Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MARC C. STEINBACH

Recursive Direct Optimization and Successive Refinement in Multistage Stochastic Programs

Preprint SC 98-27 (October 1998)

Recursive Direct Optimization and Successive Refinement in Multistage Stochastic Programs

Marc C. Steinbach

October 30, 1998

Abstract

The paper presents a new algorithmic approach for multistage stochastic programs which are 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 a primal-dual interior point method. Convergence is drastically enhanced by a successive refinement technique providing both primal and dual initial estimates. Test runs on a multistage portfolio selection problem demonstrate the performance of the method.

Keywords: Multistage stochastic programs, discrete dynamics, tree-sparse QP, KKT recursion, successive refinement

AMS Subject classification: Primary 90C15, 90C06; Secondary 65F50, 90A09

1 Introduction

Multistage stochastic programs have become a standard 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 [4, 8, 11] 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 [16, 18, 20] 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, 6, 9, 21].

This paper presents a highly efficient method adopting the complementary strategy: as in [1, 7, 12], optimality and feasibility are achieved by an interior point iteration whereas the global coupling across the tree is treated explicitly. Two crucial aspects are addressed: the algebraic complexity of KKT systems on the scenario tree, and convergence of the nonlinear iteration. For the KKT systems we propose a recursive direct solver that fully exploits the sparseness induced by the inherent dynamic structure. Two formulations of dynamics are distinguished, explicit and implicit, and corresponding *tree-sparse* quadratic programs (QP) are introduced. These purely equality-constrained standard problems represent much larger classes of possibly nonlinear optimization problems that exhibit the same algebraic structure in an interior point or sequential quadratic programming (SQP) framework. (In fact, the KKT recursion generalizes an earlier serial version that has been successfully applied in a direct SQP method for nonlinear trajectory optimization [23, 24, 27].) Rapid convergence of the primal-dual interior point method is ensured by a simple but effective refinement technique relying on the stochastic nature of the problem: from a sequence of approximate problems, it generates an ideal hot-start close to the primal-dual central path.

The paper is organized as follows. After introducing a quadratic portfolio selection problem as an example in Section 2, we formulate explicit and implicit quadratic stochastic programs, and their tree-sparse discrete counterparts in Section 3. The recursive solution algorithms of [25] are then reviewed in Section 4. Section 5 presents the successive refinement technique and investigates its effect in the interior point method. Computational results on the example problem are given in Section 6, and finally Section 7 provides 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 [10]. 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 capital is normalized and completely invested, and subsequent transactions do not change the capital. 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 [14]. Other researchers formulate the portfolio selection problem as a stochastic network problem [15, 17] or model the tradeoff between profit and risk by a utility function [8].

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. (Notice 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

$$\begin{aligned}\sigma^2_{\omega^{T+1}}[r^*_{T+1} x_T] &= \mathbf{E}_{\omega^{T+1}}[x_T^* r_{T+1} r^*_{T+1} x_T] - [\mathbf{E}_{\omega^{T+1}}(r^*_{T+1} x_T)]^2 \\ &= \mathbf{E}_{\omega^T}[x_T^*(\Sigma_T + \bar{r}_T \bar{r}_T^*) x_T] - [\mathbf{E}_{\omega^T}(\bar{r}_T^* x_T)]^2.\end{aligned}$$

Dropping the constant $[\mathbf{E}_{\omega^T}(\bar{r}_T^* x_T)]^2 = \rho^2$, this leads to the convex quadratic multistage stochastic program

$$\min_{x(\omega)} \quad \mathbf{E}_{\omega^T}\{x_T(\omega^T)^*[\Sigma_T(\omega^T) + \bar{r}_T(\omega^T)\bar{r}_T(\omega^T)^*] x_T(\omega^T)\} \tag{1}$$

$$\text{s.t.} \quad e^* x_0 = 1, \tag{2}$$

$$e^* x_t(\omega^t) = r_t(\omega^t)^* x_{t-1}(\omega^{t-1}), \qquad t = 1, \dots, T, \tag{3}$$

$$\mathbf{E}_{\omega^T}\{\bar{r}_T(\omega^T)^* x_T(\omega^T)\} = \rho. \tag{4}$$

In measure-theoretic terms, the stochastic data $\{\omega^t\}_{t=0}^T$ generate *σ-fields* $\mathcal{F}_t$ so that $\mathcal{F} = \{\mathcal{F}_t\}$ is a *filtration* of the underlying probability space $(\Omega, \mathcal{B}, P)$ and x is *$\mathcal{F}$-adapted*, that is, $\mathcal{F}_0 \subseteq \dots \subseteq \mathcal{F}_T \subseteq \mathcal{B}$ and x_t is $\mathcal{F}_t$-measurable for each t.

Although we will use that terminology, only discrete distributions are of interest in the present paper and we will not discuss any of the subtle issues associated with a general probabilistic setting.

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 $p_j \equiv P(\omega^t)$; thus the probabilities sum up to one in each level set L_t, $t = 0, \dots, T$. 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 for $j \in V$, $k \in L \equiv L_T$. Defining $Q_j := \Sigma_j + \bar{r}_j \bar{r}_j^*$, the *deterministic equivalent* of the stochastic portfolio selection problem (1–4) can then be written as the QP

$$\min_x \quad \sum_{j \in L} x_j^* (p_j Q_j) x_j \tag{5}$$

$$\text{s.t.} \quad e^* x_0 = 1, \tag{6}$$

$$e^* x_j = r_j^* x_{\pi(j)} \qquad \forall j \in V \setminus \{0\}, \tag{7}$$

$$\sum_{j \in L} (p_j \bar{r}_j^*) x_j = \rho. \tag{8}$$

Bound constraints $x_j \in [b_{lj}, b_{uj}]$ may be added to prohibit shortsales or model other restrictions imposed by the investor and/or the market. Notice that the transaction volumes do not appear in (5–8): transition equations (7) 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" $u_j := (v_j^2, \dots, v_j^n)^*$ 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 $G_j := \mathrm{Diag}(r_j^1, \ldots, r_j^n)$, $h_j := 0$, $j \in V \setminus \{0\}$, one can thus replace *implicit* transition equations (6,7) by equivalent *explicit* ones, $x_j = G_j x_{\pi(j)} + E u_j + h_j$ (where $x_{\pi(0)} \in \mathbf{R}^0$). This leads to the alternative QP formulation

$$
\begin{aligned}
\min_{u,x} \quad & \sum_{j \in L} x_j^* (p_j Q_j) x_j && (9)\\
\text{s.t.} \quad & x_0 = E u_0 + h_0, && (10)\\
& x_j = G_j x_{\pi(j)} + E u_j \qquad \forall j \in V \setminus \{0\}, && (11)\\
& \sum_{j \in L} (p_j \bar{r}_j^*) x_j = \rho. && (12)
\end{aligned}
$$

The explicit form clearly expresses the *optimal control nature* of the portfolio selection problem. Of course, bounds $u_j \in [b_{lj}^u, b_{uj}^u]$, $x_j \in [b_{lj}^x, b_{uj}^x]$ may be added as in the implicit case.

Real-life asset allocation problems typically include a cash account (with riskless, deterministic return), transaction costs, and often cash deposits and withdrawals [8]. While the implicit form above is well suited and more efficient for the simple model problem, the explicit form can easily be extended to include such model refinements. Details will be given in a forthcoming paper.

