Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MARC C. STEINBACH

Hierarchical Sparsity in Multistage Convex Stochastic Programs

00-15 (May 2000)

HIERARCHICAL SPARSITY IN MULTISTAGE CONVEX STOCHASTIC PROGRAMS

MARC C. STEINBACH

ABSTRACT. Interior point methods for multistage stochastic programs involve KKT systems with a characteristic global block structure induced by dynamic equations on the scenario tree. We generalize the recursive solution algorithm proposed in an earlier paper so that its linear complexity extends to a refined tree-sparse KKT structure. Then we analyze how the block operations can be specialized to take advantage of problem-specific sparse substructures. Savings of memory and operations for a financial engineering application are discussed in detail.

1. INTRODUCTION

Uncertainty of future events plays an essential role in many practical decision processes. The mathematical modeling of such planning problems leads to various kinds of *multistage stochastic programs* among which smooth *convex* ones represent a widely useful class, with linear or convex quadratic stochastic programs as special cases. This paper is concerned with the efficient numerical treatment of such inherently large-scale problems when stochastic influences are modeled by a scenario tree.

Well-known numerical approaches include primal decomposition methods [4, 10, 15], dual decomposition methods [16, 20], and interior point methods [2, 8, 23]; for a more exhaustive overview see [5, 22]. In any case the key to success lies in taking advantage of the characteristic problem structure. This is achieved by decomposition into node or scenario subproblems (primal and dual decomposition) or via special factorizations of the KKT systems in interior point methods. In addition, each approach offers a substantial degree of inherent parallelism [3, 7, 9, 13, 17, 21].

In [26, 28] we have proposed an interior point approach where the KKT system is reinterpreted as a linear-quadratic control problem. This view is focused on the inherent *dynamic* structure and its control-theoretic interpretation. It entails a natural classification of constraints and leads directly to a recursive factorization of the block-sparse KKT matrix. Thus we take full advantage of the generic *global* structure.

The key role of the dynamic equations is also pointed out in [18, 19] where non-smooth convex programs are considered from a more abstract viewpoint emphasizing duality. These problem classes share many similarities with ours, but the required numerical techniques are quite different.

Our approach generalizes similar methods that have proved successful in deterministic nonlinear trajectory optimization by direct SQP methods [24, 25, 29]. The available SQP code is directly applicable to non-convex stochastic programs as well, but for simplicity we restrict ourselves to linearly constrained convex problems in this paper. Also for simplicity, we choose an *implicit* form of dynamics. Such formulations are especially common in financial problems; in the control literature

1991 *Mathematics Subject Classification.* 90C15.

Key words and phrases. Multistage Stochastic Programs, Hierarchical KKT Sparsity.

(including [18, 19]) one usually finds *explicit* dynamics where control and state variables are distinguished. A brief comparison of both variants is given in [28].

This paper extends earlier work in the following ways. We refine the class of multistage stochastic programs considered in [26, 28] by adding *local* (equality and range) constraints as separate categories. Correspondingly, range constraints are included in the description of the interior point framework, and the recursive solution algorithm is generalized to cope with the refined structure. In these parts we treat the theoretical background more rigorously by stating precise regularity assumptions and proving the existence of the KKT matrix factorization. Finally we discuss how problem-specific *local* sparsity can be exploited within this generic framework.

The paper is organized as follows. In Sect. 2 we introduce some notation and present the general problem class. The interior point framework is outlined in Sect. 3, followed by the generic solution algorithm in Sect. 4 where the existence of the KKT matrix factorization is proved. Sect. 5 is devoted to the issue of local sparsity, which is discussed in detail for an asset management example. Sect. 6 gives some conclusions and future directions of research.

2. Problem class

2.1. **General convex programs.** Let us first take a global viewpoint and disregard any specific problem substructure. We consider a linearly constrained smooth convex program (CP) with lower and upper bound and range inequalities,

$$(1)\qquad \min_x \varphi(x) \qquad \text{s.t.} \qquad Ax + a = 0, \quad Bx \in [r_l, r_u], \quad x \in [b_l, b_u],$$

where $\varphi \in C^2(\mathbf{R}^n, \mathbf{R})$ with $\nabla^2\varphi(x) \geq 0\ \forall x \in \mathbf{R}^n$, and $A \in \mathbf{R}^{l\times n}$, $B \in \mathbf{R}^{k\times n}$, $l \leq n$.
Notational convention. The values $\pm\infty$ are formally allowed to indicate the absence of upper and/or lower limit components b_l^ν, b_u^ν, r_l^κ, r_u^κ. Rigorously this means there are index sets $\mathcal{B}_l, \mathcal{B}_u \subseteq \{1,\dots,n\}$, $\mathcal{R}_l, \mathcal{R}_u \subseteq \{1,\dots,k\}$ such that

$$P_{\mathcal{B}_l} b_l \leq P_{\mathcal{B}_l} x, \quad P_{\mathcal{B}_u} x \leq P_{\mathcal{B}_u} b_u, \qquad P_{\mathcal{R}_l} r_l \leq P_{\mathcal{R}_l} Bx, \quad P_{\mathcal{R}_u} Bx \leq P_{\mathcal{R}_u} r_u.$$

Here $P_\mathcal{B} \in \mathbf{R}^{|\mathcal{B}|\times n}$ denotes the gather matrix that selects the components $P_\mathcal{B} x \equiv x_\mathcal{B}$ specified by $\mathcal{B} = \{\nu_1, \dots, \nu_{|\mathcal{B}|}\}$. The associated scatter matrix is $P_\mathcal{B}^* \in \mathbf{R}^{n\times|\mathcal{B}|}$ so that $P_\mathcal{B} P_\mathcal{B}^* = I$ on $\mathbf{R}^{|\mathcal{B}|}$. Similarly we have $P_\mathcal{R} \in \mathbf{R}^{|\mathcal{R}|\times k}$ for the range constraints.
Regularity conditions. Denote by $\mathcal{F} := \{x \in [b_l, b_u] \colon Ax + a = 0,\ Bx \in [r_l, r_u]\}$ the feasible set, a closed convex polyhedron, and let $\mathcal{F}_{\text{eq}} := \{x \in \mathbf{R}^n \colon Ax + a = 0\}$. Throughout the paper we make the following assumptions.

(A0) $\mathcal{F}$ has nonempty relative interior with respect to the affine subspace $\mathcal{F}_{\text{eq}}$.
(A1) A has full rank. (Equivalently $N(A^*) = \{0\}$ since $l \leq n$.)
(A2) $\nabla^2\varphi(x)|\mathcal{N} \geq \epsilon I > 0$ for $x \in \mathcal{F}$, where $\mathcal{N}$ is the null space

$$\mathcal{N} := N(A) \cap N(P_{\mathcal{B}_l\cup\mathcal{B}_u}) \cap N(P_{\mathcal{R}_l\cup\mathcal{R}_u} B).$$

These conditions are tailored toward the barrier problems considered below; they do not imply existence or uniqueness of solutions for (1) since strong convexity (A2) is only required on the largest space of feasible directions *inside* the recession cone of $\mathcal{F}$. The CP may have multiple solutions ($\min_{x\geq 0} 0$) or a finite infimum that is not attained ($\min_{x\geq 0} e^{-x}$), or it may be unbounded ($\min_{x\geq 0} -x$). If a solution exists, however, then they guarantee that solutions will also exist for slightly perturbed problems, and that the solution is stable *if* it is unique. (By standard results in convex optimization, each solution of the CP is a global minimum, and the set $\mathcal{S}$ of all such solutions is convex.) We are primarily interested in the case where $\mathcal{S}$ is nonempty and bounded (hence compact) which is guaranteed under an additional growth condition,

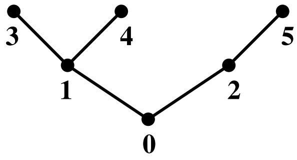

FIGURE 1. A small tree.

