Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MARC C. STEINBACH

Recursive Direct Algorithms for Multistage Stochastic Programs in Financial Engineering

Preprint SC 98-23 (August 1998)

Recursive Direct Algorithms for Multistage Stochastic Programs in Financial Engineering

Marc C. Steinbach

August 29, 1998

Abstract

Multistage stochastic programs can be seen as discrete optimal control problems with a characteristic dynamic structure induced by the scenario tree. To exploit that structure, we propose a highly efficient dynamic programming recursion for the computationally intensive task of KKT systems solution within an interior point method. Test runs on a multistage portfolio selection problem demonstrate the performance of the algorithm.

1 Introduction

Multistage stochastic programs have become an important approach to model the process of decision making under uncertainty over a finite planning horizon. Important applications include, among others, financial engineering problems such as portfolio selection or asset and liability management. Multistage stochastic programs are considered a very hard class of optimization problems since their size can become excessively large even for coarse discretizations of the probability space and possibly the time horizon. Nevertheless, the characteristic structure of scenario trees makes these problems tractable by numerical algorithms. Among the most prominent ones are several variants of decomposition methods. Primal decomposition approaches [3, 6, 9] assign a small local optimization problem to every node and treat the vertical coupling between stages iteratively, by passing intermediate solutions up and objective and feasibility cuts down the scenario tree. Dual decomposition and progressive hedging algorithms [13, 15, 17] optimize individual scenarios and iterate on the nonanticipativity condition representing the horizontal coupling. Both groups of algorithms offer a significant degree of inherent parallelism [2, 4, 7, 18].
This paper presents a highly efficient method adopting the complementary strategy: as in [1, 5, 10], optimality and feasibility are achieved by an interior point iteration whereas the global coupling across the tree is treated explicitly. The distinguishing new component of our method is a recursive direct KKT solver that fully exploits the topological structure of the scenario tree. Rapid convergence is ensured by a simple but effective refinement technique that exploits the multistage stochastic nature of the problem. Due to space limitations, this technique will be presented in a forthcoming paper [21]; our emphasis here is on the inherent *dynamic structure* of multistage stochastic programs, which we view as discrete *optimal control problems*. Two formulations of dynamics are distinguished, explicit and implicit, and corresponding classes of *tree-sparse* quadratic programs (QP) are introduced. These purely equality-constrained QP represent simple, well-understood optimization problems in their own right, whose solution by direct application of *dynamic programming* naturally leads to the KKT

recursions of interest. On the other hand, their structure is sufficiently general to catch the essentials of *nonlinear* stochastic optimization within interior point or sequential quadratic programming (SQP) methods. In fact, the KKT recursion generalizes a previous serial version that has been successfully applied in a direct SQP method for nonlinear trajectory optimization [22, 23, 24].

The remainder of the paper is organized as follows. After introducing a quadratic multistage portfolio selection problem as an example, we formulate explicit and implicit tree-sparse quadratic programs as general problem classes, and present the recursive solution algorithms. Computational results on the example problem demonstrate the excellent performance of these algorithms. Finally we provide some conclusions and future directions of research.

2 A multistage portfolio selection problem

We consider a multistage extension of the mean-variance approach as introduced by Frauendorfer [8]. Assume that a portfolio manager can invest into n risky assets. After the initial investment at $t = 0$ the portfolio may be restructured at discrete times $t = 1, \dots, T$; it is redeemed one period later at time $T+1$. Denote by v_t^ν the transaction volume in asset ν at time t (positive for buying, negative for selling), by x_t^ν the capital at time t after rebalancing, and by $v_t, x_t \in \mathbf{R}^n$ the vectors with components v_t^ν, x_t^ν, respectively. The initial wealth is normalized and fully invested, and subsequent transactions do not change the wealth. With $e := (1, \dots, 1)$ these conditions read $e^* x_0 = 1$ and $e^* v_t = 0$, $t > 0$.

The objective is to minimize risk while achieving a prescribed expected return on investment. This yields the same efficient frontier as maximizing the expected return ρ for a given level of risk and can be seen as a multistage version of the classical Markowitz approach [11]. Other researchers formulate the portfolio selection problem as a stochastic network problem [12, 14] or model the tradeoff between profit and risk by a utility function [6].

