(1.1)

$$\begin{array}{lrll} \text{maximize} & \mathbf{E}f(x) & & \\ \text{subject to} & \sum_{t=0}^{T} A_t(\omega)x_t(\omega) = b(\omega), & \omega \in \Omega, & \\ & x_t(\omega) \geq 0, & t = 0, 1, \ldots, T, & \omega \in \Omega, \\ & x_t \in \mathcal{F}_t, & t = 0, 1, \ldots, T. & \end{array}$$

[1]Note that, to be consistent with notational conventions in the optimization community, we allow lower case letters to denote random variables even though such a convention is uncommon in the stochastic processes community.

The constraints from the last group in (1.1) are called *non-anticipativity* constraints. These constraints have the obvious interpretation that decisions made today cannot depend on information received tomorrow (or any day thereafter).

The multistage nonlinear stochastic program defined by (1.2) fits the framework of robust optimization [] and depicts a special case of multi-stage stochastic programs []. A wide variety of real-world planning problems fit this model. Some typical examples inlcude: financial planning to maximize investor wealth over a fixed horizon [], design of telecommunication facilities to achieve a robust system in the face of stochastic demands and possible equipment failure [], aircraft scheduling to meet uncertain demand [], and assignment of production levels to global facilities in order to minimize uncertain currency costs []. In each case, the model quickly becomes large as the number of decision stages T and the cardinality of the sample space Ω increase. As such, multi-stage stochastic programs often require supercomputer resources in order to find an optimal solution.

As mentioned above, for each t, the σ-algebra $\mathcal{F}_t$ is finitely generated which means that there is a partition, call it F_t, of Ω into sets $F_t^1, F_t^2, \ldots, F_t^{J_t}$ such that each set in $\mathcal{F}_t$ is a union of sets from this partition. Furthermore, a random variable is $\mathcal{F}_t$ measurable if and only if it is constant on each of the generating sets. Hence, the non-anticipativity constraints in (1.1) can be rewritten as

$$x_t \text{ is constant on } F_t^j, \quad j = 1, 2, \ldots, J_t, \quad t = 0, 1, \ldots, T. \tag{1.2}$$

Since the σ-algebras in a filtration are increasing, it follows that each F_t^j is contained in some F_{t-1}^k. We say that F_t^j is a *descendent* of F_{t-1}^k if $F_t^j \subset F_{t-1}^k$. As shown in Figure 1, whenever $\mathcal{F}_0$ is trivial (i.e., has just one generator: $F_0^1 = \Omega$), this notion of descendency defines a tree structure on the generating sets (in general it defines a forest). This tree is called a *scenario tree*.

The constraint that x_t is constant on F_t^j can be expressed many ways. The most efficient is to enumerate the sample points belonging to F_t^j, say $\{\omega_1, \omega_2, \ldots, \omega_{K_{t,j}}\}$, and simply to equate the values of x_t from one point to the next:

$$x_t(\omega_1) = x_t(\omega_2) = \cdots = x_t(\omega_{K_{t,j}}).$$

For example, in Figure 1 the constraint that x_2 is constant on F_2^4 can be expressed as $x_2(\omega_5) = x_2(\omega_6) = x_2(\omega_7)$.

1.1. Compact vs. Expanded Representation. The problem defined by (1.1) is called the *expanded representation*. The so-called *compact representation* is obtained by introducing just a single decision vector for each generator