(A3) $\varphi(x^{(k)}) \to \infty$ for every sequence $x^{(k)} \in \mathcal{F}$ with $\|x^{(k)}\| \to \infty$.
This holds, for instance, if $\mathcal{F}$ is bounded or if $\mathcal{N}$ in (A2) is replaced by the larger space $\mathcal{N}' := N(A) \cap N(P_{\mathcal{B}_l \cap \mathcal{B}_u}) \cap N(P_{\mathcal{R}_l \cap \mathcal{R}_u} B)$ which *contains* the recession cone of $\mathcal{F}$.

2.2. **Multistage convex stochastic programs.** The problem class of interest models multistage decision processes under uncertainty. Considering a finite planning horizon in discrete time, $t = 0, 1, \ldots, T$, we assume that the underlying random data process has only finitely many outcomes so that the information structure can be represented by a scenario tree. Let $j \in L_t$ denote the outcomes at time t, with node probabilities $p_j > 0$. The root is $0 \in L_0$, the parent of $j \in L_t$ is $i \equiv \pi(j) \in L_{t-1}$, and the set of successors is $S(j) \subseteq L_{t+1}$. Finally we denote by $L := L_T$ the set of leaves, each representing a scenario, and by $V := \bigcup_{t=0}^{T} L_t$ the set of all nodes. Subsequently we will often use $V = \{0, \ldots, N\}$, where the node labeling is assumed to be increasing, i.e., $\pi(j) < j$ for all $j \in V \setminus \{0\}$, see Fig. 1.

The problem class treated in this paper is a specifically structured subclass of (1): it consists of *multistage convex stochastic programs (MCSP)* in the general form

$$\min_{x} \quad \sum_{j \in V} p_j \varphi_j(x_j) \tag{2}$$

$$\text{s.t.} \quad G_j x_i + h_j = P_j x_j \quad \forall j \in V, \tag{3}$$

$$F_j^x x_j + e_j^x = 0 \quad \forall j \in V, \tag{4}$$

$$F_j^r x_j \in [r_{lj}, r_{uj}] \quad \forall j \in V, \tag{5}$$

$$x \in [b_l, b_u], \tag{6}$$

$$\sum_{j \in V} p_j F_j x_j + e_V = 0. \tag{7}$$

Here the local dimensions are $x_j \in \mathbf{R}^{n_j}$, $G_j, P_j \in \mathbf{R}^{l_j \times n_j}$, $F_j^x \in \mathbf{R}^{l_j^x \times n_j}$, $F_j^r \in \mathbf{R}^{l_j^r \times n_j}$, $F_j \in \mathbf{R}^{m \times n_j}$, where $l_j \le n_j$, $l_j^x \le n_j - l_j$, and $m \le \sum_{j \in V}(n_j - l_j - l_j^x)$. Apart from the node-wise separable convex objective (2) and implicit dynamic equations (3), there are *local* (equality) constraints (4), *range* (inequality) constraints (5), *bound* (inequality) constraints (6), and *global* (equality) constraints (7).

With $x = (x_0, \ldots, x_N)$, the equality and inequality constraint matrices of (1) can be written

$$A = \begin{bmatrix} G \\ F^x \\ F \end{bmatrix}, \qquad B = \operatorname{Diag}(F_0^r, \ldots, F_N^r) \equiv \begin{bmatrix} F_0^r & & \\ & \ddots & \\ & & F_N^r \end{bmatrix}, \tag{8}$$

where $F^x = \operatorname{Diag}(F_0^x, \ldots, F_N^x)$, $F = \begin{pmatrix} p_0 F_0 & \ldots & p_N F_N \end{pmatrix}$, and G represents the dynamics for the given tree topology. In case of the example tree in Fig. 1, for instance, we have

$$G = \begin{bmatrix} -P_0 & & & & & \\ G_1 & -P_1 & & & & \\ G_2 & & -P_2 & & & \\ & G_3 & & -P_3 & & \\ & G_4 & & & -P_4 & \\ & & G_5 & & & -P_5 \end{bmatrix}. \tag{9}$$

Regularity conditions. Assumption (A0) is kept for MCSP but (A1,A2) are refined in a hierarchical fashion matching the recursive dynamic structure.

(A1.1) $\forall j \in V$: F^x_j has full rank ($= l^x_j$).

(A1.2) $\forall j \in V$: $P_j | N(F^x_j)$ has full rank ($= l_j$).

(A1.3) $F|(N(F^x) \cap N(G))$ has full rank ($= m$).

(A2*) $\forall x \in \mathcal{F}$: $\forall j \in V$: $\nabla^2 \varphi_j(x_j) | \mathcal{N}_j \geq \epsilon_j I > 0$ where

$$\mathcal{N}_j := N(P_{\mathcal{B}_{lj} \cup \mathcal{B}_{uj}}) \cap N(P_{\mathcal{R}_{lj} \cup \mathcal{R}_{uj}} F^r_j) \cap N(F^x_j) \cap \bigcap_{k \in S(j)} N(G_k).$$

For later reference we state some elementary consequences of these conditions without proof; less apparent implications will be discussed below.

Lemma 1. *The following properties hold.*

(i) Condition (A1.1) is equivalent to full rank of F^x.

(ii) Condition (A1.2) implies full rank of $G|N(F^x)$*; the converse is not true.*

(iii) Conditions (A1.1,A1.2) together imply full rank of $\binom{F^x}{G}$.

(iv) Conditions (A1.1–A1.3) together imply full rank of A*, i.e., (A1).*

Discussion. The key property of our approach is the classification of equality and inequality constraints along with the formulation of appropriate regularity conditions. That way dynamic equations on the scenario tree induce the characteristic global block structure seen above, which we call *tree-sparse* for obvious reasons. It refines the subclass of tree-sparse problems *without* local constraints (4,5) as considered in [26, 28]. The intention in choosing this specific structure is to cover a large problem class whose KKT systems in an interior point method permit the direct solution by a natural recursive algorithm having linear complexity in the tree size. Below, the basic version of the algorithm described in [26, 28] will be generalized accordingly.

Multistage stochastic programs are one of the most important classes of tree-sparse problems but actually not the only one. Another important (if much narrower) subclass, for instance, are multi-body systems in descriptor form, where the spatial system topology is represented by a spanning tree plus kinematic loops modeled as global constraints. (However, multi-body dynamics are usually formulated in explicit form, see, e.g., [12] or [24, 29].)

Generality of MCSP.. Node-wise separability of the objective (which holds typically in practice) is essentially the only structural restriction in MCSP. For instance, every multistage linear stochastic program in standard form [6] can be cast into our framework as indicated in [26]. The classification of constraints increases both insight and the potential for structure exploitation in numerical algorithms.

Generally, any (node-wise separable) convex program defined on a tree can be reformulated as an MCSP, roughly in the following way: Each inequality constraint is either a simple bound or a local range constraint, or else converted to an equality constraint by introducing a bounded surplus variable. Each of the resulting equality constraints is then either a local constraint, a dynamic equation (if it couples a node j with its parent $\pi(j)$, and P_j has full rank on the already identified $N(F^x_j)$), or else classified as a global constraint. The regularity condition on P_j is essential here and always satisfied by the rule above. Other violations of (A1.1–A2*) indicate either modeling errors or specific problem characteristics that require special attention, as in [27], e.g. (Rank deficiencies can sometimes be tolerated in the latter case.) Violations of (A1.1–A2*) are detected by our algorithm and moreover precisely located, except for the global condition (A1.3) which has no "location". (Of course, (A2*) is only checked in the current point x.)