A standard factor model is assumed for the random returns and associated risk: the risk driving factors are given by stochastic data ω_t observed at time $t = 0, \dots, T$. Denote by $\omega^t := (\omega_0, \dots, \omega_t)$ the history of random events up to time t, by $r_t(\omega^t) \in \mathbf{R}^n$ the vector of returns in period $[t-1, t]$, by $\bar{r}_T(\omega^T) = \mathbf{E}(r_{T+1}(\omega^{T+1}|\omega^T))$ the expected return in period $[T, T+1]$ conditioned on ω^T, and by $\Sigma_T(\omega^T) \in \mathbf{R}^{n\times n}$ the associated covariance matrix. The decision at time $t > 0$ is made after observing ω_t, yielding asset capitals $x_t^\nu = r_t^\nu(\omega^t) x_{t-1}^\nu + v_t^\nu$. Clearly, this leads to a *nonanticipative* policy $x = (x_0, \dots, x_T)$, that is, decisions may depend on the past but not on future events. (Note that ω_0 is observed before the initial decision and can thus be treated deterministically.)

Risk is modeled as the variance of the expected return at $T+1$, which can be written as

$$\sigma^2_{\omega^{T+1}} \left\{ r_{T+1}(\omega^{T+1})^* x_T(\omega^T) \right\} = \mathbf{E}_{\omega^T} \left\{ x_T(\omega^T)^* \left[\Sigma_T(\omega^T) + \bar{r}_T(\omega^T) \bar{r}_T(\omega^T)^* \right] x_T(\omega^T) \right\} - \rho^2. \quad (1)$$

For numerical computations, the stochastic evolvement of risk driving factors is approximated by a discrete set of *scenarios*, that is, sequences of events. Given a partial event history ω^t, we distinguish only finitely many possible outcomes for the next observation ω_{t+1}. This branching process creates a *scenario tree* rooted in the deterministic initial event ω_0. We denote by L_t the level set of nodes representing event histories ω^t up to time t, by $L \equiv L_T$ the set of leaves, and by $V := \bigcup_{t=0}^T L_t$ the complete vertex set. In the following we use $j \in V$ as node variable and denote by $j = 0$ the root, by $\pi(j)$ the predecessor (parent) of node j and by $S(j)$ its set of successors (children). The path to node j is a partial scenario with a probability p_j; thus the probabilities sum up to one in each level

set L_t, $t = 0, \ldots, T$. The tree structure reflects the growing amount of information: two scenarios that are indistinguishable up to time t share the path (the same information ω^t) up to some node $j \in L_t$ but may follow different branches when the next event ω_{t+1} is observed.
Assume that returns $r_j, \bar{r}_k$ and associated covariance matrices Σ_k are given ($j \in V$, $k \in L \equiv L_T$). Dropping the constant $-\rho^2$ in (1), the portfolio selection problem can then be written as the QP

$$\begin{aligned} \min_x \quad & \sum_{j \in L} x_j^* [p_j(\Sigma_j + \bar{r}_j \bar{r}_j^*)] x_j && (2) \\ \text{s.t.} \quad & e^* x_0 = 1, && (3) \\ & e^* x_j = r_j^* x_{\pi(j)} \qquad \forall j \in V \setminus \{0\}, && (4) \\ & \sum_{j \in L} p_j \bar{r}_j^* x_j = \rho. && (5) \end{aligned}$$

Note that the transaction volumes do not appear here: transition equations (4) simply state that the net value of the portfolio remains unchanged when it is restructured. The same condition determines the first component v_j^1 uniquely if the "control vector" $u_j := (v_j^2, \ldots, v_j^n)^*$ is given. Defining

$$E := \begin{pmatrix} -e^* \\ I \end{pmatrix} \in \mathbf{R}^{n \times (n-1)}, \qquad h_0 := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \mathbf{R}^n,$$

and $G_j := \mathrm{Diag}(r_j^1, \ldots, r_j^n)$, $h_j := 0$, $j \in V \setminus \{0\}$, we can thus replace (3,4) by explicit transition equations $x_j = G_j x_{\pi(j)} + E u_j + h_j$ (with $x_{\pi(0)} \in \mathbf{R}^0$). The complete QP reads now

$$\begin{aligned} \min_{u,x} \quad & \sum_{j \in L} x_j^* [p_j(\Sigma_j + \bar{r}_j \bar{r}_j^*)] x_j && (6) \\ \text{s.t.} \quad & x_0 = E u_0 + h_0, && (7) \\ & x_j = G_j x_{\pi(j)} + E u_j \qquad \forall j \in V \setminus \{0\}, && (8) \\ & \sum_{j \in L} p_j \bar{r}_j^* x_j = \rho. && (9) \end{aligned}$$