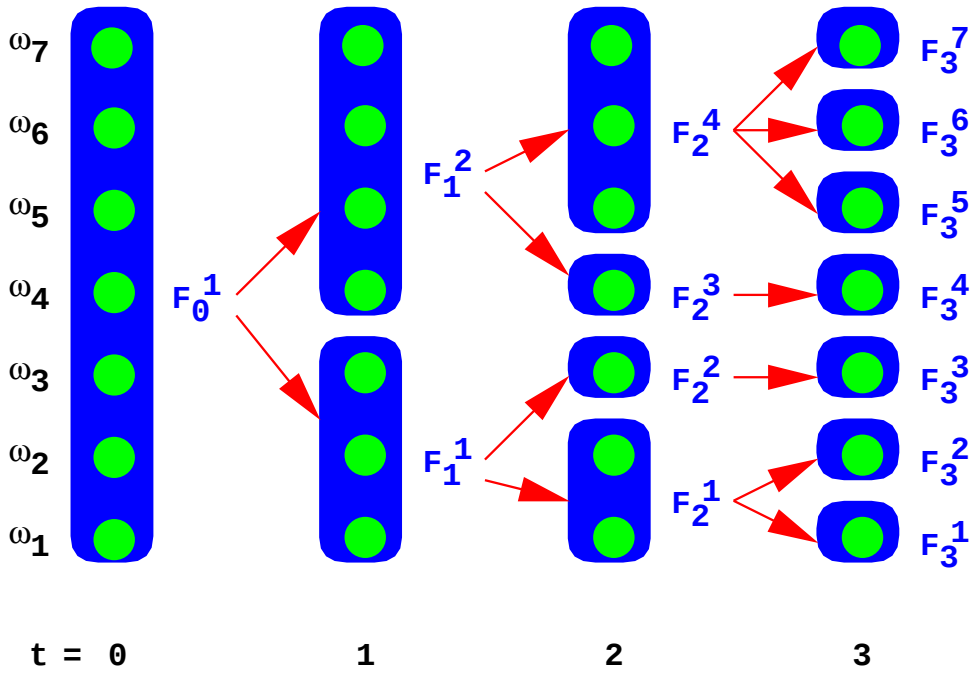

FIGURE 1. A typical scenario tree. Each filled circle represents a decision variable. Those decision variables enveloped within one darkened area must be equal to each other.

set F_t^j, call it x_t^j, and then dropping the nonanticipativity contraints:

$$\begin{array}{lll} \text{maximize} & \sum_{\omega\in\Omega} f(x(\omega)) & \\ \text{subject to} & \sum_{t=0}^{T} A_t(\omega) x_t^{j_t(\omega)} = b(\omega), & \omega \in \Omega, \\ & x_t^j \geq 0, \quad & j = 1, 2, \ldots, J_t, \\ & & t = 0, 1, \ldots, T, \end{array} \tag{1.3}$$

where $j_t(\omega)$ denotes that index j for which $\omega \in F_t^j$ and

$$x(\omega) = (x_0^{j_0(\omega)}, x_1^{j_1(\omega)}, \ldots, x_T^{j_T(\omega)}).$$

As its name implies, the compact representation has many fewer variables and constraints than the expanded representation. However, it is well known that, for large sparse optimization problems, structure determines algorithm efficiency more than overall size parameters such as numbers of variables and constraints. Such is the case in the present context in that interior-point methods (described in the next section) applied to these problems perform just as well on the expanded representation as on the compact one (in some cases even better; see, e.g., []). Hence, we shall only consider the expanded representation from this point onward.

1.2. Expression in Matrix Form. For the example shown in Figure 1, the equality constraints can be written in the following block matrix form:

$$\left[\begin{array}{c|c|c|c|c|c|c}
A_0\,A_1\,A_2\,A_3 & & & & & & \\
 & A_0\,A_1\,A_2\,A_3 & & & & & \\
 & & A_0\,A_1\,A_2\,A_3 & & & & \\
 & & & A_0\,A_1\,A_2\,A_3 & & & \\
 & & & & A_0\,A_1\,A_2\,A_3 & & \\
 & & & & & A_0\,A_1\,A_2\,A_3 & \\
 & & & & & & A_0\,A_1\,A_2\,A_3 \\
\hline
I & -I & & & & & \\
 & & & & I & -I & \\
 & & & & & I & -I \\
\hline
I & -I & & & & & \\
 & I & -I & & & & \\
 & & & I & -I & & \\
 & & & & I & -I & \\
 & & & & & I & -I \\
\hline
I & -I & & & & & \\
 & I & -I & & & & \\
 & & I & -I & & & \\
 & & & I & -I & & \\
 & & & & I & -I & \\
 & & & & & I & -I
\end{array}\right]
\left[\begin{array}{c}
x_0 \\ x_1 \\ x_2 \\ x_3 \\ \hline
x_0 \\ x_1 \\ x_2 \\ x_3 \\ \hline
x_0 \\ x_1 \\ x_2 \\ x_3 \\ \hline
x_0 \\ x_1 \\ x_2 \\ x_3 \\ \hline
x_0 \\ x_1 \\ x_2 \\ x_3 \\ \hline
x_0 \\ x_1 \\ x_2 \\ x_3 \\ \hline
x_0 \\ x_1 \\ x_2 \\ x_3
\end{array}\right]
=
\left[\begin{array}{c}
b \\ b \\ b \\ b \\ b \\ b \\ b \\ \hline
0 \\ 0 \\ 0 \\ \hline
0 \\ 0 \\ 0 \\ 0 \\ \hline
0 \\ 0 \\ 0 \\ 0 \\ \hline
0 \\ 0 \\ 0 \\ 0 \\ \hline
0 \\ 0
\end{array}\right].$$