3. Interior point approach

In this section we describe the generic approach for the CP (1). For simplicity assume first that all limits are actually present, $\mathcal{B}_l = \mathcal{B}_u = \{1, \ldots, n\}$, $\mathcal{R}_l = \mathcal{R}_u = \{1, \ldots, k\}$. We introduce strictly positive slacks for the limits, $s = (s_l, s_u) > 0$, $t = (t_l, t_u) > 0$, and approximate the CP by a family of standard logarithmic barrier problems CP(β) with positive barrier parameter β,

$$\min_{x,s,t} \quad \varphi(x) - \beta \sum_{\nu} (\ln s_l^{\nu} + \ln s_u^{\nu}) - \beta \sum_{\kappa} (\ln t_l^{\kappa} + \ln t_u^{\kappa}) \tag{10}$$

$$\text{s.t.} \quad Ax + a = 0, \tag{11}$$

$$Bx - t_l - r_l = 0, \tag{12}$$

$$-Bx - t_u + r_u = 0, \tag{13}$$

$$x - s_l - b_l = 0, \tag{14}$$

$$-x - s_u + b_u = 0. \tag{15}$$

The barrier Lagrangian (with equality multiplier z and dual slacks $u = (u_l, u_u) > 0$, $v = (v_l, v_u) > 0$) reads

$$\begin{aligned} L(x,s,t,z,u,v;\beta) \quad = \quad & \varphi(x) - \beta \textstyle\sum_{\nu} (\ln s_l^{\nu} + \ln s_u^{\nu}) - \beta \textstyle\sum_{\kappa} (\ln t_l^{\kappa} + \ln t_u^{\kappa}) - \\ & z^*(Ax + a) - u_l^*(x - s_l - b_l) - u_u^*(-x - s_u + b_u) - \\ & v_l^*(Bx - t_l - r_l) - v_u^*(-Bx - t_u + r_u). \end{aligned}$$

As KKT conditions of CP(β) this yields *primal feasibility* (11–15), *dual feasibility* $\nabla\varphi(x) - A^*z - B^*(v_l - v_u) - (u_l - u_u) = 0$, and nonlinear *$\beta$-complementarity* conditions $S_l U_l e = S_u U_u e = \beta e$, $T_l V_l e = T_u V_u e = \beta e$. Here S_l etc. denote the diagonal matrices composed of s_l etc., and $e := (1, \ldots, 1)$ in the appropriate dimension.

Interior point methods (of infeasible-start primal-dual type) solve these conditions iteratively while simultaneously reducing β. Each Newton step involves a very large and sparse KKT system (of dimension $n + l + 4(n + k)$). However, primal and dual slacks can be pre-eliminated by simple operations, cf. [24, 26], leading to a *reduced* KKT system of the form

$$\left[\begin{array}{c|c} \bar{H} & A^* \\ \hline A & \end{array}\right] \left[\begin{array}{c} \Delta x \\ \hline -\Delta z \end{array}\right] + \left[\begin{array}{c} \bar{f} \\ \hline a \end{array}\right] = 0. \tag{16}$$

Here $\bar{H} := H + \Phi + B^*\Psi B$ is formed from the current Hessian $H := \nabla^2\varphi(x)$ and the positive definite diagonal matrices

$$\Phi := \Phi_l + \Phi_u \equiv S_l^{-1}U_l + S_u^{-1}U_u, \qquad \Psi := \Psi_l + \Psi_u \equiv T_l^{-1}V_l + T_u^{-1}V_u,$$

and $\bar{f}$ is the corresponding modification of the objective gradient, $f := \nabla\varphi(x)$. (Thus $\frac{1}{2}\Delta x^* H \Delta x + f^*\Delta x$ is the second-order Taylor approximation to $\varphi(x + \Delta x) - \varphi(x)$.)

General case. For absent bounds or ranges one may formally set primal and dual slacks to $+\infty$ and zero, respectively, which means that all resulting terms are omitted or replaced by zero. Rigorously one selects relevant components by $P_{\mathcal{B}_l}$ etc., which gives the following modifications due to $\mathcal{B}_l$ (and similar ones due to $\mathcal{B}_u, \mathcal{R}_l, \mathcal{R}_u$).

The slacks satisfy only $P_{\mathcal{B}_l} s_l > 0$, $P_{\mathcal{B}_l} u_l > 0$, the barrier terms are $\sum_{\nu \in \mathcal{B}_l} \ln s_l^\nu$, and equation (14) becomes $P_{\mathcal{B}_l}(x - s_l - b_l) = 0$. In the Lagrangian, this gives the scalar product $u_l^* P_{\mathcal{B}_l}^* P_{\mathcal{B}_l}(x - s_l - b_l)$, where $P_{\mathcal{B}_l}^* P_{\mathcal{B}_l} \in \mathbf{R}^{n \times n}$ is the orthogonal projection into the $\mathcal{B}_l$-subspace. Thus dual feasibility requires

$$\nabla\varphi(x) - A^* z - B^*(P_{\mathcal{R}_l}^* P_{\mathcal{R}_l} v_l - P_{\mathcal{R}_u}^* P_{\mathcal{R}_u} v_u) - (P_{\mathcal{B}_l}^* P_{\mathcal{B}_l} u_l - P_{\mathcal{B}_u}^* P_{\mathcal{B}_u} u_u) = 0,$$

and β-complementarity reads $P_{\mathcal{B}_l} S_l U_l e = P_{\mathcal{B}_l} \beta e$. Finally we define

$$\Phi_l := P_{\mathcal{B}_l}^* \bar{\Phi}_l P_{\mathcal{B}_l} \geq 0 \quad \text{where} \quad \bar{\Phi}_l := (P_{\mathcal{B}_l} S_l P_{\mathcal{B}_l}^*)^{-1} (P_{\mathcal{B}_l} U_l P_{\mathcal{B}_l}^*) > 0.$$

(Conversely, this gives $\bar{\Phi}_l = P_{\mathcal{B}_l} \Phi_l P_{\mathcal{B}_l}^*$.) If no limits at all are present, then the interior point method essentially specializes to a direct Newton iteration on the optimality conditions of the original CP.

Lemma 2. *The reduced KKT matrix (16) is non-singular.*

Proof. Since A has full rank by (A1), we just have to show positive definiteness of the restriction $\bar{H}|N(A)$. By definition,

$$\bar{H} = H + P_{\mathcal{B}_l}^* \bar{\Phi}_l P_{\mathcal{B}_l} + P_{\mathcal{B}_u}^* \bar{\Phi}_u P_{\mathcal{B}_u} + B^*(P_{\mathcal{R}_l}^* \bar{\Psi}_l P_{\mathcal{R}_l} + P_{\mathcal{R}_u}^* \bar{\Psi}_u P_{\mathcal{R}_u}) B.$$

Each term in this sum is positive semidefinite, and $\bar{\Phi}_l, \bar{\Phi}_u, \bar{\Psi}_l, \bar{\Psi}_u$ are positive definite. Hence,

$$\begin{aligned} N(\bar{H}) &= N(H) \cap N(P_{\mathcal{B}_l}) \cap N(P_{\mathcal{B}_u}) \cap N(P_{\mathcal{R}_l} B) \cap N(P_{\mathcal{R}_u} B) \\ &= N(H) \cap N(P_{\mathcal{B}_l \cup \mathcal{B}_u}) \cap N(P_{\mathcal{R}_l \cup \mathcal{R}_u} B). \end{aligned}$$

Now $N(\bar{H}|N(A)) = N(\bar{H}) \cap N(A) = N(H) \cap \mathcal{N} = \{0\}$ by (A2), as required. □

All the barrier problems CP(β) are obviously convex and have the same feasible set (with respect to x), the relative interior $\operatorname{int}(\mathcal{F}) \subseteq \mathcal{F}_{\text{eq}}$ which is nonempty by (A0).