For a given scenario tree, both the implicit and the explicit QP represent *deterministic equivalents* of the verbally formulated stochastic program. The explicit form clearly expresses their nature as optimal control problems.
Realistic asset allocation problems typically include various inequality constraints imposed by the investor and/or the market, a cash account (with riskless, deterministic return), transaction costs [6], and often cash deposits and withdrawals. While the implicit form above is well suited and more efficient for our simple model problem, the explicit form can easily be extended to include such model refinements. Details will be given in a forthcoming paper.

3 General problem classes

In this section we introduce two general problem classes, the *tree-sparse quadratic programs*, based on explicit and implicit dynamic equations, respectively. The explicit case, as a natural extension of deterministic optimal control problems, is very similar to the problems introduced in [16]. The implicit form is often used in financial applications but has not yet received much attention from an optimal control viewpoint. A detailed study of both problem classes, the associated recursive algorithms, and the relation of explicit and implicit formulations will be given in [20].

3.1 Dynamic structure

In the following we consider dynamic systems on arbitrary trees. For the sake of brevity, the parent node of j will now be denoted by $i \equiv \pi(j)$.

Explicit dynamics. Associated with each node $j \in V$ are a *state* vector x_j, a *control* vector u_j, and an (affine) *transition mapping* p_j. Every state x_j depends on its preceding state $x_i \equiv x_{\pi(j)}$ and on control u_j through explicit dynamic equations

$$x_j = p_j(x_i, u_j) = G_j x_i + E_j u_j + h_j, \qquad j \in V. \tag{10}$$

To avoid a special notation for the root $j = 0$, we formally assume that $x_{\pi(0)}$ is zero-dimensional and not associated with any node. Every control vector $u = \{u_j\}_{j \in V}$ uniquely determines a state vector $x = \{x_j\}$. Thus the control variables represent all degrees of freedom in the dynamic system.

Implicit dynamics. As in the portfolio problem, a more compact formulation of dynamics is often possible. Associated with each node is again a state x_j but no control, and only part of x_j depends on the preceding state via implicit dynamics

$$p_j(x_i, x_j) = G_j x_i + h_j - P_j x_j = 0, \qquad j \in V. \tag{11}$$

Here the transition mapping p_j maps into a space with smaller dimension than x_j, and P_j is required to have full rank. Thus (11) leaves some local degrees of freedom ("controls") hidden in x_j.

3.2 Tree-sparse quadratic programs

We now formulate the quadratic programs which can be seen as discrete linear-quadratic optimal control problems: the aim is to find states and (possibly hidden) controls that minimize the objective while satisfying the dynamic equations and additional, problem-dependent linear constraints, the latter usually representing boundary conditions. The quadratic objective is typically a sum of local contributions, that is, it satisfies certain separability properties. This characterizes the tree-sparse case; only the linear objective term and additional constraints may introduce global coupling across the tree.

Explicit tree-sparse QP. The tree-sparse quadratic program with explicit dynamics takes the general form

$$\min_{u,x} \quad \sum_{j \in V} \left[\frac{1}{2} x_j^* H_j^0 x_j + \frac{1}{2} \begin{pmatrix} x_i \\ u_j \end{pmatrix}^* \begin{pmatrix} H_{ij} & J_j^* \\ J_j & K_j \end{pmatrix} \begin{pmatrix} x_i \\ u_j \end{pmatrix} + \begin{pmatrix} d_j \\ f_j \end{pmatrix}^* \begin{pmatrix} u_j \\ x_j \end{pmatrix} \right] \tag{12}$$

$$\text{s.t.} \qquad \begin{pmatrix} G_j & E_j \end{pmatrix} \begin{pmatrix} x_i \\ u_j \end{pmatrix} + h_j = x_j \qquad \forall j \in V, \tag{13}$$

$$\sum_{j \in V} \begin{pmatrix} D_j & F_j \end{pmatrix} \begin{pmatrix} u_j \\ x_j \end{pmatrix} + e_V = 0. \tag{14}$$