The portfolio selection problem is a special case of more general quadratic multistage stochastic programs with respective explicit and implicit forms

$$
\begin{aligned}
\min_{u,x} \quad & \mathbf{E} \left\{ \sum_{t=0}^{T} \frac{1}{2} x_t^* H_t^0 x_t + \frac{1}{2} \begin{pmatrix} x_{t-1} \\ u_t \end{pmatrix}^* \begin{pmatrix} H_{t-1,t} & J_t^* \\ J_t & K_t \end{pmatrix} \begin{pmatrix} x_{t-1} \\ u_t \end{pmatrix} + \begin{pmatrix} d_t \\ f_t \end{pmatrix}^* \begin{pmatrix} u_t \\ x_t \end{pmatrix} \right\} && (13)\\
\text{s.t.} \quad & (G_t \ \ E_t) \begin{pmatrix} x_{t-1} \\ u_t \end{pmatrix} + h_t = x_t, \qquad t = 0, \ldots, T, && (14)\\
& \mathbf{E} \left\{ \sum_{t=0}^{T} (D_t \ \ F_t) \begin{pmatrix} u_t \\ x_t \end{pmatrix} + e_t \right\} = 0, && (15)
\end{aligned}
$$

and

$$
\begin{aligned}
\min_{x} \quad & \mathbf{E} \left\{ \sum_{t=0}^{T} \frac{1}{2} x_t^* H_t x_t + f_t^* x_t \right\} && (16)\\
\text{s.t.} \quad & G_t x_{t-1} + h_t = P_t x_t, \qquad t = 0, \ldots, T, && (17)\\
& \mathbf{E} \left\{ \sum_{t=0}^{T} F_t x_t + e_t \right\} = 0. && (18)
\end{aligned}
$$

Here we require the QP data and decision variables to be adapted to some filtration $\mathcal{F}$ rather than specifying the dependence on stochastic events ω^t explicitly.

As in [28], the formulation was chosen in such a way that the Lagrangians can be written in form of expectations,

$$L(u,x,\lambda,\mu) =$$
$$\mathbf{E}\left\{\sum_{t=0}^{T}\frac{1}{2}x_t^* H_t^0 x_t + \frac{1}{2}\begin{pmatrix}x_{t-1}\\ u_t\end{pmatrix}^*\begin{pmatrix}H_{t-1,t} & J_t^*\\ J_t & K_t\end{pmatrix}\begin{pmatrix}x_{t-1}\\ u_t\end{pmatrix} + \begin{pmatrix}d_t\\ f_t\end{pmatrix}^*\begin{pmatrix}u_t\\ x_t\end{pmatrix} - \right.$$
$$\left. \lambda_t^*\left[(G_t\ \ E_t)\begin{pmatrix}x_{t-1}\\ u_t\end{pmatrix} + h_t - x_t\right] - \mu^*\left[(D_t\ \ F_t)\begin{pmatrix}u_t\\ x_t\end{pmatrix} + e_t\right]\right\},$$

and

$$L(x,\lambda,\mu) = \mathbf{E}\left\{\sum_{t=0}^{T}\frac{1}{2}x_t^* H_t x_t + f_t^* x_t - \lambda_t^*[G_t x_{t-1} + h_t - P_t x_t] - \mu^*[F_t x_t + e_t]\right\}.$$

Deterministic equivalents of the stochastic programs are given by the tree-sparse quadratic problems of the following section, where node probabilities p_j are absorbed into the node data or dual states. Details will be given when we return to the current formulation in the context of successive refinement.

3 Standard problem classes

In this section we introduce two general problem classes based on explicit and implicit dynamic equations, respectively, the *tree-sparse quadratic programs*. The explicit case, as a natural extension of deterministic optimal control problems, is very similar to the problems introduced in [19]. 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 [26]. Tree-sparse QP also arise in other contexts than stochastic programming, so the presentation is kept purely algebraic.

We emphasize that our standard problem classes are formulated as *equality-constrained convex QP* whose unique solution is obtained by solving the associated *linear* KKT system. If inequalities are included, the interior point method generates a sequence of similarly structured KKT systems with modified diagonal. Thus explicit and implicit tree-sparse QP represent the much larger class of possibly nonlinear optimization problems whose KKT systems exhibit the tree-sparse structure in an interior point framework. Multistage stochastic linear programs in standard form are shown to belong to that class.

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 $i \equiv \pi(j)$.

3.1.1 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

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

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.

3.1.2 Implicit dynamics

As in the portfolio management 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

$$p_j(x_i, x_j) = G_j x_i + h_j - P_j x_j = 0, \qquad j \in V. \tag{19}$$

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 (19) leaves some local degrees of freedom ("controls") hidden in x_j.

It is easily seen that explicit dynamics have an equivalent implicit form. The converse is also true but additional variables must be introduced in the general case [26].

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.

3.2.1 Explicit tree-sparse QP

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

$$\min_{u,x} \quad \sum_{j\in V}\left[\frac{1}{2}x_j^* H_j^0 x_j + \frac{1}{2}\begin{pmatrix}x_i\\u_j\end{pmatrix}^* \begin{pmatrix}H_{ij} & J_j^*\\ J_j & K_j\end{pmatrix}\begin{pmatrix}x_i\\u_j\end{pmatrix} + \begin{pmatrix}d_j\\f_j\end{pmatrix}^*\begin{pmatrix}u_j\\x_j\end{pmatrix}\right] \tag{20}$$

$$\text{s.t.} \quad (G_j \;\; E_j)\begin{pmatrix}x_i\\u_j\end{pmatrix} + h_j = x_j \qquad \forall j \in V, \tag{21}$$

$$\sum_{j\in V}(D_j \;\; F_j)\begin{pmatrix}u_j\\x_j\end{pmatrix} + e_V = 0. \tag{22}$$

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 more 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}.

The boundary conditions (22) are specified in the most general form possible; potentially they 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. Furthermore, if *local* constraints are present in many nodes, these should also be treated separately. In [23] we distinguish state constraints $F_j^x x_j + e_j^x = 0$, control constraints $D_j^u u_j + e_j^u = 0$, and coupled constraints $F_{ij}^c x_i + D_j^c u_j + e_j^c = 0$. Their respective treatment (given in [23] for the chain case) generalizes immediately to trees, except for minor technical complications when the number of local constraints exceeds the number of local degrees of freedom. Range constraints $F_{ij}^r x_i + D_j^r u_j \in [r_{lj}, r_{uj}]$ may also be included.

Regularity assumption. For the QP (20–22) we require that (a) the constraints (21,22) have full rank, and (b) the objective is strictly convex on the null space of dynamic equations (21). These conditions are slightly stronger than usual: in addition to guaranteeing a unique minimizer, they also ensure that the QP can be solved by the recursive *tree-sparse projected Hessian method* developed below.

3.2.2 Implicit tree-sparse QP

The general tree-sparse quadratic program with implicit dynamics reads

$$\min_{x} \quad \sum_{j \in V} \Big[\frac{1}{2} x_j^* H_j x_j + f_j^* x_j\Big] \tag{23}$$

$$\text{s.t.} \quad G_j x_i + h_j = P_j x_j \qquad \forall j \in V, \tag{24}$$

$$\sum_{j \in V} F_j x_j + e_V = 0. \tag{25}$$

Again, initial conditions may be conveniently modeled by the root transition mapping p_0, and local constraints should be treated separately if present. Due to the absence of controls we distinguish only state constraints $F_j^x x_j + e_j^x = 0$ and possibly range constraints $F_j^r x_j \in [r_{lj}, r_{uj}]$.