Lemma 3. *If the barrier problem CP(β) has a solution $(x_0, s_0, t_0, z_0, u_0, v_0)$, then it is unique (up to irrelevant slack components).*

Proof. Let $(x_1, s_1, t_1, z_1, u_1, v_1)$ also be a solution. Since CP(β) is convex, the entire line segment between the two solutions is optimal, and their difference vector lies in the null space of the Hessian of the barrier objective (10) at $(x_0, \ldots, v_0)$,

$$\operatorname{Diag}(\nabla^2\varphi(x_0), \beta P_{\mathcal{B}_l}^* (P_{\mathcal{B}_l} S_{l0} P_{\mathcal{B}_l}^*)^{-2} P_{\mathcal{B}_l}, \ldots, \beta P_{\mathcal{R}_u}^* (P_{\mathcal{R}_u} T_{u0} P_{\mathcal{R}_u}^*)^{-2} P_{\mathcal{R}_u}).$$

Thus $x_1 - x_0 \in N(\nabla^2\varphi(x_0))$, $s_{l1} - s_{l0} \in N(P_{\mathcal{B}_l})$, $\ldots$, $t_{u1} - t_{u0} \in N(P_{\mathcal{R}_u})$. The last four relations imply $x_1 - x_0 \in \mathcal{N}$ by virtue of primal feasibility (11–15). But $N(\nabla^2\varphi(x_0)) \cap \mathcal{N} = \{0\}$ by (A2), which proves uniqueness of the primal components (x_0, s_0, t_0). Now uniqueness of u_0, v_0 follows directly from β-complementarity, and uniqueness of z_0 follows from dual feasibility since $N(A^*) = \{0\}$ by (A1). □

Loosely speaking, if CP(β) has *no* solution, there must be some (one-sided) barrier term that drives the iterates $x^{(k)}$ away to infinity in a direction d where the curvature $d^* \nabla^2\varphi(x^{(k)}) d$ eventually tends to zero. The underlying CP must have an unbounded set of solutions or an infimum that is not attained (as in $\min_{x \geq 0} 0$ or $\min_{x \geq 0} e^{-x}$), and CP(β) becomes unbounded. This cannot happen when $\mathcal{S}$ is bounded: then φ becomes eventually strictly increasing in every feasible direction, and the unique solutions of CP(β) converge to a unique point in $\mathcal{S}$ for $\beta \downarrow 0$.

For more detailed information on the broad field of interior point methods see, e.g., [32] and references therein.

Observe finally that the reduced KKT system (16) is equivalent to a strongly convex *quadratic* program (QP) *without inequalities*,

$$\min_{\Delta x} \frac{1}{2}\Delta x^* \bar{H} \Delta x + \bar{f}^* \Delta x \quad \text{s.t.} \quad A\Delta x + \bar{a} = 0$$

where $\bar{a} := Ax + a$. The interior point method thus reduces the solution of MCSP to solving a sequence of simplified problems from the *same* class. This view restores a direct optimization context for the KKT system and provides—in the tree-sparse case—the control-theoretic interpretation on which our solution algorithm is based, cf. [24, 26]. In the following we are concerned with the treatment of that QP only.

4. Tree-sparse solution algorithm

In the case of interest, problem MCSP, the convex QP associated with the reduced KKT system (16) has precisely the same structure as the original problem. This is due to the separability of range constraints: contributions from all inequalities are absorbed into the local H_j (and f_j) *separately* for each node,

$$\bar{H}_j := \nabla^2 \varphi_j(x_j) + \Phi_j + F_j^{r*} \Psi_j F_j^r \geq 0. \tag{17}$$

Thus we arrive at the tree-sparse QP

$$\min_{\Delta x} \quad \sum_{j \in V} \frac{1}{2} \Delta x_j^* p_j \bar{H}_j \Delta x_j + p_j \bar{f}_j^* \Delta x_j \tag{18}$$

$$\text{s.t.} \quad G_j \Delta x_i + \bar{h}_j = P_j \Delta x_j \tag{19}$$

$$F_j^x \Delta x_j + \bar{e}_j^x = 0 \tag{20}$$

$$\sum_{j \in V} p_j F_j \Delta x_j + \bar{e}_V = 0. \tag{21}$$

The origin of this QP as a step direction is of no interest for the purpose of developing the KKT solution algorithm; therefore we simplify notation by dropping all Δ's and overbars from now on.

The Lagrangian inherits the separability properties of the (simplified) MCSP. With dual states λ_j and local and global equality multipliers μ_j^x, μ, respectively, it reads

$$L(x, \lambda, \mu^x, \mu) = \sum_{j \in V} p_j L_j(x_i, x_j, \lambda_j, \mu_j^x, \mu)$$

where nodes are coupled only by the dynamics and the global equality constraints,

$$\begin{aligned} L_j(x_i, x_j, \lambda_j, \mu_j^x, \mu) \quad = \quad & \frac{1}{2} x_j^* H_j x_j + f_j^* x_j - \lambda_j^* (G_j x_i + h_j - P_j x_j) - \\ & \mu_j^{x*} (F_j^x x_j + e_j^x) - \mu^* (F_j x_j + e_V). \end{aligned}$$

As pointed out in [26], the objective, global constraints, Lagrangian (and hence dual dynamic equations) involve (conditional) expectations. Here we are only interested in the *algebraic* structure, not in the stochastic interpretation. Henceforth, node probabilities p_j will therefore be absorbed into H_j, f_j, F_j and λ_j, μ_j^x; similarly they disappear into redefined dual slacks u, v.

When variables and right hand side are arranged as (x, λ, μ^x, μ) and (f, h, e^x, e_V), respectively, the equality constraint matrix A of the reduced KKT system (16) is obtained as in (8), and the Hessian is $H = \text{Diag}(H_0, \ldots, H_N)$.

In the previous sections we have written all linear systems in the form $Ax + a = 0$ which arises in linearizations of nonlinear equations. From now on we simply switch the sign of the right hand sides, and treat all linear equations in the form $Ax = a$. The tree-sparse reduced KKT system in node-wise representation then reads

$$H_j x_j + P_j^* \lambda_j - \sum\nolimits_{k \in S(j)} G_k^* \lambda_k - F_j^{x*} \mu_j^x - F_j^* \mu = f_j \quad \forall j \in V, \tag{22}$$

$$G_j x_i - P_j x_j = h_j \quad \forall j \in V, \tag{23}$$

$$F_j^x x_j = e_j^x \quad \forall j \in V, \tag{24}$$

$$\sum\nolimits_{j \in V} F_j x_j = e_V. \tag{25}$$

In the following sections we develop the complete solution algorithm, the *tree-sparse Schur complement method* (or SC method), taking both a global and a local view.

4.1. **Projection.** From a global viewpoint, the first step in our algorithm is a *projection* into the null space $N(F^x)$ of local constraints. It is accomplished by eliminating the state components x_1 that are determined by those constraints, then partitioning the remaining KKT system accordingly, and finally eliminating the local multipliers μ^x. The result is a projected KKT system in terms of the null space components x_2,

$$\left[\begin{array}{c|cc} H_{22} & G_2^* & F_2^* \\ \hline G_2 & & \\ F_2 & & \end{array}\right] \left[\begin{array}{c} x_2 \\ \hline -\lambda \\ -\mu \end{array}\right] = \left[\begin{array}{c} \bar{f}_2 \\ \hline \bar{h} \\ \bar{e}_V \end{array}\right]. \tag{26}$$

Locally this gives a separate projection in each node since $F^x = \operatorname{Diag}(F_j^x)$. The rank condition (A1.1) ensures that one can factor $F_j^x = L_j^x \begin{pmatrix} I & 0 \end{pmatrix} U_j$ where L_j^x, U_j are non-singular and L_j^x is lower triangular. (In dense standard factorizations U_j is typically upper triangular or orthogonal; cf., e.g., [11].) In the local partitioning of the KKT system one defines $\begin{pmatrix} P_{j1} & P_{j2} \end{pmatrix} := P_j U_j^{-1}$, similarly for F_j and G_k, $k \in S(j)$, and

$$\begin{pmatrix} x_{j1} \\ x_{j2} \end{pmatrix} := U_j x_j, \quad \begin{pmatrix} f_{j1} \\ f_{j2} \end{pmatrix} := U_j^{-*} f_j, \quad \begin{pmatrix} H_{j11} & H_{j21}^* \\ H_{j21} & H_{j22} \end{pmatrix} := U_j^{-*} H_j U_j^{-1}.$$

State components $x_{j1} = (L_j^x)^{-1} e_j^x$ are now obtained immediately, and local multipliers are formally eliminated after substituting x_{j1} into the remaining equations,