Note that the linear objective terms and constraints involve node variables u_j, x_j, whereas the mixed quadratic terms couple x_i, u_j along edge (i, j). These mixed terms may arise as second derivatives of nonlinear transition mappings in general optimization problems, hence the different contributions H_{ij}. We define $H_j := H_j^0 + \sum_{k \in S(j)} H_{jk}$ to collect pure quadratic terms associated with x_j, and reformulate the objective by substituting H_j for H_j^0 and zero for H_{ij}. Potentially the boundary conditions (14) couple all decision variables globally across the tree. Note, however, that initial conditions may be modeled separately by suitable choice of the root transition mapping p_0.

Regularity assumption. For QP (12–14) we require that (a) the constraints (13,14) have full rank, and (b) the objective is strictly convex on the null space of dynamic equations (13). Condition (b) is slightly stronger than usual: in addition to guaranteeing a unique minimizer, (a) and (b) also ensure that the QP can be solved by the recursive tree-sparse PH method developed below.

Implicit tree-sparse QP. The tree-sparse quadratic program with implicit dynamics has the general form

$$\min_{x} \quad \sum_{j\in V}\Big[\frac{1}{2}x_j^* H_j x_j + f_j^* x_j\Big] \tag{15}$$

$$\text{s.t.} \quad G_j x_i + h_j = P_j x_j \qquad \forall j \in V, \tag{16}$$

$$\sum_{j\in V} F_j x_j + e_V = 0. \tag{17}$$

Regularity assumption. For QP (15–17) we require more restrictive conditions than before: (a) the constraints (16,17) have full rank, (b) the objective is strictly convex on the full space, i.e., every H_j is positive definite. These conditions ensure that the tree-sparse SC method can be applied to compute the unique minimizer.

4 Recursive solution

Under the respective regularity assumptions stated above, both the implicit and explicit tree-sparse QP have unique minimizers that can be obtained through recursive solution of the KKT conditions. Basically these recursive algorithms proceed as follows. In a leaf, all the local variables can be formally eliminated. This modifies only the KKT data associated with the parent node, leaving a similarly structured KKT system on the subtree without the leaf. Thus leaves are cut off in an inward recursion until the root is eliminated, and a positive definite system determines the global multiplier μ. Node variables are then computed in an outward recursion. The inward recursion actually defines a direct symmetric factorization of the KKT matrix and transformation of the right hand side by one factor, while the outward recursion corresponds to the adjoint transformation. Details will be given in [20] along with proofs that only positive definite blocks are being inverted; here we restrict ourselves to an algorithmic description of the recursions. In both cases we introduce artificial zero blocks in the KKT conditions. These appear in positions that are modified during the recursion (fill-in), so they might contain nonzero entries without invalidating the algorithms.

4.1 Recursion for the explicit QP: tree-sparse PH method

The Lagrangian of the explicit tree-sparse QP reads

$$\begin{aligned} L(u,x,\lambda,\mu) \quad = \quad & \sum_{j\in V}\left[\frac{1}{2}x_j^* H_j x_j + \frac{1}{2}\begin{pmatrix} x_i \\ u_j \end{pmatrix}^* \begin{pmatrix} 0 & J_j^* \\ J_j & K_j \end{pmatrix} \begin{pmatrix} x_i \\ u_j \end{pmatrix} + \begin{pmatrix} d_j \\ f_j \end{pmatrix}^* \begin{pmatrix} u_j \\ x_j \end{pmatrix}\right] - \\ & \sum_{j\in V} \lambda_j^* \left[(G_j \; E_j)\begin{pmatrix} x_i \\ u_j \end{pmatrix} + h_j - x_j\right] - \mu^* \left[\sum_{j\in V}(D_j \; F_j)\begin{pmatrix} u_j \\ x_j \end{pmatrix} + e_V\right]. \end{aligned}$$