Regularity assumption. For the QP (23–25) we require that (a) the constraints (24,25) have full rank, and (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 Schur complement method* of Section 4.2 can compute the unique minimizer. (Actually (b) may be replaced by a slightly weaker condition [26].)

3.3 Multistage stochastic linear programs

Dynamics and constraints are not distinguished in standard-form multistage stochastic *linear* programs; nevertheless these problems may be seen as a special case of the bound-constrained implicit tree-sparse QP. In the notation of [5], the standard form can be written

$$\min_{x_0, \dots, x_T} \quad c_0^* x_0 + \cdots + c_T^* x_T \tag{26}$$

$$\text{s.t.} \quad A_{t0} x_0 + \cdots + A_{tt} x_t = b_t \in \mathbf{R}^{m_t}, \qquad t = 0, \dots, T, \tag{27}$$

$$l_t \le x_t \le u_t \in \mathbf{R}^{n_t}, \qquad t = 0, \dots, T, \tag{28}$$

where the matrices A_{ts} and vectors c_t, b_t, l_t, u_t may all be random. That is, the corresponding quantities of a deterministic equivalent, $A_{jk}, c_j, b_j, l_j, u_j$, $j \in L_t$, $k \in L_s$, may differ in all nodes of the scenario tree.

Assume first that x_{t+1} depends only on the present and not on the past, that is, $A_{t+1,s} = 0$ for $s < t$. Then we simply set $H_j := 0$, $f_j := c_j$, $b_{lj} := l_j$, $b_{uj} := u_j$, $G_j := -A_{ji}$, $P_j := A_{jj}$, and $h_j := b_j$. In the general case one carries along past states and formulates the problem in variables $\bar{x}_j := (x_0, \dots, x_i, x_j) \in \mathbf{R}^{N_j}$ where

$N_j := n_0 + \cdots + n_{t(j)}$. This yields the remaining quantities $f_j := (0, \dots, 0, c_j)$, $b_{lj} := (-\infty, \dots, -\infty, l_j)$, $b_{uj} := (\infty, \dots, \infty, u_j)$, and

$$G_j := \begin{pmatrix} I & & \\ & \ddots & \\ & & I \\ -A_{j0} & \dots & -A_{ji} \end{pmatrix}, \qquad P_j := \begin{pmatrix} I & & & \\ & \ddots & & \\ & & I & \\ 0 & \dots & 0 & A_{jj} \end{pmatrix}, \qquad h_j := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_j \end{pmatrix}$$

where $G_j \in \mathbf{R}^{M_j \times N_i}$, $P_j \in \mathbf{R}^{M_j \times N_j}$, $h_j \in \mathbf{R}^{M_j}$, and $M_j := N_i + m_{t(j)}$. In both cases the rank condition on P_j requires that A_{jj} is nonsingular. Naturally, the specific structure in the general case should receive special treatment in numerical computations.

Of course, range constraints may also be included in (26–28) and, depending on the structure of the matrices A_{jk}, it may often be convenient to classify some rows of (27) as global constraints or state constraints. Note in specific that this is necessary if A_{jj} is rank-deficient: then one can find a transformation U_j so that the upper rows of $U_j A_{jj}$ form a nonsingular matrix $\tilde{A}_{jj}$ and the lower rows vanish. Hence $\tilde{A}_{jj}$ replaces A_{jj} in the definition of P_j, and the lower rows of $U_j(A_{j0}, \dots, A_{ji}, b_j)$ define a local condition for $\bar{x}_i$.

4 Tree recursion

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. In [25] we have given an elementary algebraic derivation of the recursions. Here we explain the basic elimination steps, summarize the node operations in tabular form, and indicate how the algorithms can be interpreted as symmetric factorizations of the KKT matrix with corresponding solution procedures. In both cases we introduce artificial zero blocks in the KKT conditions. These appear in positions where fill-in is created during the recursion, so they might contain nonzero entries without invalidating the algorithms.

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

With dual states λ_j and global multiplier μ, the Lagrangian of the explicit tree-sparse QP reads

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

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

$$
\begin{aligned}
J_j x_i + K_j u_j - E_j^*\lambda_j - D_j^*\mu + d_j &= 0 \qquad \forall j\in V, && (29)\\
H_j x_j + \sum_{k\in S(j)} J_k^* u_k + \lambda_j - \sum_{k\in S(j)} G_k^*\lambda_k - F_j^*\mu + f_j &= 0 \qquad \forall j\in V, && (30)\\
G_j x_i + E_j u_j - x_j + h_j &= 0 \qquad \forall j\in V, && (31)\\
\sum_{j\in V}(D_j u_j + F_j x_j) + X_V\mu + e_V &= 0. && (32)
\end{aligned}
$$

These form a linear indefinite system whose structure is best seen when written in matrix form $\Omega z = -a$. For the example tree of Fig. 1, that system reads

$$
\left[\begin{array}{cccccccccccc|ccccccc}
K_0 &&&&&&&&&&&& E_0^* &&&&&& D_0^*\\
& H_0 & J_1^* && J_2^* &&&&&&&& -I & G_1^* & G_2^* &&&& F_0^*\\
& J_1 & K_1 &&&&&&&&&&& E_1^* &&&&& D_1^*\\
&&& H_1 &&& J_3^* && J_4^* &&&&& -I && G_3^* & G_4^* && F_1^*\\
& J_2 &&& K_2 &&&&&&&&&& E_2^* &&&& D_2^*\\
&&&&& H_2 &&&&& J_5^* &&&& -I &&& G_5^* & F_2^*\\
&&& J_3 &&& K_3 &&&&&&&&& E_3^* &&& D_3^*\\
&&&&&&& H_3 &&&&&&&& -I &&& F_3^*\\
&&& J_4 &&&&& K_4 &&&&&&&& E_4^* && D_4^*\\
&&&&&&&&& H_4 &&&&&&& -I && F_4^*\\
&&&&& J_5 &&&&& K_5 &&&&&&& E_5^* & D_5^*\\
&&&&&&&&&&& H_5 &&&&&& -I & F_5^*\\
\hline
E_0 & -I &&&&&&&&&&&&&&&&&\\
& G_1 & E_1 & -I &&&&&&&&&&&&&&&\\
& G_2 &&& E_2 & -I &&&&&&&&&&&&&\\
&&& G_3 &&& E_3 & -I &&&&&&&&&&&\\
&&& G_4 &&&&& E_4 & -I &&&&&&&&&\\
&&&&& G_5 &&&&& E_5 & -I &&&&&&&\\
D_0 & F_0 & D_1 & F_1 & D_2 & F_2 & D_3 & F_3 & D_4 & F_4 & D_5 & F_5 &&&&&&& -X_V
\end{array}\right]
\left[\begin{array}{c}
u_0\\ x_0\\ u_1\\ x_1\\ u_2\\ x_2\\ u_3\\ x_3\\ u_4\\ x_4\\ u_5\\ x_5\\ \hline -\lambda_0\\ -\lambda_1\\ -\lambda_2\\ -\lambda_3\\ -\lambda_4\\ -\lambda_5\\ -\mu
\end{array}\right]
= -
\left[\begin{array}{c}
d_0\\ f_0\\ d_1\\ f_1\\ d_2\\ f_2\\ d_3\\ f_3\\ d_4\\ f_4\\ d_5\\ f_5\\ \hline h_0\\ h_1\\ h_2\\ h_3\\ h_4\\ h_5\\ e_V
\end{array}\right].
$$

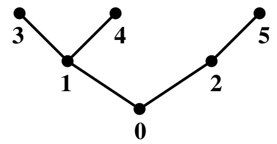

Figure 1: A simple tree.