(27)
$$-\mu_j^x = (L_j^x)^{-*} \Big[-H_{j21}^* x_{j2} - P_{j1}^* \lambda_j + \sum\nolimits_{k \in S(j)} G_{k1}^* \lambda_k + F_{j1}^* \mu + (f_{j1} - H_{j11} x_{j1}) \Big].$$

U_j is required in the partitioning of the blocks G_k, $k \in S(j)$, which are all in the same column, see (9). Otherwise the block operations are independent for each node and do not create any fill-in outside the original QP blocks: The resulting projected system has precisely the tree-sparse structure of the original KKT system but without local constraints. In node-wise representation it has the form

$$H_{j22} x_{j2} + P_{j2}^* \lambda_j - \sum\nolimits_{k \in S(j)} G_{k2}^* \lambda_k - F_{j2}^* \mu = \bar{f}_{j2} \quad \forall j \in V, \tag{28}$$

$$G_{j2} x_{i2} - P_{j2} x_{j2} = \bar{h}_j \quad \forall j \in V, \tag{29}$$

$$\sum\nolimits_{j \in V} F_{j2} x_{j2} = \bar{e}_V. \tag{30}$$

The elementary node operations of the projection are listed in Table 1 (steps 1–7) and will be discussed in Sect. 4.3.

4.2. **Recursion.** The main part of our algorithm eliminates primal and dual states x_2, λ from (26) by Schur complement calculations. The obvious straightforward approach would calculate $x_2 = H_{22}^{-1}(G_2^*\lambda + F_2^*\mu + \bar{f}_2)$ first, replacing the zero lower right corner of the KKT matrix by the Schur complement

$$\begin{bmatrix} -\hat{Y} & \hat{Z}^* \\ \hat{Z} & -\hat{X}_V \end{bmatrix} := \begin{bmatrix} -Y & Z^* \\ Z & -X_V \end{bmatrix} - \begin{bmatrix} G_2 \\ F_2 \end{bmatrix} H_{22}^{-1} \begin{bmatrix} G_2^* & F_2^* \end{bmatrix}, \qquad \begin{bmatrix} -Y & Z^* \\ Z & -X_V \end{bmatrix} := 0.$$

Next, $\lambda = \hat{Y}^{-1}(\hat{Z}^*\mu + \hat{h})$ with $\hat{h} := \bar{h}_2 - G_2 H_{22}^{-1}\bar{f}_2$ would be obtained.

The key difference in our approach is a *sparse* elimination of x_2, λ in a recursion based on alternating *node-wise* analogs of the primal and dual Schur complement steps sketched above,

$$x_{j2} = H_{j22}^{-1}(-P_{j2}^*\lambda_j + F_{j2}^*\mu + \bar{f}_{j2}), \qquad -\lambda_j = \hat{Y}_j^{-1}(G_{j2}x_{i2} - \hat{Z}_j^*\mu - \hat{h}_j).$$

Here the simple trick consists in choosing a leaf $j \in L$ for the elimination, so that the sum over $S(j)$ in (28) is empty. The two steps effectively "cut off" the leaf, and the procedure is repeated for the resulting KKT system on the smaller tree.

More details and explanations of the recursion have been presented in [26, 28]; the elementary node operations are found in Table 1 (steps 8–13 and 14–19).

Fill-in occurs precisely in the blocks X_V, Y, Z, where $Y = \mathrm{Diag}(Y_0, \ldots, Y_N)$ and $Z = \begin{pmatrix} Z_0 & \ldots & Z_N \end{pmatrix}$. The blocks Y_j, Z_j are processed in step j, immediately after being filled, whereas X_V accumulates contributions from the entire tree. Finally, after cutting off the root, the KKT system $X_\emptyset \mu = e_\emptyset$ on the empty tree gives the global multiplier $\mu = X_\emptyset^{-1} e_\emptyset$.

Now we prove that the eliminations just described are all possible under regularity conditions (A1.1–A2*).

Lemma 4. *The matrices $H_{j22}, \hat{Y}_j, X_\emptyset$ above are all positive definite.*

Proof. (a) By definition, H_{j22} is the restriction $\bar{H}_j|N(F_j^x)$ with $\bar{H}_j$ from (17). As in Lemma 2 one has

$$N(\bar{H}_j) = N(H_j) \cap N(P_{\mathcal{B}_{lj} \cup \mathcal{B}_{uj}}) \cap N(P_{\mathcal{R}_{lj} \cup \mathcal{R}_{uj}} F_j^r).$$

Condition (A2*) therefore implies positive definiteness of

$$H_{j22}\Big| \bigcap_{k \in S(j)} N(G_{k2}). \tag{31}$$

If j is a leaf, we get $H_{j22} > 0$ directly. This in turn implies positive definiteness of $\hat{Y}_j = P_{j2} H_{j22}^{-1} P_{j2}^*$ since P_{j2} is the restriction $P_j|N(F_j^x)$, which has full rank by (A1.2). If j is not a leaf, then H_{j22} has already been modified by previous operations in the child nodes $k \in S(j)$ and must actually be replaced by

$$\tilde{H}_j = H_{j22} + \sum_{k \in S(j)} G_{k2}^* \hat{Y}_k^{-1} G_{k2},$$

cf. step (18) in Table 1. This is positive definite by (31) and the positive-definiteness of $\hat{Y}_k$ for $k \in S(j)$ (which has just been proved). The argument proceeds inductively.

(b) By construction, x_2, λ solve the KKT subsystem

$$\begin{bmatrix} H_{22} & G_2^* \\ G_2 & \end{bmatrix} \begin{bmatrix} x_2 \\ -\lambda \end{bmatrix} = \begin{bmatrix} \bar{f}_2 + F_2^*\mu \\ \bar{h} \end{bmatrix}$$

which is non-singular by (a). The third equation $F_2 x_2 = \bar{e}_V$ in (26) therefore gives

$$X_\emptyset = \begin{bmatrix} F_2 & 0 \end{bmatrix} \begin{bmatrix} U & V^* \\ V & W \end{bmatrix} \begin{bmatrix} F_2^* \\ 0 \end{bmatrix} = F_2 U F_2^* \quad \text{where} \quad \begin{bmatrix} U & V^* \\ V & W \end{bmatrix} := \begin{bmatrix} H_{22} & G_2^* \\ G_2 & \end{bmatrix}^{-1}.$$

We have $G_2U = 0$ and $UH_{22} + V^*G_2 = I$. The range of U satisfies $R(U) \subseteq N(G_2)$ by the first relation. Thanks to the second relation we actually have $R(U) = N(G_2)$. Otherwise $x_2 \in (N(U) \cap N(G_2)) \setminus \{0\}$ exists, and $\|x_2\|_2^2 = x_2^*(UH_{22} + V^*G_2)x_2 = 0$: a contradiction. Observe now that $F_2|N(G_2)$ has full rank by (A1.3). Thus $X_\emptyset$ is non-singular, and it remains to show that $X_\emptyset \geq 0$. From Table 1 we get the node-wise representation $X_\emptyset = \sum_{j\in V}(\hat{F}_j\hat{F}_j^* - \tilde{Z}_j\tilde{Z}_j^*)$. The node term expands to

$$\hat{F}_j\hat{F}_j^* - \hat{F}_j\hat{P}_j^*\hat{Y}_j^{-1}\hat{P}_j\hat{F}_j^* =: \hat{F}_j(I - Q)\hat{F}_j^*$$

where $Q = Q^* = Q^2$ is an orthogonal projection. Thus $Q \leq I$, implying $X_\emptyset \geq 0$. $\square$

The lemma shows that the recursive elimination works under weaker conditions than the global Schur complement approach: obviously the latter requires $H_{22} > 0$, i.e., $H_{j22} > 0$ in *all* nodes. Another drawback of the global version is the greater amount of fill-in: $\hat{Y} = G_2H_{22}^{-1}G_2^*$ is clearly *not* block-diagonal (as its counterpart in the recursion); it contains many off-diagonal blocks due to the tree structure in G_2.

