A barrier Lagrangian dual method for multi-stage stochastic convex semidefinite optimization

Asma Gafour[1,2] · **Baha Alzalg**[1,3] *

Abstract In this paper, we present a polynomial-time barrier algorithm for solving multi-stage stochastic convex semidefinite optimization based on the Lagrangian dual method which relaxes the nonanticipativity constraints. We show that the barrier Lagrangian dual functions for our setting form self-concordant families with respect to barrier parameters. We also use the barrier function method to improve the smoothness of the dual objective function so that the Newton search directions can be exploited for usage. We finally implement the proposed algorithm on different test problems to show its effectiveness and efficiency.

Keywords Semidefinite programming · Multi-stage stochastic convex programming · Lagrangian dual · Interior point methods · Polynomial-time complexity

Mathematics Subject Classification 90C15 · 90C22 · 90C25 · 90C51 · 68Q25

1 Introduction

We study the multi-stage stochastic convex semidefinite programming (SCSDP) problem:

$$\min \ \mathbb{E}^{\zeta} f_0\left(x^{\zeta};\zeta\right) \tag{1}$$

$$\text{s.t.} \ \ x^{\zeta} \in \mathsf{S}_+^k(\zeta), \quad \forall \zeta \in S, \tag{2}$$

$$x_t^{\zeta} = x_t^{\xi}, \ \ \text{if } \zeta^t = \xi^t, \quad \forall\, \zeta,\ \xi \in S, \ \ t = 1,2,\ldots,T-1, \tag{3}$$

where

$$\mathsf{S}_+^k(\zeta) := \left\{\text{vec}\left(X^{\zeta}\right) : X^{\zeta} \in \mathbb{R}^{k\times k} \text{ is symmetric positive semidefinite}\right\} \subset \mathbb{R}^{k^2}$$

is the cone of vectors obtained from the vectorization of symmetric positive semidefinite matrices, the subvector ζ_1 is a deterministic vector which means that there is no uncertainty at the present time (stage 1), and the vector $\zeta := (\zeta_1, \zeta_2, \ldots, \zeta_T)$ is the random vector which comprises subvectors ζ_t that are observed at stages $t = 1, 2, \ldots, T$. The constraints in (3) are referred to as nonanticipativity constraints which mean that decisions made at the t-th stage, for each $t = 1, 2, \ldots, T-1$, can only be based on observations made up to that point.

Barrier methods are one of the most attractive methods for convex programming [1–6]. We in particular introduce a barrier Lagrangian dual algorithm for solving T-stage SCSDP (1)-(3) by utilizing the work of Zhao [7]. Some authors [8, 9] studied the two-stage version of the SCSDP problem with a linear objective function. The multi-stage SCSDP not only has more scenarios by definition, but also more complicated scenarios of tree structures. Furthermore, the usage of convex objective can provide a more general problem

✉ *B. Alzalg (corresponding author)
b.alzalg@ju.edu.jo

A. Gafour
asm9170457@ju.edu.jo

1 Department of Mathematics, The University of Jordan, Amman, Jordan 11942
2 Department of Mathematics, University of Djilali Liabes, Sidi Bel Abbes, Algeria 22038
3 Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210
Friday 7th January, 2022 (17:39)

setting that is applicable to a wider variety of applications. Therefore, solving the multi-stage SCSDP is significantly more difficult than solving the two-stage semidefinite linear programming. For applications of multi-stage stochastic programs, see for example [10–12]. For applications of stochastic semidefinite programming, see for example [13–15].

There are three basic formulations to solve stochastic conic programming, namely: Deterministic equivalence, Benders decomposition, and Lagrangian dual. The methods proposed in [16–18] are based on the deterministic equivalence formulation, those proposed in [8, 9, 12, 19–25] are based on the Benders decomposition formulation, and those proposed in [7, 26–29] and the one we propose in this paper are based on the Lagrangian dual formulation. The Lagrangian dual formulation associates a small program with each scenario, connects these small programs using nonanticipativity constraints, and then relaxes the nonanticipativity constraints. The Lagrangian dual formulation is appropriate for multi-stage stochastic nonlinear programming because the nonlinearity only appears in small programs and the linearity of the nonanticipativity constraints leads to a simple formula for subgradients of the master objective function in the Lagrangian dual.