Consider a leaf j all of whose siblings are also leaves, $S(i) \subseteq L$, and observe that $S(j) = \emptyset$ in (30). Elimination of x_j and λ_j from (31) and (30), respectively, yields

$$x_j = G_j x_i + E_j u_j + h_j, \qquad \lambda_j = -H_j x_j + F_j^* \mu - f_j. \tag{33}$$

These expressions are substituted into the remaining KKT conditions, and from the resulting modification of (29) one obtains

$$u_j = -\hat{K}_j^{-1}(\hat{J}_j x_i - \hat{D}_j^* \mu + \hat{d}_j) \tag{34}$$

where the modified data $\hat{D}_j, \hat{J}_j, \hat{K}_j, \hat{d}_j$ are defined in Table 1. Substituting u_j into the remaining conditions completes one recursion step: local variables and conditions of node j are eliminated, and the resulting KKT system has the same structure as the original one, with modified data in the global conditions and in the local conditions of the parent node $i = \pi(j)$.

Recursive repetition of this process creates the positive definite system $X_\emptyset \mu = -e_\emptyset$ after eliminating the root, as indicated by the index $\emptyset$ (the current vertex set) on X, e. (The positive-definiteness of $X_\emptyset$ and of $\hat{K}_j$ in (34) will be proved in [26].) Using the solution μ, the node variables u_j, x_j, λ_j (in that order) are then calculated from (34,33) in the outward recursion. The algorithm works with any symmetric positive semidefinite block X_V (of suitable dimensions) or, more generally, any symmetric block yielding a positive definite $X_\emptyset$.

Three phases of the algorithm are distinguished. The modification of KKT data in the inward recursion defines a symmetric *factorization* of the KKT matrix, $\Omega = \Lambda \Pi \Lambda^*$, and the associated *transformation* of the right hand side, $\bar{a} = \Lambda^{-1} a$. In analogy to the dense case, we call the final calculation of (negative) variables in the outward recursion the *substitution* phase, $-z = \Lambda^{-*} \bar{z}$ where $\bar{z} = \Pi^{-1} \bar{a}$.

Individual block operations of one recursion step are listed in Table 1. The symmetry can be seen in corresponding operations of the three phases; slight asymmetries result from merging multiplication by Π^{-1} into the substitution phase. For simplicity we assume standard Cholesky factorizations of $\hat{K}_j$ and $X_\emptyset$, which may be replaced by other variants if appropriate.

Table 1: Node operations and additional root operations (last line) of the tree-sparse PH method. Factorization and transformation proceed from top to bottom, substitution proceeds in reverse order, producing $-z = \Omega^{-1}a$. Indices on X, e denote the current vertex set; matrix/vector notation neglects possible modifications from earlier recursion steps. w is a "workspace" vector.

Factorization	Transformation	Substitution
$\hat{F}_i = F_i + F_j G_j$	$\hat{e}_{V\setminus\{j\}} = e_V + F_j h_j$	$\lambda_j = \quad F_j^* \mu +$
$\hat{D}_j = D_j + F_j E_j$		
$\hat{J}_j = J_j + E_j^* H_j G_j$	$w = f_j + H_j h_j$	$H_j(-x_j) - f_j$
$\hat{H}_i = H_i + G_j^* H_j G_j$	$\hat{f}_i = f_i + G_j^* w$	$-x_j = \quad G_j(-x_i) +$
$\hat{K}_j = K_j + E_j^* H_j E_j$	$\hat{d}_j = d_j + E_j^* w$	$E_j(-u_j) - h_j$
$\hat{K}_j = L_j L_j^*$		
$\tilde{D}_j = \hat{D}_j L_j^{-*}$		
$\tilde{J}_j = L_j^{-1} \hat{J}_j$	$\tilde{d}_j = L_j^{-1} \hat{d}_j$	$-u_j = \quad L_j^{-*}[$
$\tilde{F}_i = \hat{F}_i - \tilde{D}_j \tilde{J}_j$		
$\tilde{H}_i = \hat{H}_i - \tilde{J}_j^* \tilde{J}_j$	$\tilde{f}_i = \hat{f}_i - \tilde{J}_j^* \tilde{d}_j$	$-\tilde{J}_j(-x_i)$
$X_{V\setminus\{j\}} = X_V + \tilde{D}_j \tilde{D}_j^*$	$e_{V\setminus\{j\}} = \hat{e}_{V\setminus\{j\}} - \tilde{D}_j \tilde{d}_j$	$-\tilde{D}_j^* \mu + \tilde{d}_j]$
$X_\emptyset = L_\emptyset L_\emptyset^*$	$\hat{e}_\emptyset = L_\emptyset^{-1} e_\emptyset$	$\mu = \quad L_\emptyset^{-*}(-\hat{e}_\emptyset)$

The elimination of x_j, λ_j projects the KKT system on 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, and we refer to the recursion as the *tree-sparse PH method.* See [26, 23] for more details.

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

The Lagrangian of the implicit tree-sparse QP reads

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

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

$$H_j x_j + P_j^* \lambda_j - \sum_{k\in S(j)} G_k^* \lambda_k - F_j^* \mu + f_j = 0 \qquad \forall j \in V, \tag{35}$$

$$G_j x_i - P_j x_j + Y_j \lambda_j - Z_j^* \mu + h_j = 0 \qquad \forall j \in V, \tag{36}$$

$$\sum_{j\in V}(F_j x_j - Z_j \lambda_j) + X_V \mu + e_V = 0. \tag{37}$$

In matrix form $\Omega z = -a$, this system (for the example tree in Fig. 1) reads

$$\left[\begin{array}{cccccc|cccccc|c}
H_0 &&&&&& -P_0^* & G_1^* & G_2^* &&&& F_0^* \\
& H_1 &&&&&& -P_1^* && G_3^* & G_4^* && F_1^* \\
&& H_2 &&&&&& -P_2^* &&& G_5^* & F_2^* \\
&&& H_3 &&&&&& -P_3^* &&& F_3^* \\
&&&& H_4 &&&&&& -P_4^* && F_4^* \\
&&&&& H_5 &&&&&& -P_5^* & F_5^* \\
\hline
-P_0 &&&&&& -Y_0 &&&&&& Z_0^* \\
G_1 & -P_1 &&&&&& -Y_1 &&&&& Z_1^* \\
G_2 && -P_2 &&&&&& -Y_2 &&&& Z_2^* \\
& G_3 && -P_3 &&&&&& -Y_3 &&& Z_3^* \\
& G_4 &&& -P_4 &&&&&& -Y_4 && Z_4^* \\
&& G_5 &&& -P_5 &&&&&& -Y_5 & Z_5^* \\
F_0 & F_1 & F_2 & F_3 & F_4 & F_5 & Z_0 & Z_1 & Z_2 & Z_3 & Z_4 & Z_5 & -X_V
\end{array}\right]
\left[\begin{array}{c} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline -\lambda_0 \\ -\lambda_1 \\ -\lambda_2 \\ -\lambda_3 \\ -\lambda_4 \\ -\lambda_5 \\ -\mu \end{array}\right]
= - \left[\begin{array}{c} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ \hline h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ e_V \end{array}\right].$$