It is easily seen that the recursion extends to the case where the lower right block $\begin{pmatrix} -Y & Z^* \\ Z & -X_V \end{pmatrix}$ is initially negative semidefinite rather than zero. If that block is actually negative *definite* and H_{j22} is positive definite, then the projected system becomes *quasi-definite* [30], and the recursive part of our algorithm can be seen as a special case of the general approach in [31] which has been adapted to multistage stochastic programs in [2] by a special pre-ordering technique called *tree dissection*. Thus the two algorithms become very similar. Differences are due to the finer adaptation of our tree-sparse algorithm to the rich structure of MCSP: we require less restrictive regularity conditions, we perform block level operations, the order of pivot blocks is fixed a priori, and we use off-diagonal pivots if appropriate (i.e., in the projection).

4.3. **Summary and overview.** Table 1 gives a complete overview of individual node operations of the tree-sparse SC method. Three phases of the algorithm are distinguished: a symmetric *factorization* of the KKT matrix, $\Omega = \Lambda\Pi\Lambda^*$, the associated *transformation* of the right hand side, $\bar{\alpha} = \Lambda^{-1}\alpha$, and the calculation of solution variables in the *substitution* phase, $\zeta = \Lambda^{-*}\bar{\zeta}$ where $\bar{\zeta} = \Pi^{-1}\bar{\alpha}$. Elementary steps in each phase are grouped to give the *projection* (1–7) and *recursion* parts (8–20). The latter has subgroups corresponding to the elimination of *primal* states x_{j2} (8–13), *dual* states λ_j (14–19), and global multipliers μ (20). Operations in the factorization and transformation phases proceed in the given order (and can be combined in a single traversal of the tree); the substitution phase proceeds in reverse order (from bottom to top). Correspondingly, the recursion (8–20) proceeds inward in the factorization and transformation phases and outward in the substitution, whereas the projection proceeds recursively outward in the factorization and transformation but inward in the substitution phase. The projection step thus establishes independent pre- and post-processing operations of the basic recursion.

The vertical coupling on the tree is seen in steps (5,7,18) which operate on the data of parent i and child j; all other steps are strictly local to node j (but may access global data X_V, e_V, μ, of course). The coupling steps (5,7,18) are omitted in the root (which has no parent) whereas the final step (20) is performed only in the root.

The symmetry of the factorization is apparent in corresponding operations of the three phases. (Dual variables always appear with their minus sign to enhance this.) Slight asymmetries result from merging the multiplication by Π^{-1} into the transformation and substitution phases.

Indices $V, V', \emptyset$ on X, e indicate the vertex set of the current tree, where it is assumed that j is the first leaf to be cut off, and $V' := V \setminus \{j\}$. (Of course, V, V'

TABLE 1. Node operations of the tree-sparse SC method. For explanations see Sect. 4.3.

Step	Factorization	Transformation	Substitution
(1)	$F_j^x = L_j^x \begin{pmatrix} I & 0 \end{pmatrix} U_j$	$x_{j1} = (L_j^x)^{-1} e_j^x$	$-\mu_j^x = (L_j^x)^{-1} \bar{\mu}_j^x$
(2)	$\begin{pmatrix} H_{j11} & H_{j21}^* \\ H_{j21} & H_{j22} \end{pmatrix} = U_j^{-*} H_j U_j^{-1}$	$\begin{pmatrix} f_{j1} \\ f_{j2} \end{pmatrix} = U_j^{-*} f_j$	$x_j = U_j^{-1} \begin{pmatrix} x_{j1} \\ x_{j2} \end{pmatrix}$
(3)		$\bar{f}_{j1} = f_{j1} - H_{j11} x_{j1}$	
(4)		$\bar{f}_{j2} = f_{j2} - H_{j21} x_{j1}$	$\bar{\mu}_j^x = -H_{j21}^* x_{j2} + \tilde{\mu}_j^x$
(5)*	$\begin{pmatrix} G_{j1} & G_{j2} \end{pmatrix} = G_j U_i^{-1}$	$\bar{h}_j = h_j - G_{j1} x_{i1}$	$\tilde{f}_{i1} = -G_{j1}^*(-\lambda_j) + \bar{f}_{i1}$
(6)	$\begin{pmatrix} P_{j1} & P_{j2} \end{pmatrix} = P_j U_j^{-1}$	$+ P_{j1} x_{j1}$	$\tilde{\mu}_j^x = +P_{j1}^*(-\lambda_j)$
(7)	$\begin{pmatrix} F_{j1} & F_{j2} \end{pmatrix} = F_j U_j^{-1}$	$\bar{e}_V = \bar{e}_{V'} - F_{j1} x_{j1}$	$-F_{j1}^*(-\mu) + \tilde{f}_{j1}$
(8)	$H_{j22} = L_j L_j^*$		
(9)	$\hat{P}_j = P_{j2} L_j^{-*}$		
(10)	$\hat{F}_j = F_{j2} L_j^{-*}$	$\hat{f}_j = L_j^{-1} \bar{f}_{j2}$	$x_{j2} = L_j^{-*}[$
(11)	$\hat{Z}_j = Z_j + \hat{F}_j \hat{P}_j^*$		
(12)	$\hat{Y}_j = Y_j + \hat{P}_j \hat{P}_j^*$	$\hat{h}_j = \bar{h}_j + \hat{P}_j \hat{f}_j$	$+\hat{P}_j^*(-\lambda_j)$
(13)	$\hat{X}_{V'} = X_V + \hat{F}_j \hat{F}_j^*$	$\hat{e}_{V'} = \bar{e}_V - \hat{F}_j \hat{f}_j$	$-\hat{F}_j^*(-\mu) + \hat{f}_j]$
(14)	$\hat{Y}_j = \hat{L}_j \hat{L}_j^*$		
(15)	$\tilde{Z}_j = \hat{Z}_j \hat{L}_j^{-*}$		
(16)	$\tilde{G}_j = \hat{L}_j^{-1} G_{j2}$	$\tilde{h}_j = \hat{L}_j^{-1} \hat{h}_j$	$-\lambda_j = \hat{L}_j^{-*}[$
(17)*	$\tilde{F}_i = F_{i2} + \tilde{Z}_j \tilde{G}_j$		
(18)*	$\tilde{H}_i = H_{i22} + \tilde{G}_j^* \tilde{G}_j$	$\tilde{f}_i = \bar{f}_{i2} + \tilde{G}_j^* \tilde{h}_j$	$\tilde{G}_j x_{i2} +$
(19)	$X_{V'} = \hat{X}_{V'} - \tilde{Z}_j \tilde{Z}_j^*$	$e_{V'} = \hat{e}_{V'} + \tilde{Z}_j \tilde{h}_j$	$\tilde{Z}_j(-\mu) - \tilde{h}_j]$
(20)	$X_\emptyset = L_\emptyset L_\emptyset^*$	$\hat{e}_\emptyset = L_\emptyset^{-1} e_\emptyset$	$-\mu = L_\emptyset^{-*}(-\hat{e}_\emptyset)$

change when further nodes are processed.) Similarly, the notation in steps (8) and (10) assumes that j is a leaf. Otherwise $H_{j22}, F_{j2}, \bar{f}_{j2}$ have already been modified by child nodes $k \in S(j)$ in steps (17,18), and must be replaced by $\tilde{H}_j, \tilde{F}_j, \tilde{f}_j$, respectively. Conversely, in step (7) we define $\bar{e}_\emptyset := e_V$, and $\tilde{f}_{j1} := \bar{f}_{j1}$ to make the notation consistent with step (5). (Here $j \in L$ is the *last* node to be projected.)