Nonetheless, there are two difficulties are included in the Lagrangian dual formulation: First, the master objective function is neither strictly convex nor smooth. Second, the master problem has a massive number of variables. Because of these structural limitations, the master problem is mostly solved by nonsmooth methods that only use subgradients. The rate of convergence is slow in these methods. The method proposed in this paper uses self-concordant barriers to improve the smoothness and convexity of the Lagrangian dual, allowing us to use higher-order derivatives. The previously mentioned second difficulty is obvious in the computation of Newton directions. We will propose an approach for computing Newton directions that make use of special structures of nonanticipativity constraints.

In our problem, the low-cost solutions can be found because the subproblem is unconstrained and strictly convex. We will see that we can solve the problem in polynomial time because the master problem's objective function is a self-concordant family. The subproblem can also be solved efficiently in parallel because it is separable into scenarios. Let s be the number of all scenarios and ϵ be the desired accuracy of the final solution, we will also see that we need at most $O(s^{1/2}k\ \ln(r^{(0)}/\epsilon))$ Newton iterations to terminate the short-step algorithm from an initial value of the barrier parameter r^0 to the stopping value ϵ, and we need at most $O(sk^2\ \ln(r^{(0)}/\epsilon))$ Newton iterations in the long-step algorithm for this terminating.

We use the following notations throughout the paper. The set $\mathbf{S}^k$ is used to describe the set of k-th order real symmetric matrices, and the set $\mathbf{S}^k_{++}(\zeta)$ is used to describe the cone generated by positive definite matrices in the cone $\mathbf{S}^k_+(\zeta)$. For $A, B \in \mathbb{S}^n$, we write $A \succeq 0$ ($A \succ 0$) to mean that A is positive semidefinite (positive definite), and we use $A \succeq B$ or $B \preceq A$ to mean that $A - B \succeq 0$. Associated with each matrix $X \in \mathbb{R}^{k\times k}$, we use $x := \mathrm{vec}(X)$ to denote the k^2-th dimensional vector whose $((j-1)k+i)$-th entry is the (i, j)-th entry of X. Associated with each vector $x \in \mathbb{R}^{k^2}$, we use $X := \mathrm{Mat}(x)$ to denote the $k \times k$ matrix whose (i, j)-th entry is the $((j-1)k+i)$-th entry of x.

We use I and O to denote the identity and zero matrices, respectively, of appropriate dimensions. The product $A \otimes B$ represents the Kronecker product of matrices A and B with arbitrary sizes. This product satisfies the relationship

$$(A \otimes B)(C \otimes D) = AC \otimes BD$$

assuming that the number of rows in A and B equals the number of columns in C and D. We also have

$$(A \otimes B)\mathrm{vec}(C) = \mathrm{vec}\left(BCA^{\mathsf{T}}\right).$$

We use "∇" to denote the Jacobian of a vector function. Let r be the barrier parameter, λ represent the vector of Lagrange multipliers, and x represent the vectorization of decision matrix variables. For any function g, "$'$" represents the partial derivative of $g(r, \lambda, x)$ with respect to r, and ∇_λ and ∇_x represent the partial derivatives with respect to λ and x, respectively. As an example,

$$g'(r,\lambda,x) = \frac{\partial}{\partial r}g(r,\lambda,x),\ \ \nabla_x g(r,\lambda,x) = \frac{\partial}{\partial x}g(r,\lambda,x),\ \ \nabla_x g''(r,\lambda,x) = \frac{\partial^3}{\partial^2 r \partial x}g(r,u,x).$$

If $x = x(r, \lambda)$ is a vector function of r and λ, and $h(r) = g(r, x(r, \lambda))$, then

$$h'(r) = \frac{\partial}{\partial r}g(r,x(r,\lambda)) = g'(r,x(r,\lambda)) + \nabla_x g(r,x(r,\lambda))^{\mathsf{T}} x'(r,\lambda).$$

We use $\nabla^k_{x,\dots,x} g(r,\lambda,x)[d_1,d_2,\dots,d_k]$ to denote the value of the k-th partial derivative of g taken at x along the collection of directions $d_1,d_2,\dots,d_k$.

The next pages of the paper are structured as follows: In Section 2, we write the Lagrangian dual of the multi-stage SCSDP, incorporate a barrier function into it, and compute some of their derivatives. Section 3 demonstrates that the master objective functions $\eta(r,\cdot)$ are a self-concordant family. Section 4 describes an efficient method for computing the Newton directions for the master problem. Section 5 presents the proposed algorithm. In Section 6, we prove the convergence of the algorithm and estimate the number of iterations required by both short- and long-step algorithms. In Section 7, we provide numerical results. Finally, Section 8 contains some concluding remarks and throws some open questions for possible future work.