As before, consider a set of leaves $S(i) \subseteq L$ with common parent i, so that $S(j) = \emptyset$ in (35) and

$$x_j = -H_j^{-1}(P_j^* \lambda_j - F_j^* \mu + f_j). \tag{38}$$

After substitution of that expression into the remaining KKT conditions, the resulting modification of (36) yields

$$\lambda_j = -\hat{Y}_j^{-1}(G_j x_i - \hat{Z}_j^* \mu + \hat{h}_j) \tag{39}$$

where $\hat{Y}_j, \hat{Z}_j, \hat{h}_j$ are defined in Table 2. (Positive-definiteness of $\hat{Y}_j$ is an immediate consequence of the rank condition on P_j.) Substituting λ_j completes one recursion step, and the inward recursion again yields a positive definite system $X_\emptyset \mu = -e_\emptyset$ on the empty tree. Node variables λ_j, x_j are then calculated from (39,38) in the outward recursion. The algorithm will also work with any symmetric positive semidefinite blocks in place of X_V, Y_j (more generally, any symmetric blocks leading to positive definite $X_\emptyset, \hat{Y}_j$), and arbitrary rectangular blocks instead of Z_j.

As before, the recursions define a symmetric factorization $\Omega = \Lambda \Pi \Lambda^*$ and corresponding transformation and solution phases whose respective node operations are listed in Table 2.

Formal elimination of λ_j involves the *Schur complement* $\hat{Y}_j$, and each recursion step represents a local version of the Schur complement (SC) method. Therefore we call this algorithm a *tree-sparse SC method*. See [26, 23] for details.

Table 2: Node operations and additional root operations (last line) of the tree-sparse SC method. Factorization and transformation proceed from top to bottom, substitution proceeds in reverse order, producing $-z = \Omega^{-1}a$. Indices on X, e denote the current vertex set; matrix/vector notation neglects possible modifications from earlier recursion steps.

Factorization	Transformation	Substitution
$H_j = L_j L_j^*$		
$\hat{P}_j = P_j L_j^{-*}$		
$\hat{F}_j = F_j L_j^{-*}$	$\hat{f}_j = L_j^{-1} f_j$	$-x_j = \quad L_j^{-*}[$
$\hat{Z}_j = Z_j + \hat{F}_j \hat{P}_j^*$		
$\hat{Y}_j = Y_j + \hat{P}_j \hat{P}_j^*$	$\hat{h}_j = h_j + \hat{P}_j \hat{f}_j$	$\hat{P}_j^* \lambda_j$
$\hat{X}_{V \setminus \{j\}} = X_V + \hat{F}_j \hat{F}_j^*$	$\hat{e}_{V \setminus \{j\}} = e_V - \hat{F}_j \hat{f}_j$	$-\hat{F}_j^* \mu + \hat{f}_j]$
$\hat{Y}_j = \hat{L}_j \hat{L}_j^*$		
$\tilde{Z}_j = \hat{Z}_j \hat{L}_j^{-*}$		
$\tilde{G}_j = \hat{L}_j^{-1} G_j$	$\tilde{h}_j = \hat{L}_j^{-1} \hat{h}_j$	$\lambda_j = \quad \hat{L}_j^{-*}[$
$\tilde{F}_i = F_i + \tilde{Z}_j \tilde{G}_j$		
$\tilde{H}_i = H_i + \tilde{G}_j^* \tilde{G}_j$	$\tilde{f}_i = f_i + \tilde{G}_j^* \tilde{h}_j$	$\tilde{G}_j(-x_i) +$
$X_{V \setminus \{j\}} = \hat{X}_{V \setminus \{j\}} - \tilde{Z}_j \tilde{Z}_j^*$	$e_{V \setminus \{j\}} = \hat{e}_{V \setminus \{j\}} + \tilde{Z}_j \tilde{h}_j$	$\tilde{Z}_j \mu - \tilde{h}_j]$
$X_\emptyset = L_\emptyset L_\emptyset^*$	$\hat{e}_\emptyset = L_\emptyset^{-1} e_\emptyset$	$\mu = \quad L_\emptyset^{-*}(-\hat{e}_\emptyset)$

5 Successive refinement

Successive refinement is a heuristic start-up technique. For a given scenario tree, we consider a sequence of approximate problems on successively smaller trees representing aggregated scenarios. The smallest tree carries a very coarse approximation of the original problem; its solution is used as initial estimate to hot-start the iteration on the next finer tree, and so on until the original problem on the full tree is attacked. (Note that adaptive *construction* of scenario trees is not considered here.) Of course, the whole procedure is unnecessary for the equality-constrained standard QP of Section 3; bound constraints are therefore always included in the following.

Aggregation techniques are employed by Birge [3] to obtain bounds on the optimal value of stochastic linear programs. Wright [28] extends these results significantly using the Lagrangian dual and partially aggregated problems in a general probabilistic setting, and moreover describes disaggregation schemes that yield primal or dual feasible starting points for the simplex method on the original problem. In our context of an infeasible-start primal-dual interior point method, centrality of the starting point is much more important than feasibility, and disaggregation becomes trivial: we always disaggregate primal *and* dual variables and furthermore use the barrier parameter of the (fully) aggregated problem. Subsequent theoretical considerations and the numerical evidence confirm that this procedure indeed provides excellent initial estimates.

We start by outlining the interior point framework, then introduce a specific way of aggregating scenarios and describe its effect on the tree-sparse quadratic programs from a local and a global viewpoint, and finally analyze the initial barrier problem after disaggregation.

5.1 Barrier problems

Consider a general convex QP with lower and upper bounds,

$$\min_y \frac{1}{2} y^* H y + f^* y \qquad \text{s.t.} \qquad Cy + c = 0, \quad y \in [b_l, b_u]. \tag{40}$$

We introduce slacks $s = (s_l, s_u) > 0$ and approximate the QP by a family of barrier problems QP(β) with positive barrier parameter β,

$$\min_{y,s} \quad \frac{1}{2} y^* H y + f^* y - \beta \sum_\nu (\ln s_l^\nu + \ln s_u^\nu) \tag{41}$$

$$\text{s.t.} \qquad Cy + c = 0, \tag{42}$$

$$y - s_l - b_l = 0, \tag{43}$$

$$-y - s_u + b_u = 0. \tag{44}$$

The Lagrangian (with dual slacks $w = (w_l, w_u) > 0$) is

$$\begin{aligned} L(y, s, \lambda, w) \; = \; & \frac{1}{2} y^* H y + f^* y - \beta \sum_\nu (\ln s_l^\nu + \ln s_u^\nu) - \\ & \lambda^*(Cy + c) - w_l^*(y - s_l - b_l) - w_u^*(-y - s_u + b_u), \end{aligned}$$

yielding as KKT conditions of QP(β) *primal feasibility* (42–44), *dual feasibility* $Hy + f - C^* \lambda - w_l + w_u = 0$, and *$\beta$-complementarity* $S_l W_l e = S_u W_u e = \beta e$. As usual, S_l, S_u, W_l, W_u denote the diagonal matrices composed of s_l, s_u, w_l, w_u, respectively, and $e := (1, \ldots, 1)$.