Let us finally give some remarks on the complexity of the algorithm, i.e., of the factorization. Since $l_j, l_j^x \le n_j$, one finds that the number of operations in node j is $O(n_j \max(n_j, l_j^r, m)^2)$. If we set $\hat{l} := \max_j \max(n_j, l_j^r, m)$, this gives the total number $O(\hat{l}^3 |V|)$. That is, the KKT system solution has *linear* complexity in the tree size and cubic complexity in the largest local dimension—provided that $\hat{l}$ is independent of $|V|$. This shows that an appropriate treatment of each category of constraints is essential for efficiency: although the recursion in its basic form (i.e., without local projections, [26, 28]) can principally handle the local constraints by subsuming them under the global ones, this may lead to overall cubic complexity $O(|V|^3)$.

5. LOCAL SPARSITY

Apart from some nonlinear operations and lots of vector-vector operations, the interior point method performs matrix-vector *multiplication* with the KKT matrix Ω (or its principal components A, B, H) and KKT system *solutions*, the latter being implemented by the tree-sparse factorization. We discuss how to take advantage of sparse blocks in the *factorization* phase, including reduction of the large barrier KKT matrix. Exploiting sparsity in the transformation and substitution phases is straightforward (once the factors are available); the same is of course true for matrix-vector multiplication.

5.1. **General discussion.**

Reduction. The only operation involving a non-diagonal block is the augmentation of H_j by the symmetric product $F_j^{r*}\Psi_j F_j^r$. Exploitation of sparsity is straightforward, with one exception: In some cases it may be preferable to defer this operation until after the projection step. That is, F_j^r is first partitioned as $\begin{pmatrix} F_{j1}^r & F_{j2}^r \end{pmatrix} = F_j^r U_j^{-1}$, and then $F_{j2}^{r*}\Psi_j F_{j2}^r$ is added to H_{j22}. In that case the sparsity of F_{j2}^r should of course be taken into account in the projection step.

Projection. The local projections are the most complicated steps in the entire tree-sparse algorithm. They affect the fill-in in all local blocks and (during the recursion) possibly in other nodes toward the root. They also influence numerical stability. Choosing the factorization of F_j^x is therefore the most critical step in local structure exploitation. (It is also the step that offers the greatest freedom of choice.)

We do not claim to have a recipe that yields the "best" factorization in each situation. Note, however, that the clear structure of the tree-sparse algorithm—particularly the lack of choices in other operations and the confinement of fill-in to the given blocks—provides strong guidelines for the sparsity analysis. (The principal goal is basically a simple structure of the inverse transformation U_j^{-1}.) Below we give an example where a very good choice is easily made by close inspection; the general case will be a subject of future research.

Recursion. Structure exploitation in that (central) part of the algorithm is again mostly straightforward. Basically one just has to monitor where fill-in occurs. Although sparse Cholesky factorizations of H_{j22} and $\hat{Y}_j$ may sometimes be possible, these blocks tend to be dense in general: before being factored, they accumulate information of the entire subtree rooted in j. Similar statements apply of course to the final factorization of $X_\emptyset$.

5.2. **An application example.**

In [1], Ainassaari, Kallio, and Ranne develop an asset management model for long-term planning in Finnish pension insurance companies. The primary goal of the model is to maintain the working capital above a certain minimum level. Otherwise the company will be set under the special supervision of the Ministry of Social Affairs and Health, and eventually it will lose its licence unless solvency is recovered. The model includes further constraints representing other legal restrictions etc., and a convex objective that maximizes solvency. Results are reported for a scenario tree with 1024 scenarios and $T = 5$ periods covering $2+3+5+10+10 = 30$ years. In [14], Kallio and Salo employ an iterative parallel algorithm to solve the same problem with up to 192 k scenarios.

This portfolio management problem fits precisely in the framework of our general class: the objective is (nonlinear) convex and all categories of constraints are present except for global equalities. The variables in each node are $x_j \in \mathbf{R}^{10}$, with 2 (implicit) dynamic equations, 3 local constraints, and 4 range constraints. This gives block dimensions $H_j \in \mathbf{R}^{10\times 10}$, $G_j, P_j \in \mathbf{R}^{2\times 10}$, $F_j^x \in \mathbf{R}^{3\times 10}$, $F_j^r \in \mathbf{R}^{4\times 10}$, yielding $55 + 20 + 20 + 30 + 40 = 165$ entries per node (where only the lower triangle of H_j is stored). Since X_V and Z are void due to the absence of global constraints, the entire fill-in during KKT solution consists of three Y_j entries per node. Operation counts for the KKT matrix factorization with completely dense blocks are displayed in Table 2. The augmentation of H_j by $F_j^{r*}\Psi_j F_j^r$ is included as step (0).

Let us now consider the actual sparse substructure in this particular problem. The respective symbols '$\cdot$', '$+$', '$-$', and '$*$' refer to zero, $+1$, -1, and generic matrix entries:

entry	zero	$+1$	-1	generic
symbol	$\cdot$	$+$	$-$	$*$

The Hessian blocks H_j are all dense. The internal structure of the remaining blocks (after simple a priori permutations of rows and columns) looks as follows:

$$G_j = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & * & \cdot & * \\ * & \cdot & * & * & * & * & * & * & * & * \end{bmatrix}, \qquad F_j^x = \begin{bmatrix} - & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & * & \cdot & \cdot \\ \cdot & - & \cdot & \cdot & \cdot & \cdot & \cdot & * & \cdot & \cdot \\ + & + & + & + & + & + & + & \ominus & - & \cdot \end{bmatrix},$$

$$P_j = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \oplus & \odot & \ominus \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \oplus & \oplus \end{bmatrix}, \qquad F_j^r = \begin{bmatrix} \cdot & \cdot & \cdot & + & \cdot & \cdot & \cdot & * & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & + & \cdot & \cdot & * & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & + & \cdot & * & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & - & * & \cdot & \cdot \end{bmatrix}.$$

The six encircled entries indicate positions where fill-in will occur.

Reduction. From the structure of F_j^r it is apparent that the symmetric product $F_j^{r*}\Psi_j F_j^r$ vanishes outside rows and columns $\mathcal{R} = \{4, \ldots, 8\}$; the nonzero block is

$$P_{\mathcal{R}}(F_j^{r*}\Psi_j F_j^r)P_{\mathcal{R}}^* = \begin{bmatrix} * & \cdot & \cdot & \cdot & * \\ \cdot & * & \cdot & \cdot & * \\ \cdot & \cdot & * & \cdot & * \\ \cdot & \cdot & \cdot & * & * \\ * & * & * & * & * \end{bmatrix}.$$

Obviously it modifies only nine entries in the lower triangle of H_j.

Projection. The factorization of F_j^x is almost trivial. After partitioning $F_j^x = \big(F_{j1}^x \;\; F_{j2}^x\big) \in \mathbf{R}^{3\times(3+7)}$ we observe that F_{j1}^x is lower triangular and self-inverse. Hence we let $L_j^x := F_{j1}^x \equiv (F_{j1}^x)^{-1}$ and $W_j := F_{j1}^x F_{j2}^x$ to obtain $F_j^x = L_j^x\big(I \;\; W_j\big)$. This gives

$$U_j = \begin{bmatrix} I & W_j \\ & I \end{bmatrix}, \quad U_j^{-1} = \begin{bmatrix} I & -W_j \\ & I \end{bmatrix}, \qquad W_j = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & * & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & * & \cdot & \cdot \\ + & + & + & + & \circledast & - & \cdot \end{bmatrix}.$$

W_j differs from F_{j2}^x only in the encircled entry (the fill-in) and in the two entries above (whose sign has switched). A detailed analysis reveals that the subsequent operations preserve sparseness very well: the local projection into $N(F_j^x)$ does not affect P_j at all, it does not create any further fill-in, and it modifies 46 elements of H_j and only six elements of each G_k, $k \in S(j)$. Moreover, the values of all pivots are ± 1 and the projection is stable. Table 2 shows that only the symmetric transformation of H_j in step (2) requires a significant amount of work. Otherwise the projection, like the previous reduction, comes almost for free.