2 Barrier Lagrangian dual

A scenario is a sequence of realizations $\zeta = (\zeta_1,\zeta_2,\dots,\zeta_T)$ if the event T will occur with a positive probability for each $t \in \{1,2,\dots,T\}$, the event ζ_t will happen with a positive probability, given realizations $\zeta_1,\zeta_2,\dots,\zeta_{t-1}$. The vector $\zeta^t := (\zeta_1,\zeta_2,\dots,\zeta_t)$ is referred to as a subscenario. We do not distinguish between a random variable, a realization, or a scenario because they can all be recognized from the context. Let the set of all scenarios be denoted by S, and its cardinality be denoted by $s := |S|$. We take s to be finite throughout the paper. For each scenario $\zeta \in S$ and each stage $t \in \{1,2,\dots,T\}$, the decision variable x^ζ_t is a k^2_t-th dimensional vector, where $k^2 = \sum_{t=1}^T k^2_t$.

To put the exposition more simply, let denote

$$x := \left(x^\zeta : \zeta \in S\right) \in \mathbb{R}^{sk^2},\quad f(x) := \mathbb{E}^\zeta f_0\left(x^\zeta;\zeta\right),\quad \mathsf{S}_+ := \prod_{\zeta\in S} \mathsf{S}^k_+(\zeta) \subset \mathbb{R}^{sk^2},\quad \text{and}\quad \mathsf{S}_{++} := \prod_{\zeta\in S} \mathsf{S}^k_{++}(\zeta) \subset \mathbb{R}^{sk^2}.$$

Let $Ax = b$ be the nonanticipativity constraints in (3) (it is clear that $b = 0$ in the constraint), where A is a $n \times sk^2$ matrix and $b \in \mathbb{R}^n$.

Using the above notations, the multi-stage SCSDP can be compactly written as

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax = b, \\ & x \in \mathsf{S}_+. \end{array} \tag{4}$$

We put forward the following two assumptions.

Assumption 2.1 *The matrix A has full row rank and $n < sk^2$.*

Assumption 2.2 $\mathsf{S}_{++} \neq \emptyset$.

In the nonanticipativity constraint, the matrix A is built by users who can always choose independent rows to form a full-row-rank matrix A. In Section 4, we will give some specific examples of the matrix A in detail. We can discuss several methods for expressing the nonanticipativity constraint and constructing a full-row-rank matrix A. However, for our subsequent analysis, we only need the requirement on Assumption 2.1 to ensure invertibility of some operators. Assumption 2.2 is a standard assumption in convex programming to ensure the strong duality.

Observe that $f : \mathbb{R}^{sk^2} \to \mathbb{R}$ and S_+ are separable into scenarios, whereas the nonanticipativity constraint $Ax = b$ connects x^ζ for all scenarios. Here, the random vector $\zeta = (\zeta^1,\zeta^2,\dots,\zeta^T)$ is made up of subvectors ζ^t that are observed at stages $t = 1,2,\dots,T$. As a result, we attempt to relax the constraint $Ax = b$ by employing the Lagrangian dual of problem (4) as follows:

$$\min_{\lambda\in\mathbb{R}^n} \eta(\lambda) \tag{5}$$

where the subproblem η can be written as

$$\eta(\lambda) := \max\left\{-f(x) + \lambda^\mathsf{T}(Ax - b) : x \in \mathsf{S}_{++}\right\}, \tag{6}$$

and $\lambda \in \mathbb{R}^n$ is the Lagrange multiplier vector. Because (6) is separable into scenarios, parallel processors can solve it effectively. In execution, the augmented Lagrangian method outperforms the Lagrangian dual method. We do not discuss the augmented Lagrangian method in this paper. Even so, we would like to emphasize that all results in Sections 3, 5 and 6 are valid for the augmented Lagrangian method because f is not required to be separable in these sections. As a result, the augmented term is easily incorporated into the nonlinear function f.

The fact that the function $\eta(\cdot)$ may not be strictly convex and differentiable is a considerable hindrance in solving the problems (5)–(6). To overcome this hindrance, we add a self-concordant logarithmic barrier function $-\log\det(x) = -\mathbb{E}^{\zeta}\log\det(x^{\zeta};\zeta)$ to the objective function in (6), where the barrier $-\log\det(\cdot\,;\zeta)$ is defined on the interior of the positive semidefinite cone, $\mathsf{S}^k_{++}(\zeta)$. The resulting problem can be written as:

$$\min_{\lambda} \eta(r,\lambda), \tag{7}$$

and

$$\eta(r,\lambda) := \max_{x\in\mathsf{S}_{++}} -f(x) + r\log\det X + \lambda^{\mathsf{T}}(Ax-b), \tag{8}$$

where $X := \mathrm{Mat}(x)$.