Equating partial derivatives with respect to all variables u_j, x_j, λ_j, μ to zero and introducing $X_V := 0$ yields the (linear indefinite) system of KKT conditions

$$J_j x_i + K_j u_j - E_j^* \lambda_j - D_j^* \mu + d_j = 0 \qquad \forall j \in V, \tag{18}$$

$$H_j x_j + \sum_{k \in S(j)} J_k^* u_k + \lambda_j - \sum_{k \in S(j)} G_k^* \lambda_k - F_j^* \mu + f_j = 0 \qquad \forall j \in V, \tag{19}$$

$$G_j x_i + E_j u_j - x_j + h_j = 0 \qquad \forall j \in V, \tag{20}$$

$$\sum_{j \in V} (D_j u_j + F_j x_j) + X_V \mu + e_V = 0. \tag{21}$$

Consider a leaf j all of whose siblings are also leaves, $S(i) \subset L$, and observe that $S(j) = \emptyset$ in (19). From (20) and (19) we obtain immediately

$$x_j = G_j x_i + E_j u_j + h_j, \qquad \lambda_j = -H_j x_j + F_j^* \mu - f_j. \tag{22}$$

Substituting the expression for x_j into (21) yields the local modification (in $\{i\} \cup S(i)$)

$$F_i x_i + \sum_{j \in S(i)} (D_j u_j + F_j x_j) + X_V \mu + e_V = \hat{F}_i x_i + \sum_{j \in S(i)} \hat{D}_j u_j + X_V \mu + \hat{e}_{V \setminus S(i)} \tag{23}$$

where $\hat{F}_i := F_i + \sum_j F_j G_j$, $\hat{D}_j := D_j + F_j E_j$, and $\hat{e}_{V \setminus S(i)} := e_V + \sum_j F_j h_j$. Similarly, substitution of $\lambda_j = -H_j G_j x_i - H_j E_j u_j + F_j^* \mu - \bar{f}_j$, $\bar{f}_j := f_j + H_j h_j$ into (18) and the parent's equation (19) yields

$$\hat{J}_j x_i + \hat{K}_j u_j - \hat{D}_j^* \mu + \hat{d}_j = 0, \qquad \hat{H}_i x_i + \sum_{j \in S(i)} \hat{J}_j^* u_j + \lambda_i - \hat{F}_i^* \mu + \hat{f}_i = 0, \tag{24}$$

respectively, where $\hat{J}_j := J_j + E_j^* H_j G_j$, $\hat{K}_j := K_j + E_j^* H_j E_j$, $\hat{d}_j := d_j + E_j^* \bar{f}_j$, $\hat{H}_i := H_i + \sum_j G_j^* H_j G_j$, and $\hat{f}_i := f_i + \sum_j G_j^* \bar{f}_j$. From the first equation of (24) one obtains now

$$u_j = -\hat{K}_j^{-1} (\hat{J}_j x_i - \hat{D}_j^* \mu + \hat{d}_j). \tag{25}$$

Substituting that expression into the second equation of (24) and into (23) restricts the KKT system to the subtree with vertex set $V \setminus S(i)$,

$$\tilde{H}_i x_i + \lambda_i - \tilde{F}_i^* \mu + \tilde{f}_i = 0, \qquad \hat{F}_i x_i + \sum_{j \in S(i)} \hat{D}_j u_j + X_V \mu + \hat{e}_{V \setminus S(i)} = \tilde{F}_i x_i + X_{V \setminus S(i)} \mu + e_{V \setminus S(i)}.$$

Here the modified data are $\tilde{H}_i := \hat{H}_i - \sum_j \hat{J}_j^* \hat{K}_j^{-1} \hat{J}_j$, $\tilde{F}_i := \hat{F}_i - \sum_j \hat{D}_j \hat{K}_j^{-1} \hat{J}_j$, $\tilde{f}_i := \hat{f}_i - \sum_j \hat{J}_j^* \hat{K}_j^{-1} \hat{d}_j$, $X_{V \setminus S(i)} := X_V + \sum_j \hat{D}_j \hat{K}_j^{-1} \hat{D}_j^*$, and $e_{V \setminus S(i)} := \hat{e}_{V \setminus S(i)} - \sum_j \hat{D}_j \hat{K}_j^{-1} \hat{d}_j$.
Repeating this process recursively creates the positive definite system $X_\emptyset \mu = -e_\emptyset$ after eliminating the root node. Using the solution μ and formulae (25,22), the node variables u_j, x_j, λ_j (in that order) are then calculated in the outward recursion. $X_V = 0$ may be replaced by any symmetric positive semidefinite block (of suitable dimensions) or, more generally, by any symmetric block yielding a positive definite $X_\emptyset$.
The elimination of x_j, λ_j is a projection of the KKT system onto the null space of the local dynamic equation, while elimination of u_j is a local minimization using the *projected Hessian* $\hat{K}_j$. Thus each recursion step represents a local version of the projected Hessian (PH) method (or null space method), so we call the recursion a *tree-sparse PH method.* See [20, 22] for more details.