Note that the matrices A_t, the vectors x_t, and the right-hand side b are all random variables and their multiple appearances in the above equations represent evaluations at the different sample points. For example, in the first row of constraints each of these random variables are evaluated at ω_1, in the second row they are evaluated at ω_2, etc. In order to define an algorithm for solving the stochastic program, we shall need notations for these block matrices and vectors. Hence, we shall let $\boldsymbol{A}$ denote the constraint matrix, $\boldsymbol{b}$ shall denote the right-hand side vector, and $\boldsymbol{x}$ shall denote the vector of decision variables.

Viewed simply as a function of the vector of decision variables, $\boldsymbol{x}$, we shall write the objective function as

$$\phi(\boldsymbol{x}) = \mathbf{E}f(x) = \sum_{\omega\in\Omega} f(x_0(\omega), \ldots, x_T(\omega)).$$

It should be noted that the objective function ϕ is partially separable. This fact will be exploited when developing efficient implementations, but can for the moment be forgotten.

In terms of our matrix notation, the multistage nonlinear stochastic program

can now be written succinctly as:

$$\begin{array}{ll} \text{maximize} & \phi(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{array} \tag{1.4}$$

In the next section, we shall present an interior-point method for solving (1.4) and indicate that an efficient implementation of this method hinges on being able to efficiently solve a large system of equations called the reduced KKT system. For numerical stability reasons, it is generally felt that direct methods provide the best approach to solving this system of equations. That is, some permutation of the system is to be factored and then solved using forward and backward substitution. The efficiency of direct methods depends critically on finding a good ordering heuristic to determine a row/column permutation which offers very little fill-in in the factorization process. These issues shall be taken up in section 3, where we shall describe our new ordering heuristic. This heuristic is a sort of generalization of the well-known nested-dissection heuristic. We refer to it as *filtered dissection*. Finally, Section 4 contains our computational results.

2. An Interior-Point Method.

As a starting point for our implementation we used LOQO. The algorithm implemented in LOQO is a one-phase primal-dual path-following method. As implemented, it operates directly on quadratic programming problems presented in the following general form:

$$\begin{array}{ll} \text{minimize} & f + \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{b} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} + \boldsymbol{r} \\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{array} \tag{2.1}$$

Here, $\boldsymbol{A}$ is an $m \times n$ matrix of coefficients, $\boldsymbol{b}$ is called the *right-hand side* (even though it appears on the left), $\boldsymbol{r}$ is the vector of *ranges* on the constraints, and $\boldsymbol{u}$ and $\boldsymbol{l}$ are the vectors of *upper bounds* and *lower bounds*, respectively, on the variables. The objective function, $f + \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$, is a quadratic function. The matrix $\boldsymbol{H}$ appearing in the quadratic term is assumed to be positive semidefinite so that the objective function is convex.

For the multi-stage nonlinear stochastic program given in (1.4), the lower bounds and ranges are zero and the upper bounds are infinite. Hence, we shall describe LOQO only as it applies to such cases. Also, the objective in (1.4) is a maximization. To convert it to a minimization we simply negate the objective function. More crucially, the objective function in (1.4) is not necessarily quadratic. We handle general convex objective functions as follows. LOQO is an algorithm that starts with an initial guess at the solution and iteratively

updates this guess until it arrives at an essentially optimal solution. At the beginning of each iteration, we approximate the general objective function by the first few terms of its Taylor series expansion to get a quadratic approximation. LOQO has built-in hooks that make it easy to modify the problem within the solution process and using these hooks we are able to handle quite easily (smooth) convex objective functions.