Note that we used the same symbol η for the original Lagrangian dual function defined in (6) as well as for the parametrized one defined in (8). This can facilitate thinking about their relation, and since they can be recognized from the context, it will not be confusing. We refer to (7) as the master problem, and refer to (8) as the subproblem. Since the logarithmic barrier function is separable, the subproblem $\eta(r,\lambda)$ defined in (8) is also separable. We note that the subproblem is an unconstrained problem because the optimal solution is a vectorization of a strictly positive definite matrix (due to the barrier).

Now, we compute the partial derivatives of the Lagrangian dual function $\eta(r,\cdot)$. Based on Assumption 2.2, it is obvious that the subproblem (8) has a unique solution for any λ and $r > 0$. This solution is denoted by $x(r,\lambda)$. Let

$$h(r,x) := f(x) - r\log\det X, \quad \text{and} \quad \rho(r,\lambda,x) := -h(r,x) + \lambda^{\mathsf{T}}(Ax-b).$$

It is known that (see [31] for example)

$$\nabla_x \log\det X = (X^{-1/2}\otimes X^{-1/2})\mathrm{vec}(I), \quad \text{and} \quad \nabla^2_{xx}\log\det X = X^{-1}\otimes X^{-1}.$$

It follows that

$$\nabla_x h(r,x) = \nabla_x f(x) - r\left(X^{-1/2}\otimes X^{-1/2}\right)\mathrm{vec}(I), \quad \text{and} \quad \nabla^2_{xx}h(r,x) = \nabla^2_{xx}f(x) + rX^{-1}\otimes X^{-1}. \tag{9}$$

Since $x(r,\lambda)$ maximizes $\rho(r,\lambda,x)$ for $x\in\mathsf{S}_+$, it satisfies the optimality condition

$$\nabla_x \rho(r,\lambda,x) = -\nabla_x h(r,x(r,\lambda)) + A^{\mathsf{T}}\lambda = 0. \tag{10}$$

That is,

$$-\nabla_x f(x) + r\left(X^{-1/2}\otimes X^{-1/2}\right)\mathrm{vec}(I) + A^{\mathsf{T}}\lambda = 0.$$

Differentiating (10) with respect to r and using the first equation in (9), we get

$$\begin{aligned} x'(r,\lambda) &= -\left[\nabla^2_{xx}h(r,x(r,\lambda))\right]^{-1}\nabla_x h'(r,x(r,\lambda)) \\ &= \left[\nabla^2_{xx}f(x(r,\lambda)) + rX^{-1}(r,\lambda)\otimes X^{-1}(r,\lambda)\right]^{-1}\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I). \end{aligned} \tag{11}$$

Differentiating (10) with respect to λ and using the second equation in (9), we get

$$\nabla_\lambda x(r,\lambda) = \left[\nabla^2_{xx}h(r,x(r,\lambda))\right]^{-1}A^{\mathsf{T}} = \left[\nabla^2_{xx}f(x(r,\lambda)) + rX^{-1}(r,\lambda)\otimes X^{-1}(r,\lambda)\right]^{-1}A^{\mathsf{T}}. \tag{12}$$

Here $X(r,\lambda) := \mathrm{Mat}(x(r,\lambda))$. The convex function $\eta(r,\cdot)$ has a gradient that can be written as

$$\nabla_\lambda \eta(r,\lambda) = Ax(r,\lambda) - b. \tag{13}$$

Differentiating (13) with respect to λ and using (12), we obtain

$$\nabla^2_{\lambda\lambda}\eta(r,\lambda) = A\nabla_\lambda x(r,\lambda) = A\left[\nabla^2_{xx}h(r,x(r,\lambda))\right]^{-1}A^\mathsf{T}, \tag{14}$$

and hence, using (12) again

$$\nabla^2_{\lambda\lambda}\eta(r,\lambda) = A\left[\nabla^2_{xx}f(x(r,\lambda)) + rX^{-1}(r,\lambda)\otimes X^{-1}(r,\lambda)\right]^{-1}A^\mathsf{T}. \tag{15}$$

Note that, based on Assumptions 2.1 and 2.2, the Hessian matrix $\nabla^2_{\lambda\lambda}\eta(r,\lambda)$, for any $r>0$, is positive definite, and hence the function $\eta(r,\cdot)$ is strictly convex.