It is well-known that the problems QP(β) have unique solutions which converge to a solution of the original QP as $\beta \to 0$. Relying on that result, interior point methods generate a sequence of approximate solutions to barrier problems with decreasing parameter β. This is all we need for the following.

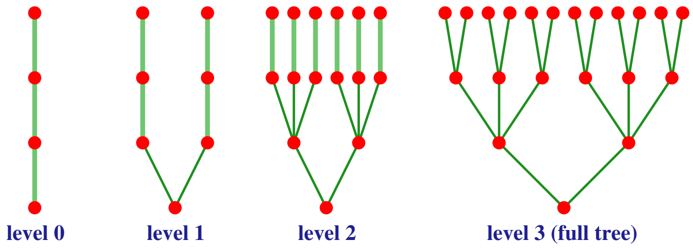

Figure 2: Successive refinement by subtree aggregation on 3 levels.

5.2 Subtree aggregation

Abstractly, aggregation means coarsening the information structure by introducing a *sub-filtration* of $\mathcal{F}$, that is, a filtration $\hat{\mathcal{F}}$ with $\hat{\mathcal{F}}_t \subseteq \mathcal{F}_t$ for each t. Consider as quadratic analogue of the linear problems in [28] the bound-constrained version of (16–18). Aggregation yields a problem of precisely the same form,

$$\min_{\hat{x}} \quad \mathbf{E}\left\{\sum_{t=0}^{T} \frac{1}{2}\hat{x}_t^* \hat{H}_t \hat{x}_t + \hat{f}_t^* \hat{x}_t\right\} \tag{45}$$

$$\text{s.t.} \quad \hat{G}_t \hat{x}_{t-1} + \hat{h}_t = \hat{P}_t \hat{x}_t, \qquad t = 0, \dots, T, \tag{46}$$

$$\mathbf{E}\left\{\sum_{t=0}^{T} \hat{F}_t \hat{x}_t + \hat{e}_t\right\} = 0, \tag{47}$$

$$\hat{x}_t \in [\hat{b}_{lt}, \hat{b}_{ut}], \qquad t = 0, \dots, T, \tag{48}$$

where the QP data are replaced by conditional expectations with respect to the sub-filtration, $\hat{H}_t := \mathbf{E}\{H_t | \hat{\mathcal{F}}_t\}$ etc., and the minimization takes place over $\hat{\mathcal{F}}$-adapted variables $\hat{x}$. This holds for arbitrary distributions and sub-filtrations, and the explicit case (13–15) is completely analogous.

Various possibilities exist in multistage problems, like aggregation over the (local) state space or aggregation of time stages; we aggregate only groups of scenarios that form a subtree. The information structure is thus coarsened by collapsing whole subtrees into single paths. If $j \in L_t$ is the root of such a subtree, then each level set $S^l(j)$ of l-th generation successors, $l = 1, \dots, T - t$, becomes a supernode having the same node probability $\hat{p}_{S^l(j)} = p_j$. Even more specifically, for $t = 0, \dots, T$ we define the *level t aggregation* by collapsing *all* subtrees rooted in L_t, so that level 0 defines the *expected-value problem* and level T the original problem, see Fig. 2. The level t sub-filtration is simply $\hat{\mathcal{F}} = \{\hat{\mathcal{F}}_\tau\}_{\tau=0}^T$ with $\hat{\mathcal{F}}_\tau := \mathcal{F}_{\min(\tau,t)}$.

In the tree-sparse (deterministic) formulation, node probabilites p_j are included in the objective and constraint data H_j, f_j, F_j but not in $G_j, h_j, P_j, b_{lj}, b_{uj}$. Consequently, for each supernode of the level t problem, the latter are averaged as usual,

$$\hat{G}_{S^l(j)} = \sum_{k \in S^l(j)} \frac{p_k}{p_j} G_k$$

and similarly $\hat{h}_{S^l(j)}, \hat{P}_{S^l(j)}, \hat{b}_{lS^l(j)}, \hat{b}_{uS^l(j)}$, whereas conditional expectations of the former are obtained by simple addition,

$$\hat{H}_{S^l(j)} \equiv \hat{p}_{S^l(j)} \left[\frac{\hat{H}_{S^l(j)}}{\hat{p}_{S^l(j)}} \right] = p_j \sum_{k \in S^l(j)} \frac{p_k}{p_j} \left[\frac{H_k}{p_k} \right] \equiv \sum_{k \in S^l(j)} H_k$$

and similarly $\hat{f}_{S^l(j)}, \hat{F}_{S^l(j)}$. QP data up to level t and $\hat{e}_V := e_V$ remain unchanged. Conversely, disaggregation yields approximate solution variables $\bar{x}_j, \bar{\lambda}_j, \bar{\mu}$ of the original problem, where variables up to level t are not modified (including $\bar{\mu} := \hat{\mu}$), and for $k \in S^l(j)$ primal states are simply duplicated whereas dual states are weighted by their respective conditional probabilities,

$$\bar{x}_k := \hat{x}_{S^l(j)}, \qquad \bar{\lambda}_k := \frac{p_k}{p_j} \hat{\lambda}_{S^l(j)}.$$

Thus dual states also include the node probabilities. Again, the explicit case is completely analogous: $H_j^0, H_{ij}, J_j, K_j, d_j, f_j, D_j, F_j$ are added in the supernodes, $G_j, E_j, h_j, b_{lj}, b_{uj}$ are averaged, and the QP data in lower stages remain unchanged. Controls are disaggregated like states, by duplicating.

Interestingly, the different treatment of primal and dual parts of the problem increases the symmetry between the linear transformations of aggregation and disaggregation, as a global view using QP (40) reveals. The transformations can be expressed by primal and dual *aggregation matrices* R_p, R_d whose columns and rows correspond to the nodes before and after aggregation, respectively. (The last row of R_d corresponds to the global constraints.) For instance, the level 0 aggregation matrices of the example tree are

$$R_p = \begin{bmatrix} I & & & & & \\ & I & I & & & \\ & & & I & I & I \end{bmatrix}, \qquad R_d = \begin{bmatrix} I & & & & & & \\ & \frac{p_1}{p_0} I & \frac{p_2}{p_0} I & & & & \\ & & & \frac{p_3}{p_0} I & \frac{p_4}{p_0} I & \frac{p_5}{p_0} I & \\ & & & & & & I \end{bmatrix},$$

where the identity block in column j of R_p has the dimension of (u_j, x_j) in the explicit case and of x_j in the implicit case.