Before presenting the interior-point algorithm implemented in LOQO, we must first introduce artificial variables as appropriate. In the present context of vanishing lower bounds and ranges and infinite upper bounds, this amounts to rewriting the problem as follows:

$$\begin{array}{lrcl} \text{minimize} & f + \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} & & \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} - \boldsymbol{w} & = & \boldsymbol{b} \\ & \boldsymbol{w} + \boldsymbol{p} & = & \boldsymbol{0} \\ & \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{p} & \geq & \boldsymbol{0}. \end{array} \tag{2.2}$$

The reader may be wondering why we have included the second constraint. Since $\boldsymbol{w}$ and $\boldsymbol{p}$ are assumed to be nonnegative, the second constraint implies that both $\boldsymbol{w}$ and $\boldsymbol{p}$ must be zero. But, as we shall see shortly, our method is an infeasible method. This means that equality constraints are not enforced at the start – they only take effect as the algorithm approaches an optimal solution. In fact, both vectors $\boldsymbol{w}$ and $\boldsymbol{p}$ are initialized to be strictly positive and the iterations of the algorithm eventually push them close to zero (they never get quite there). By putting these positive variables into the formulation we arrive at an algorithm that is much more stable than what one would get without them. As far as we know, LOQO is the only implementation of an interior-point method that exploits this observation.

The dual of (2.2) is:

$$\begin{array}{lrcl} \text{maximize} & f + \boldsymbol{b}^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} & & \\ \text{subject to} & \boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{z} - \boldsymbol{H}\boldsymbol{x} & = & \boldsymbol{c} \\ & \boldsymbol{y} + \boldsymbol{q} - \boldsymbol{v} & = & \boldsymbol{0} \\ & \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{q} & \geq & \boldsymbol{0} \\ & \boldsymbol{y} \text{ free.} & & \end{array} \tag{2.3}$$

2.1. Central Path. In order to define the interior-point method, we must first introduce the primal-dual *central path*. We parametrize this path by a positive real parameter μ. Indeed, for each $\mu > 0$, we define the associated central-path point in primal-dual space as the unique point that simultaneously satisfies the conditions of primal feasibility, dual feasibility, and μ-complementarity. Ignoring nonnegativity (which is enforced separately), these

conditions are:

$$
\begin{aligned}
\boldsymbol{A}\boldsymbol{x} - \boldsymbol{w} &= \boldsymbol{b} \\
\boldsymbol{w} + \boldsymbol{p} &= \boldsymbol{0} \\
\\
\boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{z} - \boldsymbol{H}\boldsymbol{x} &= \boldsymbol{c} \\
\boldsymbol{y} + \boldsymbol{q} - \boldsymbol{v} &= \boldsymbol{0} \\
\\
\boldsymbol{X}\boldsymbol{Z}\boldsymbol{e} &= \mu\boldsymbol{e} \\
\boldsymbol{V}\boldsymbol{W}\boldsymbol{e} &= \mu\boldsymbol{e} \\
\boldsymbol{P}\boldsymbol{Q}\boldsymbol{e} &= \mu\boldsymbol{e}.
\end{aligned}
\tag{2.4}
$$

The last four equations are the μ-complementarity conditions. As usual, each upper case letter that appears on the left in these equations denotes the diagonal matrix having the components of the corresponding lower-case vector on its diagonal. This is a nonlinear system of $2n + 5m$ equations in $2n + 5m$ unknowns. It has a unique solution in the strict interior of the appropriate orthant in primal-dual space:

$$
\{(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{q}) : \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{q} \geq \boldsymbol{0}\}. \tag{2.5}
$$

This fact can be seen by noting that these equations are the first order optimality conditions for an associated strictly convex barrier problem (see, e.g. [13]).

As μ tends to zero, the central path converges to the optimal solution to both the primal and dual problems. A *primal-dual path-following algorithm* is defined as any iterative process that starts from a point in the strict interior of (2.5) and at each iteration estimates a value of μ representing a point on the central path that is in some sense closer to the optimal solution than the current point and then attempts to step toward this central-path point making sure that the new point remains in the strict interior of the appropriate orthant.

2.2. Apply Newton's Method to get Step Directions. Suppose for the moment that we have already decided on the target value for μ and let $(\boldsymbol{x}, \ldots, \boldsymbol{q})$ denote the current point in the strict interior of (2.5). Applying Newton's method to find a solution to (2.4), we get the following equations

for the step directions $(\Delta x, \ldots, \Delta q)$:

(2.6)
$$\begin{array}{rcll} A\Delta x - \Delta w & = & b - Ax + w & =: \ \rho \\ \Delta w + \Delta p & = & -w - p & =: \ \alpha \\ \\ A^T\Delta y + \Delta z - H\Delta x & = & c - A^T y - z + Hx & =: \ \sigma \\ -\Delta y - \Delta q + \Delta v & = & y + q - v & =: \ \beta \\ \\ X^{-1}Z\Delta x + \Delta z & = & \mu X^{-1}e - z & =: \ \gamma_z \\ V^{-1}W\Delta v + \Delta w & = & \mu V^{-1}e - w & =: \ \gamma_w \\ P^{-1}Q\Delta p + \Delta q & = & \mu P^{-1}e - q & =: \ \gamma_q, \end{array}$$

where we have introduced notations $\rho, \ldots, \gamma_q$ as shorthands for the right-hand side expressions. This system can be mostly solved by inspection yielding a much smaller system to solve. The details are given in [] so we just summarize the result here. First, we must solve

(2.7)
$$\left[\begin{array}{c|c} -(H+D) & A^T \\ \hline A & E \end{array}\right] \left[\begin{array}{c} \Delta x \\ \hline \Delta y \end{array}\right] = \left[\begin{array}{c} \sigma - X^{-1}Z\hat{\nu} \\ \hline \rho - E(\hat{\beta} - P^{-1}Q\hat{\alpha}) \end{array}\right],$$

where E and D are diagonal matrices given by

$$E = \left(VW^{-1} + P^{-1}Q\right)^{-1}$$

and

(2.8)
$$D = X^{-1}Z,$$

and $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\nu}$ are given by

$$\begin{array}{rcl} \hat{\alpha} & = & \alpha - PQ^{-1}\gamma_q \\ \hat{\beta} & = & \beta - VW^{-1}\gamma_w \\ \hat{\nu} & = & XZ^{-1}\gamma_z. \end{array}$$

Then, compute the remaining step directions sequentially as follows:

$$\begin{array}{rcl} \Delta w & = & -E(\hat{\beta} - P^{-1}Q\hat{\alpha} + \Delta y) \\ \Delta q & = & P^{-1}Q(\Delta w - \hat{\alpha}) \\ \Delta z & = & X^{-1}Z(\hat{\nu} - \Delta x) \\ \Delta p & = & PQ^{-1}(\gamma_q - \Delta q) \\ \Delta v & = & VW^{-1}(\gamma_w - \Delta w). \end{array}$$

System (2.7) is called the *reduced KKT system*. It is important to emphasize that both E and D have *strictly positive* diagonal entries. The fact that E has strictly positive diagonal entries is a direct consequence of our allowing w and p to be strictly positive and having the algorithm force these variables

toward zero as optimality is approached (as opposed to setting them to zero from the start).

2.3. Compute Step Lengths. Finally, for Newton's method the desired step length is one, but a ratio test is performed to see whether it needs to be shortened to maintain strict positivity of the nonnegative variables.

2.4. Solving the Reduced KKT System. There are many possible orders in which one can solve the reduced KKT system. For example, if we were to use the first equation in (2.7) to solve for $\boldsymbol{\Delta} x$ and then eliminate it from the second equation, the resulting system for $\boldsymbol{\Delta} y$ would involve the matrix

$$\boldsymbol{A}(\boldsymbol{D}+\boldsymbol{H})^{-1}\boldsymbol{A}^T$$

(this is the *normal-equations* approach used in OB1 [7, 8] and CPLEX-barrier). Similarly, if we were to solve the second equation for $\boldsymbol{\Delta} y$ and then eliminate this variable from the first equation, the resulting system for $\boldsymbol{\Delta} x$ would involve the matrix

$$\boldsymbol{A}^T\boldsymbol{E}^{-1}\boldsymbol{A}$$

(this is the approach advocated by the optimization group at the National Institute of Standards and Technology [1, 10]). Both of these will generally entail fill-in, which in some cases can be considerable. For example, if the Hessian $\boldsymbol{H}$ is not a diagonal matrix or if the matrix $\boldsymbol{A}$ has a dense column, the first form will suffer "catastrophic fill-in". On the other hand, if $\boldsymbol{A}$ has a dense row, the second form will be very bad. If A has both dense columns and dense rows, then both of these forms will suffer unnecessarily large amounts of fill-in and one would prefer, in that case, to work with the larger system to find a pivot order that does not generate so much fill-in. This idea was first suggested by Turner [11] and has been adopted by Saunders et.al. [6, 5, 4] and Mehrotra [2]. All three use a Bunch-Parlet factorization of the indefinite system. Solving the larger system is also the approach adopted in LOQO, but LOQO does not employ a Bunch-Parlet factorization. Instead, LOQO uses a modified Cholesky factorization code that has been altered to solve *symmetric quasidefinite systems* (see [12]). Equation (2.7) is an example of such a system. The matrices defining these systems share with symmetric semidefinite matrices the nice property that the (diagonal) pivots can be selected based only on a fill-in minimizing heuristic; i.e., without regard for the numerical values, which may only be known later.