Recursion. The blocks $H_{j22} \in \mathbf{R}^{7\times 7}$ and $\hat{Y}_j \in \mathbf{R}^{2\times 2}$ are completely dense; otherwise sparsity is mostly preserved and leads to simplifications in the block operations. Fill-in occurs only in five elements of P_j when forming $\hat{P}_j$ in step (9).

Table 2 shows that the savings here are less dramatic than in previous steps. However, the recursion is comparatively cheap anyway, taking about 25% of the total work in the dense case but more than 50% now. Total savings of algebraic operations are quite significant with roughly 75% over the dense case.

Memory. It suffices to store in each node the 55 relevant entries of H_j, the 17 generic entries marked by '$*$' in the remaining original blocks, and the 9 elements occurring as fill-in: three in $\hat{Y}_j$, one in F_j^x, and five in P_j. Thus a sparse implementation proves highly memory-efficient: it requires only 81 matrix elements per node versus 168 in the dense case—a reduction of more than 50%.

TABLE 2. Operation counts of factorization phase in example problem.

	Dense blocks		Sparse blocks		Savings due to sparse blocks			
Step	Add	Mult	Add	Mult	Add	Mult	Add%	Mult%
(0)	220	260	12	8	208	252	94.5	96.9
(1)	47	95	2	0	45	95	95.7	100.0
(2)	272	309	115	30	157	279	57.7	90.3
(5)	48	54	7	2	41	52	88.5	96.3
(6)	48	54	0	0	48	54	100.0	100.0
(8)	56	84	56	84	0	0	0.0	0.0
(9)	42	56	3	9	39	47	92.9	83.9
(12)	21	21	7	7	14	14	66.7	66.7
(14)	4	10	4	10	0	0	0.0	0.0
(15)	21	7	11	2	10	5	47.6	71.4
(18)	56	56	31	31	25	25	44.6	44.6
Sum	835	1006	248	183	587	823	70.3	81.8

5.3. **Summary.** We have seen in the example problem that the generic tree-sparse approach can be specialized to take full advantage of the particular local sparsity pattern. This requires an analysis of the node operations and results in specialized versions with an appropriate memory layout for the local matrix blocks. In accordance with the general philosophy of our approach it is important that the specialization is carried out *a priori* so that there is no significant overhead (especially memory management) once the iteration is started.

The simple structure of the example problem made the analysis quite easy. But is this typical, or were we just lucky? Of course, the example is *not* typical in general. We argue, however, that the experienced potential for sparsity exploitation *is* typical in financial applications like portfolio optimization or asset and liability management, for the following reasons:

Decoupling. Gains and losses in one position (stock, bond, liability, ...) do not depend on the amount of capital in other positions.

Unit entries. Constraints often describe capital transfers or summations of individual positions; hence the coefficient values are ± 1.

Stability. The dynamic equations involve return coefficients (one plus interest rate), which are in the order of unity.

Thus computational finance is among the primary application fields for further developments of local sparse-matrix techniques within our framework. Ideally, one would like to have a tool that performs the sparsity analysis, designs the memory layout, and generates code for the node operations. In the financial context this seems not too far-fetched as an ultimate goal. Research in that direction will probably stimulate similar developments in other application fields, and will also provide valuable experience and hints for investigations of the general case.

6. Conclusions

We have investigated a general class of multistage convex stochastic programs and the associated tree-sparse solution algorithm for the KKT systems that arise in a standard interior point framework. We have also explored the potential of exploiting problem-specific local sparse structures within this algorithmic framework. The experience gained so for leads us to believe that our approach is indeed very well suited to take advantage of the rich structure in multistage stochastic programs.

Some promising areas of future research have been pointed out in the section on local sparsity. Related investigations might be directed toward important specializations of the general problem class. A typical substructure in the dynamics arises, e.g., when future states depend not only on the presence but also on the past (cf. [26]).

All these issues are of similar interest for the problem variants with explicit dynamics mentioned in the introduction. The distinction of state and control variables lends them a finer structure, often with a higher potential for exploitation of sparsity. On the other hand, it makes the analysis more difficult. In a forthcoming paper we will analyze a real-life problem in this form and present computational results.

7. Acknowledgments

To a large extent, this work has been influenced and inspired by a close cooperation with the Operations Research group of Karl Frauendorfer at the University of St. Gallen. I am also indebted to Markku Kallio who provided me with the details of his asset management model.

References

[1] K. Ainassaari, M. Kallio, and A. Ranne, *An asset management model for a pension insurance company*, technical report, Helsinki School of Economics, 1997.
[2] 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, 1995.
[3] A. J. Berger, J. M. Mulvey, and A. Ruszczyński, *An extension of the DQA algorithm to convex stochastic programs*, SIAM J. Optim., 4 (1994), pp. 735–753.
[4] J. R. Birge, *Decomposition and partitioning methods for multistage stochastic linear programs*, Oper. Res., 33 (1985), pp. 989–1007.
[5] ———, *Stochastic programming computations and applications*, INFORMS J. Comput., 9 (1997), pp. 111–133.
[6] 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, (1987), pp. 1–19.
[7] 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.
[8] 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. Comput., 7 (1995), pp. 474–490.
[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. Comput., 7 (1995), pp. 386–401.
[10] H. I. Gassmann, *MSLiP: A computer code for the multistage stochastic linear programming problem*, Math. Programming, 47 (1990), pp. 407–423.
[11] G. H. Golub and C. F. V. Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, MD, USA, 2nd ed., 1989.
[12] A. Jain, *Unified formulation of dynamics for serial rigid multibody systems*, AIAA J. Guidance, 14 (1991), pp. 531–542.
[13] E. R. Jessup, D. Yang, and S. A. Zenios, *Parallel factorization of structured matrices arising in stochastic programming*, SIAM J. Optim., 4 (1994), pp. 833–846.
[14] M. Kallio and S. Salo, *Parallel computing tests on large-scale convex optimization*, technical report, Helsinki School of Economics, Aug. 1997.
[15] D. P. Morton, *An enhanced decomposition algorithm for multistage stochastic hydroelectric scheduling*, Ann. Oper. Res., 64 (1996), pp. 211–235.
[16] J. M. Mulvey and A. Ruszczyński, *A new scenario decomposition method for large-scale stochastic optimization*, Oper. Res., 43 (1995), pp. 477–490.
[17] S. S. Nielsen and S. A. Zenios, *Scalable parallel Benders decomposition for stochastic linear programming.*, Parallel Comput., 23 (1997), pp. 1069–1088.

[18] R. T. Rockafellar, *Duality and optimality in multistage stochastic programming*, Ann. Oper. Res., 85 (1999), pp. 1–19.

[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 Optimization, 28 (1990), pp. 810–822.

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

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

[22] ———, *Decomposition methods in stochastic programming*, Math. Programming, 79 (1997), pp. 333–353.

[23] E. Schweitzer, *An interior random vector algorithm for multistage stochastic linear programs*, SIAM J. Optim., 8 (1998), pp. 956–972.

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

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

[26] ———, *Recursive direct optimization and successive refinement in multistage stochastic programs*, Preprint SC-98-27, ZIB, 1998. To appear in Ann. Oper. Res.

[27] ———, *Markowitz revisited: single-period and multi-period mean-variance models*, Preprint SC-99-30, ZIB, 1999.

[28] ———, *Recursive direct algorithms for multistage stochastic programs in financial engineering*, in Operations Research Proceedings 1998, P. Kall and H.-J. Lüthi, eds., Springer-Verlag, 1999, pp. 241–250.

[29] 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.

[30] R. J. Vanderbei, *Symmetric quasidefinite matrices*, SIAM J. Optim., 5 (1995), pp. 100–113.

[31] R. J. Vanderbei and T. J. Carpenter, *Symmetric indefinite systems for interior point methods*, Math. Programming, 58 (1993), pp. 1–32.

[32] Y. Ye, *Interior point algorithms. Theory and analysis*, John Wiley, New York, 1997.

Marc C. Steinbach, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, 14195 Berlin-Dahlem, Germany

E-mail address: `steinbach@zib.de`

URL: `http://www.zib.de/steinbach`