With these transformations one obtains aggregated QP matrices

$$\begin{bmatrix} \hat{H} & \hat{C}^* \\ \hat{C} & \end{bmatrix} = \begin{bmatrix} R_p & \\ & R_d \end{bmatrix} \begin{bmatrix} H & C^* \\ C & \end{bmatrix} \begin{bmatrix} R_p^* & \\ & R_d^* \end{bmatrix}$$

and vectors

$$\begin{bmatrix} \hat{f} \\ \hat{c} \end{bmatrix} = \begin{bmatrix} R_p & \\ & R_d \end{bmatrix} \begin{bmatrix} f \\ c \end{bmatrix}, \qquad (\hat{b}_l, \hat{b}_u) = R_p(b_l, b_u),$$

and disaggregated primal and dual solution estimates

$$\begin{bmatrix} \bar{y} \\ \bar{\lambda} \end{bmatrix} = \begin{bmatrix} R_p^* & \\ & R_d^* \end{bmatrix} \begin{bmatrix} \hat{y} \\ \hat{\lambda} \end{bmatrix}, \qquad (\bar{s}_l, \bar{s}_u, \bar{w}_l, \bar{w}_u) = R_p^*(\hat{s}_l, \hat{s}_u, \hat{w}_l, \hat{w}_u).$$

5.3 Disaggregation in barrier problems

Some elementary results can now be proved to support the intuition that successive refinement yields good starting points for the interior point method. For that purpose the barrier problem QP(β) and a level t aggregation $\widehat{\mathrm{QP}}(\beta)$ are considered, where the underlying QP is assumed to have an (explicit or implicit) tree-sparse structure.

Consider a solution $\hat{z} = (\hat{y}, \hat{s}, -\hat{\lambda}, -\hat{w})$ of the level t barrier problem $\widehat{\mathrm{QP}}(\beta)$, determined by the KKT conditions

$$\begin{aligned} \hat{H}\hat{y} + \hat{f} - \hat{C}^*\hat{\lambda} - \hat{w}_l + \hat{w}_u &= 0, \\ \hat{S}_l \hat{W}_l e - \beta e &= 0, \\ \hat{S}_u \hat{W}_u e - \beta e &= 0, \\ \hat{C}\hat{y} + \hat{c} &= 0, \\ \hat{y} - \hat{s}_l - \hat{b}_l &= 0, \\ -\hat{y} - \hat{s}_u + \hat{b}_u &= 0, \end{aligned}$$

and the disaggregation $\bar{z} = (\bar{y}, \bar{s}, -\bar{\lambda}, -\bar{w})$. We form an auxiliary problem $\overline{\mathrm{QP}}(\beta)$ by replacing each matrix block and each vector of the original problem by its conditional expectation with respect to $\hat{\mathcal{F}}$. Thus the auxiliary problem has the large size of QP(β) but the coarse information structure $\hat{\mathcal{F}}$ of $\widehat{\mathrm{QP}}(\beta)$, containing many unnecessary identical scenarios. The following statement is obvious.

Lemma 1 *$\overline{QP}(\beta)$ and $\widehat{QP}(\beta)$ are equivalent in the sense that the solution of $\overline{QP}(\beta)$ is precisely the disaggregated solution of $\widehat{QP}(\beta)$.*

One can now form the "difference" of the original problem to the aggregated problem: $\Delta H := H - \bar{H}$, $\Delta f := f - \bar{f}$, etc., are the deviations of the QP data from their conditional expectations, giving a quality measure for the approximation. These are used to analyze the *residual error* at the start of the iteration on the original barrier problem,

$$r := \left[\begin{array}{l} H\bar{y} + f - C^*\bar{\lambda} - \bar{w}_l + \bar{w}_u \\ \bar{S}_l \bar{W}_l e - \beta e \\ \bar{S}_u \bar{W}_u e - \beta e \\ \hline C\bar{y} + c \\ \bar{y} - \bar{s}_l - b_l \\ -\bar{y} - \bar{s}_u + b_u \end{array}\right]. \tag{49}$$

Theorem 1 *The starting point $\bar{z}$ obtained from an exact solution $\hat{z}$ of $\widehat{QP}(\beta)$ has the following properties.*

(a) $\bar{z}$ satisfies the β-complementarity condition of QP(β).

(b) The residual r has conditional expectation zero with respect to $\hat{\mathcal{F}}$.

(c) The residual has Euclidian norm-square

$$\|r\|_2^2 = \left\| \begin{pmatrix} \Delta H & \Delta C^* \\ \Delta C & \end{pmatrix} \begin{pmatrix} \bar{y} \\ -\bar{\lambda} \end{pmatrix} + \begin{pmatrix} \Delta f \\ \Delta c \end{pmatrix} \right\|_2^2 + \|\Delta b_l\|_2^2 + \|\Delta b_u\|_2^2.$$

In specific, if $\Delta H = 0$ and $\Delta C = 0$, this reduces to the conditional variance of the QP vectors, $\|r\|_2^2 = \sigma_{\hat{\mathcal{F}}}^2(\Delta f, \Delta c, \Delta b_l, \Delta b_u)$.

(d) If $\Delta b_l = \Delta b_u = 0$, then $\bar{z} \in [b_l, b_u]$.

Proof. Statement (d) is trivial (but practically important, see below). The β-complementarity conditions in (49) depend only on the current slacks $\bar{s}$ and $\bar{w}$ but not on the QP data, which proves (a) and yields the initial residual

$$r = \left[\begin{array}{l} H\bar{y} + f - C^*\bar{\lambda} - \bar{w}_l + \bar{w}_u \\ 0 \\ 0 \\ \hline C\bar{y} + c \\ \bar{y} - \bar{s}_l - b_l \\ -\bar{y} - \bar{s}_u + b_u \end{array}\right].$$

Subtracting the residual of the auxiliary problem (which is zero by Lemma 1) yields

$$r = \begin{bmatrix} \Delta H\bar{y} + \Delta f - \Delta C^*\bar{\lambda} \\ 0 \\ 0 \\ \hline \Delta C\bar{y} + \Delta c \\ -\Delta b_l \\ \Delta b_u \end{bmatrix}.$$

Statements (b) and (c) are immediate consequences of this representation. □

Corollary 1 *As in Section 4, let* $z := (y, -\lambda)$ *and define* Ω, a *accordingly. Then*

$$\begin{aligned} \|r\|_1 &= \|\Delta\Omega\bar{z} + \Delta a\|_1 + \|\Delta b_l\|_1 + \|\Delta b_u\|_1 \\ &\le \|\Delta\Omega\|_1\|\bar{z}\|_1 + \|\Delta a\|_1 + \|\Delta b_l\|_1 + \|\Delta b_u\|_1 \end{aligned}$$

and

$$\begin{aligned} \|r\|_\infty &= \max\{\|\Delta\Omega\bar{z} + \Delta a\|_\infty, \|\Delta b_l\|_\infty, \|\Delta b_u\|_\infty\} \\ &\le \max\{\|\Delta\Omega\|_\infty\|\bar{z}\|_\infty, \|\Delta a\|_\infty, \|\Delta b_l\|_\infty, \|\Delta b_u\|_\infty\}. \end{aligned}$$

Some remarks are in order. Statement (a) of the theorem proves the desired property that $\bar{z}$ is as close to the central path as possible, that is, as close as primal and dual infeasibilities permit. If both vanish, the distance is zero. Statements (b) and (c) describe the statistical properties of the residual and its precise dependence on the QP data and $\bar{z}$. Statement (d) shows that the starting point satisfies the bounds if only groups of scenarios with identical bounds are aggregated, which holds trivially in the important case of deterministic (scenario-independent) bounds. Hence, if the large original problem is infeasible, this is already detected on the small aggregated problem. Observe also that only the β-complementarity conditions are nonlinear. Therefore all other KKT conditions are satisfied in the next iterate if the interior point method takes a full step.

The estimates in the corollary (based on computationally inexpensive norms) may be of interest in *adaptive refinement* procedures, under suitable conditions even for continuous probability distributions in the original problem.