There are two types of fill-in minimizing heuristics: myopic heuristics, such as *minimum-degree* [3], which sequentially minimize the fill-in produced in each subsequent stage of elimination and global heuristics, such as *nested-dissection* [9], which analyze the overall structure to find good orderings. In the symmetric quasidefinite system (2.7), there is an obvious global structure

that one can exploit. For example, the lower-right block is a diagonal matrix (as is the upper-left block whenever H is diagonal or absent). Hence, initial pivots selected from the lower-right block can only produce fill-in in the upper-left block. It is generally advantageous to exploit this structure. Hence, we have implemented a priority minimum-degree method. Each diagonal element is assigned a small integer representing an elimination priority. At first, pivots are selected only from the elements assigned priority zero. Within this priority class, pivots are selected according to the usual minimum degree heuristic. Only after all priority zero pivots have been eliminated do we proceed to the priority one elements. Again, within this priority class, the elimination order is determined using the minimum-degree heuristic. This process is continued through each priority class until all elements are eliminated.

By default LOQO uses a heuristic to determine how to set the priority classes. However, for multi-stage nonlinear stochastic programs we have altered the code so that the priority classes are set to reflect the inherent scenario tree structure. We call this ordering filtered dissection. It is discussed in detail in the next section.

2.5. Further Issues. There are many further implementations issues that we have not touched on such as

- Initializing $(\boldsymbol{x}, \ldots, \boldsymbol{q})$.
- Detecting optimality, unboundedness, and infeasibility.
- Setting μ appropriately in each iteration.
- Computing step lengths.
- Using the so-called predictor-corrector technique to find better search directions.
- Using iterative refinement and diagonal perturbation to improve numerical stability.

These issues and more are discussed in depth in [].

3. Tree Dissection.

4. Computational Results.

References

[1] P.T. Boggs, P.D. Domich, J.E. Rogers, and C. Witzgall. An interior-point method for linear and quadratic programming problems. Technical Report NISTIR-4556, National Institute of Standards and Technology, 1991.

[2] R. Fourer and S. Mehrotra. Solving symmetric indefinite systems in an interior point method for linear programming. *Math. Prog.*, 62:15–40, 1991.

[3] A. George and J.W.H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.

[4] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders. Primal-dual methods in linear programming. Technical Report SOL 91-3, Systems Optimization Laboratory, Stanford Univ., Stanford, CA, April 1991.
[5] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders. Solving reduced kkt systems in barrier methods for linear and quadratic programming. Technical Report SOL 91-7, Systems Optimization Laboratory, Stanford Univ., Stanford, CA, July 1991.
[6] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix Anal. Appl.*, 13(1):292–311, 1992.
[7] I.J. Lustig, R.E. Marsten, and D.F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Lin. Alg. and Appl.*, 152:191–222, 1991.
[8] I.J. Lustig, R.E. Marsten, and D.F. Shanno. On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM J. on Optimization*, 2:435–449, 1992.
[9] S. Pissanetzky. *Sparse Matrix Technology*. Academic Press, 1984.
[10] J.E. Rogers, P.T. Boggs, and P.D. Domich. A predictor-corrector-O3D formulation for quadratic programming. Technical report, National Institute of Standards and Technology, 1994.
[11] K. Turner. Computing projections for the Karmarkar algorithm. *Linear Algebra and Its Applications*, 152:141–154, 1991.
[12] R.J. Vanderbei. Symmetric quasi-definite matrices. *SIAM J. Optimization*, 1994. To appear.
[13] R.J. Vanderbei, A. Duarte, and B. Yang. An algorithmic and numerical comparison of several interior-point methods. Technical Report SOR 94-05, Princeton University, 1994.

Statistics and Operations Research, Princeton University, Princeton, NJ 08544
E-mail address: ajberger@emerald.princeton.edu

Statistics and Operations Research, Princeton University, Princeton, NJ 08544
E-mail address: mulvey@macbeth.princeton.edu

Silicon Graphics Inc., ??, CA
E-mail address: ??

Statistics and Operations Research, Princeton University, Princeton, NJ 08544
E-mail address: rvdb@princeton.edu