4.2 Recursion for the implicit QP: tree-sparse SC method

The Lagrangian of the implicit tree-sparse QP is

$$L(x,\lambda,\mu) = \sum_{j\in V}\Big[\frac{1}{2}x_j^* H_j x_j + f_j^* x_j\Big] - \sum_{j\in V}\lambda_j^*\left[G_j x_i + h_j - P_j x_j\right] - \mu^*\Big[\sum_{j\in V} F_j x_j + e_V\Big].$$

After introducing $X_V := 0$ and $Y_j := 0$, $Z_j := 0$, $j \in V$, the KKT conditions read

$$H_j x_j + P_j^*\lambda_j - \sum_{k\in S(j)} G_k^*\lambda_k - F_j^*\mu + f_j = 0 \qquad \forall j\in V, \tag{26}$$

$$G_j x_i - P_j x_j + Y_j\lambda_j - Z_j^*\mu + h_j = 0 \qquad \forall j \in V, \tag{27}$$

$$\sum_{j\in V}(F_j x_j - Z_j\lambda_j) + X_V\mu + e_V = 0. \tag{28}$$

Again we consider a set of leaves $S(i) \subset L$ with common parent i, so that $S(j) = \emptyset$ in (26) and

$$x_j = -H_j^{-1}(P_j^*\lambda_j - F_j^*\mu + f_j). \tag{29}$$

Substituting x_j into (28) yields the local modification

$$\sum_{j\in S(i)}(F_j x_j - Z_j\lambda_j) + X_V\mu + e_V = -\sum_{j\in S(i)}\hat{Z}_j\lambda_j + \hat{X}_{V\setminus S(i)}\mu + \hat{e}_{V\setminus S(i)} \tag{30}$$

where $\hat{Z}_j := Z_j + F_j H_j^{-1}P_j^*$, $\hat{X}_{V\setminus S(i)} := X_V + \sum_j F_j H_j^{-1}F_j^*$, and $\hat{e}_{V\setminus S(i)} := e_V - \sum_j F_j H_j^{-1} f_j$. Substitution of x_j into (27) gives

$$G_j x_i + \hat{Y}_j\lambda_j - \hat{Z}_j^*\mu + \hat{h}_j = 0$$

with $\hat{Y}_j := Y_j + P_j H_j^{-1}P_j^*$, $\hat{h}_j := h_j + P_j H_j^{-1} f_j$, and consequentially

$$\lambda_j = -\hat{Y}_j^{-1}(G_j x_i - \hat{Z}_j^*\mu + \hat{h}_j). \tag{31}$$

From that we obtain final modifications of (30) and of the parent's equation (26),

$$F_i x_i - \sum_{j\in S(i)}\hat{Z}_j\lambda_j + \hat{X}_{V\setminus S(i)}\mu + \hat{e}_{V\setminus S(i)} = \tilde{F}_i x_i + X_{V\setminus S(i)}\mu + e_{V\setminus S(i)}, \quad \tilde{H}_i x_i + P_i^*\lambda_i - \tilde{F}_i^*\mu + \tilde{f}_i = 0,$$