Of course, $\hat{z}$ is not exact in practice and the previous results hold only in an approximate sense, but this is not really significant since a highly accurate solution of the small aggregated problem is relatively inexpensive.

Table 3: 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 (≈ 207 MB).

QP form	periods	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

6 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 scenarios and $(9^{T+1} - 1)/8$ nodes for the T-period problem. Stochastic test data, supplied by the Institute of Operations Research, University of St. Gallen, are generated from an n-dimensional Brownian motion of the asset prices, resulting in a multivariate lognormal distribution of the returns, see [13]. Benchmark runs are performed on a Silicon Graphics O2 workstation with a 175 MHz R10000 processor. All codes are implemented in C++. The recursive algorithms treat all matrix blocks as dense and node-dependent.

6.1 KKT recursion

We construct problems with up to five periods, and two additional six-period problems that fill all available memory (with stage-dependent numbers of branches). Table 3 lists the sizes of KKT systems and the CPU times in seconds. These data clearly demonstrate the efficiency of the recursions.

6.2 Successive refinement

Table 4 lists iteration numbers and CPU times for the bound-constrained example problem with and without successive refinement. The bounds are nonnegativity constraints on all asset values x_j^ν. Problems with $T = 3, 4, 5$ are considered, and successive refinement proceeds from the level 0 problem (1 scenario) over the

Table 4: Comparison of interior point convergence with refinement levels $0/2/T$ versus full tree.

QP form	periods	variables	iteration numbers		computing times (in seconds)		
	T		level $0/2/T$	full	level $0/2/T$	total	full
explicit	3	12,300	8/1/2	35	0.01/0.12/1.39	1.79	20.2
explicit	4	110,715	8/2/2	60	0.01/0.33/14.7	17.3	358
implicit	3	6,560	8/1/2	35	0.01/0.03/0.39	0.50	5.58
implicit	4	59,048	8/2/2	60	0.01/0.09/5.54	6.37	126
implicit	5	531,440	8/3/3	101	0.01/0.18/71.8	78.9	2110

level 2 problem (81 scenarios) to the original level T problem. Total times include the overhead for aggregation and disaggregation. The tolerance for the interior point method is set to 10^{-10}.

Initial estimates are generated by solving the unconstrained QP (with regularized H if necessary) and then multiplying all negative asset values by $-\beta$. Primal and dual slacks are set to $s_l = y$ and $w_l = \beta S_l^{-1} e$, respectively, and dual states are initialized to zero. This is clearly a simple heuristic, and convergence on the full tree problem could probably be improved by a better strategy. The important point is that successive refinement is insensitive to the initialization. The very small level 0 problem in our example is always solved in 8 iterations taking 0.01 seconds, and even twice as many iterations due to a poor start would be insignificant. As a result of the hot start, all subsequent problems converge in just a few iterations (at most three in our example) so that the largest problem with more than half a million variables can be solved in less than 80 seconds.

To give another impression of the performance of our method, consider as competitor any scenario decomposition approach, such as augmented Lagrangian decomposition [16, 2] or progressive hedging [20]. Then individual scenarios are optimized and the nonanticipativity condition is solved iteratively. The scenario subproblems are modified in each iteration to include updated dual information and quadratic regularization terms, but inequality constraints of the original problem are still present. Assume that these modified scenario subproblems are solved by the interior point method used in our approach, and that modifications do not affect the computing time. Then, on the largest problem above (implicit QP, 59059 scenarios), checking convergence for all subproblems takes about 10 seconds, and performing one iteration for all subproblems takes 40 seconds. Hence, even in the very unlikely situation that scenario subproblems can be solved in only two iterations on average, the dual approach needs 90 seconds to solve each of them just once, while our method is already completely finished!

7 Conclusions

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 [23]: we treat nonlinearities and inequality constraints *globally* by suitable iterative algorithms (interior point or SQP methods), and move most of the computational work to the *linear algebra* level where the problem-inherent (sparse) structure is exploited. 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, successive refinement provides a natural means to enhance convergence of the global iteration on inequalities by exploiting nonlinear stochastic characteristics of the problem. Theoretical and numerical results demonstrate that it works ideally together with the primal-dual interior point method, yielding a significant speed-up even for our simple variant based on level-wise subtree aggregation. More sophisticated adaptive refinement techniques using the same concept may be developed in the future.

Our modular approach 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.

Although only the basic recursive structure of dynamics has been treated in this work, we have indicated the possible types of *local* constraints. With these 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 each class 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 or P_j are identical in all nodes. This can be used to save CPU time and memory simply by specializing the node operations; a commercial code with these characteristics is currently being developed. The more difficult combination of sparse blocks and 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 [22]: the development of specialized methods and high quality software for certain application areas.

Acknowledgements

The author gratefully acknowledges an inspiring cooperation with K. Frauendorfer and H. Siede, who also supplied the stochastic test data.

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(4):735–753, 1994.

[3] J. R. Birge. Aggregation bounds in stochastic linear programming. *Math. Programming*, 31:25–41, 1985.

[4] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Oper. Res.*, 33(5):989–1007, 1985.

[5] J. R. Birge, M. A. H. Dempster, H. I. Gassmann, E. A. Gunn, A. J. King, and S. W. Wallace. A standard input format for multiperiod stochastic linear programs. *COAL newsletter*, (17):1–19, Dec. 1987.

[6] 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*, 75:327–352, 1996.

[7] 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(4):474–490, 1995.

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

[9] 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(4):386–401, 1995.

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

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

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

[13] R. M. Jones and K. S. Miller. On the multivariate lognormal distribution. *The Journal of the Industrial Mathematics Society*, 15:63–76, 1966.

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

[15] J. M. Mulvey. Nonlinear network models in finance. *Adv. Math. Progr. Fin. Planning*, 1:253, 1987.

[16] J. M. Mulvey and A. Ruszczyński. A new scenario decomposition method for large-scale stochastic optimization. *Oper. Res.*, 43(3):477–490, 1995.

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

[18] J. M. Mulvey and H. Vladimirou. Applying the progressive hedging algorithm to stochastic generalized networks. *Annals Oper. Res.*, 31:399–424, 1991.

[19] 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(4):810–822, 1990.

[20] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.*, 16:119–147, 1991.

[21] A. Ruszczyński. Parallel decomposition of multistage stochastic programming problems. *Math. Programming*, 58:201–228, 1993.

[22] A. Ruszczyński. Decomposition methods in stochastic programming. *Math. Programming*, 79(1–3):333–353, Oct. 1997. Invited lectures of the 16th International Symposium on Mathematical Programming, Lausanne EPFL.

[23] M. C. Steinbach. *Fast Recursive SQP Methods for Large-Scale Optimal Control Problems*. Ph. D. dissertation, University of Heidelberg, 1995.

[24] M. C. Steinbach. Structured interior point SQP methods in optimal control. *Z. Angew. Math. Mech.*, 76(S3):59–62, 1996.

[25] M. C. Steinbach. Recursive direct algorithms for multistage stochastic programs in financial engineering. In P. Kall and H.-J. Lüthi, editors, *Operations Research — Ongoing Progress*, pages 236–245. Springer Verlag, 1998. To appear. Also ZIB Preprint SC-98-23.

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

[27] 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(4):303–340, 1998.

[28] S. E. Wright. Primal-dual aggregation and disaggregation for stochastic linear programs. *Math. Oper. Res.*, 19(4):893–908, 1994.