respectively, where the modified data are $\tilde{F}_i := F_i + \sum_j \hat{Z}_j\hat{Y}_j^{-1}G_j$, $X_{V\setminus S(i)} := \hat{X}_{V\setminus S(i)} - \sum_j \hat{Z}_j\hat{Y}_j^{-1}\hat{Z}_j^*$, $e_{V\setminus S(i)} := \hat{e}_{V\setminus S(i)} + \sum_j \hat{Z}_j\hat{Y}_j^{-1}\hat{h}_j$, $\tilde{H}_i := H_i + \sum_j G_j^*\hat{Y}_j^{-1}G_j$, and $\tilde{f}_i := f_i + \sum_j G_j^*\hat{Y}_j^{-1}\hat{h}_j$.
The inward recursion again yields a system of the form $X_\emptyset\mu = -e_\emptyset$ defined on the empty tree, and node variables λ_j, x_j are calculated (in that order) from equations (31,29). X_V, Y_j may be replaced by arbitrary symmetric positive semidefinite blocks (more generally, any symmetric blocks leading to positive definite $X_\emptyset, \hat{Y}_j$), and Z_j by any rectangular blocks of suitable dimensions.
The elimination steps (31) for λ_j involve *Schur complements* $\hat{Y}_j$, and each recursion step represents a local version of the Schur complement method (or range space method). Hence we call that recursion a *tree-sparse SC method*. Again, see [20, 22] for details.

QP form	stages	scenarios	nodes	variables	constraints	matrix entries	time
explicit	2	81	91	1,365	729		0.01
explicit	3	729	820	12,300	6,561		0.22
explicit	4	6,561	7,381	110,715	59,049		2.11
explicit	6*	48,000	59,977	899,655	479,817	25,430,010	18.4
implicit	3	729	820	6,560	821		0.06
implicit	4	6,561	7,381	59,048	7,382		0.80
implicit	5	59,049	66,430	531,440	66,431		7.21
implicit	6*	217,728	259,939	2,079,512	259,940	23,394,504	33.1

Table 1: Sizes of KKT systems and solution times for tree-sparse recursions on balanced trees with 9 branches per node. *Tree not balanced; largest problem fitting into memory (KKT data $\approx 207\,\mathrm{MB}$).

5 Computational results

As test problems we consider explicit and implicit formulations of the portfolio management problem with $n = 8$ assets. The scenario trees have nine branches per node, giving $(9^{T+1} - 1)/8$ nodes and 9^T scenarios for the T-stage problem. In addition, two 6-stage problems that fill all available memory are constructed (with different numbers of branches). Benchmark runs are performed on a Silicon Graphics O2 workstation with a 175 MHz R10000 processor. The algorithms are implemented in C++ and treat all matrix blocks as dense and scenario-dependent. Table 1 lists the sizes of KKT systems and the CPU times in seconds. These data clearly demonstrate the efficiency of the recursions.

6 Conclusions and extensions

We have proposed a new method for multistage stochastic programs, based on a general strategy that is often pursued in nonlinear programming and clearly formulated in [22]: we treat nonlinearities and inequality constraints *globally* by suitable iterative algorithms (interior point or SQP methods), and exploit any problem-inherent structure on the *linear algebra* level. Clearly, that strategy has proved successful here due to the excellent performance of tree-sparse recursion in the critical task of KKT systems solution. In addition, convergence of the global iteration on inequalities is significantly enhanced by exploiting nonlinear characteristics of the problem through successive refinement [21]. Our method is not only conceptually elegant, it is also advantageous from a software engineering viewpoint. Since any relevant problem structure is concentrated in the KKT systems, one can apply generic iterative algorithms globally and still obtain high efficiency. For instance, linear algebra operations on subtrees are completely independent in the tree-sparse recursions, so they can easily be parallelized without affecting the interior point method.

In this work we have only considered the basic recursive structure of dynamics. In applications, *local* constraints involving only one control u_j, state x_j, or pair (x_i, u_j) may appear in many nodes. Such constraints can be eliminated by local projections; the details given in [22] (for the chain case) generalize immediately to the tree case, except for minor technical complications when the number of local constraints exceeds the number of local degrees of freedom.

With local constraints included, our view on multistage stochastic programs as optimal control problems with a recursive dynamic structure provides a general framework for classifying variables

and constraints, and for treating them appropriately. It also provides a guideline to exploit not only the common dynamic structure of multistage stochastic programs but also the local sparse structure of every specific application. In the portfolio selection problem, for instance, matrices G_j, E_j are (almost) diagonal in the explicit QP, and matrices E_j are identical in all nodes for both QP forms. This can be used to save CPU time and memory simply by specializing the node operations. The more difficult situation of sparse blocks in the presence of local constraints can be dealt with efficiently in a number of financial engineering problems. In the general case, suitable sparse local projections have to be constructed, which requires highly sophisticated sparse matrix techniques and remains a promising subject of future research.
We believe that the method proposed here is a significant step towards meeting at least one of the challenges formulated by Ruszczyński [19]: the development of specialized methods and high quality software for certain application areas.

7 Acknowledgement

This work is a result of an ongoing cooperation with K. Frauendorfer and H. Siede, to whom the author wishes to express his thanks for many inspiring discussions.

References

[1] A. J. Berger, J. M. Mulvey, E. Rothberg, and R. J. Vanderbei, *Solving multistage stochastic programs using tree dissection*, Technical Report SOR-95-07, Princeton University, June 1995.

[2] A. J. Berger, J. M. Mulvey, and A. Ruszczyński, *An extension of the DQA algorithm to convex stochastic programs*, SIAM J. Optimization, 4 (1994), pp. 735–753.

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

[4] J. R. Birge, C. J. Donohue, D. F. Holmes, and O. G. Svintsitski, *A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs*, Math. Programming, 75 (1996), pp. 327–352.

[5] J. Czyzyk, R. Fourer, and S. Mehrotra, *A study of the augmented system and column-splitting approaches for solving two-stage stochastic linear programs by interior-point methods*, ORSA J. Computing, 7 (1995), pp. 474–490.

[6] G. B. Dantzig and G. Infanger, *Multi-stage stochastic linear programs for portfolio optimization*, Annals Oper. Res., 45 (1993), pp. 59–76.

[7] E. A. Eschenbach, C. A. Shoemaker, and H. M. Caffey, *Parallel algorithms for stochastic dynamic programming with continuous state and control variables*, ORSA J. Computing, 7 (1995), pp. 386–401.

[8] K. Frauendorfer, *The stochastic programming extension of the Markowitz approach*, Int. J. Mass-Parallel Comput. Inform. Syst., 5 (1995), pp. 449–460.

[9] H. I. Gassmann, *MSLiP: A computer code for the multistage stochastic linear programming problem*, Math. Programming, 47 (1990), pp. 407–423.

[10] E. R. Jessup, D. Yang, and S. A. Zenios, *Parallel factorization of structured matrices arising in stochastic programming*, SIAM J. Optimization, 4 (1994), pp. 833–846.

[11] H. M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*, John Wiley, New York, 1959.

[12] J. M. Mulvey, *Nonlinear network models in finance*, Adv. Math. Progr. Fin. Planning, 1 (1987), p. 253.

[13] J. M. Mulvey and A. Ruszczyński, *A new scenario decomposition method for large-scale stochastic optimization*, Oper. Res., 43 (1995), pp. 477–490.

[14] J. M. Mulvey and H. Vladimirou, *Stochastic network optimization models for investment planning*, Annals Oper. Res., 20 (1989), pp. 187–217.

[15] ——, *Applying the progressive hedging algorithm to stochastic generalized networks*, Annals Oper. Res., 31 (1991), pp. 399–424.

[16] R. T. Rockafellar and R. J.-B. Wets, *Generalized linear-quadratic problems of deterministic and stochastic optimal control in discrete time*, SIAM J. Control Optim., 28 (1990), pp. 810–822.

[17] ——, *Scenarios and policy aggregation in optimization under uncertainty*, Math. Oper. Res., 16 (1991), pp. 119–147.

[18] A. Ruszczyński, *Parallel decomposition of multistage stochastic programming problems*, Math. Programming, 58 (1993), pp. 201–228.

[19] ——, *Decomposition methods in stochastic programming*, Math. Programming, 79 (1997), pp. 333–353. Invited lectures of the 16th International Symposium on Mathematical Programming, Lausanne EPFL.

[20] M. C. Steinbach, *Back to the roots: recursive optimization on dynamic trees.* In preparation.

[21] ——, *Recursive direct optimization and successive refinement in multistage stochastic programs.* In preparation.

[22] ——, *Fast Recursive SQP Methods for Large-Scale Optimal Control Problems*, Ph. D. dissertation, University of Heidelberg, 1995.

[23] ——, *Structured interior point SQP methods in optimal control*, Z. Angew. Math. Mech., 76 (1996), pp. 59–62.

[24] M. C. Steinbach, H. G. Bock, G. V. Kostin, and R. W. Longman, *Mathematical optimization in robotics: Towards automated high speed motion planning*, Surv. Math. Ind., 7 (1998), pp. 303–340.