3 Self-concordance

In this section, we show a fundamental property which is helpful for proving the convergence and analyzing the complexity of the proposed algorithm based on the self-concordance of the function family $\eta(r,\cdot)$. The proof is complicated by the fact that the function family $\eta(r,\cdot)$ is inherently described by the function f and the log-barrier function through the maximization.

The following definition was introduced by Nesterov and Nemirovskii [30, Definitions 2.1.1 and 3.2.1].

Definition 3.1 *Let $\mathcal{E}$ be a finite-dimensional real vector space, and $\mathbb{G}$ be an open nonempty convex subset of $\mathcal{E}$. Let $\ell:\mathbb{G}\longrightarrow\mathbb{R}$ and $f:\mathbb{G}\longrightarrow\mathbb{R}$ be convex C^3 functions. Let α, θ and β be positive constants.*

(i) ℓ is called an α-self-concordant function if for any $x\in int\,\mathbb{G}$, we have

$$|\nabla^3_{xxx}\ell(x)[d,d,d]| \le \frac{2}{\sqrt{\alpha}}\left(\nabla^2_{xx}\ell(x)[d,d]\right)^{3/2},\ \ \forall\, d\in\mathcal{E}.$$

The function ℓ is said to be strongly self-concordant if $\ell(x)$ tends to infinity for any sequence of x approaching the boundary of $\mathbb{G}$.

(ii) ℓ is called a θ-self-concordant barrier if it is a strongly 1-self-concordant function and for any $x\in int\,\mathbb{G}$ we have

$$|\nabla_x\ell(x)d| \le \sqrt{\theta}\left(\nabla^2_{xx}\ell(x)[d,d]\right)^{1/2},\ \forall d\in\mathcal{E}. \tag{16}$$

(iii) f is said to be β-compatible with a θ-self-concordant barrier ℓ if for any $x\in int\,\mathbb{G}$ we have

$$|\nabla^3_{xxx}f(x)[d,d,d]| \le \beta\left(3\nabla^2_{xx}f(x)[d,d]\right)\left(3\nabla^2_{xx}\ell(x)[d,d]\right)^{1/2},\ \forall d\in\mathcal{E}.$$

The reader is referred to [30,32,33] for further details about the self-concordance and associated properties. The following lemma is due to [30, Proposition 5.4.5].

Lemma 3.1 *The function $-\log\det(\cdot)$ is an sk-self-concordant barrier for the cone S_{++}.*

To analyze the convergence and complexity of the algorithm suggested in Section 5, we add the following assumption to the list of our assumptions.

Assumption 3.1 *The function $f:\mathbb{R}^{sk^2}\longrightarrow\mathbb{R}$ is β-compatible with the logarithmic barrier function $-\log\det(\cdot)$.*

Assumption 3.1 is used to show that the family of functions $h(r,\cdot)=f(\cdot)-r\log\det(\cdot)$ is self-concordant, which is required for the polynomial complexity of path-following methods. This assumption will be used to show that $\eta(r,\cdot)$ defined by (8) is a self-concordant family. Our work is challenging because the map $h(r,\cdot)$ is explicitly formulated by the function f, the log-barrier function $\log\det(\cdot)$, and the barrier parameter r, whereas the map $\eta(r,\cdot)$ is implicitly dependent on the function f, the log-barrier function $\log\det(\cdot)$, and the barrier parameter r through the maximization.

The convexity of the feasible region S_+ is necessary to guarantee the polynomial complexity. In fact, the intersection of the cone S_+ with a level set $\mathcal{L}_\sigma := \{x : f(x)\le\sigma\}$, for some constant σ, can always be used to replace S_+. It is known that the set of optimal solutions of Problem (4) is bounded if and only if $\mathsf{S}_+\cap\mathcal{L}_\sigma$ is compact (see also [7, Remark 4]). Therefore, the compactness of S_+ is also essential.

Lemma 3.2 *For any $r > 0$, the function $h(r, \cdot) = f(\cdot) - r \log\det(\cdot)$ is $\frac{r}{(1+\beta)^2}$-self-concordant. That is, for any $x \in \mathsf{S}_{++}$, we have*

$$\left|\nabla^3_{xxx} h(r,x)[d,d,d]\right| \leq \frac{2(1+\beta)}{\sqrt{r}} \left(\nabla^2_{xx} h(r,x)[d,d]\right)^{3/2}, \quad \forall\, d \in \mathbb{R}^{sk^2}. \tag{17}$$

Proof: The proof follows from Assumption 3.1 and using Lemma 3.1 and [30, Proposition 2.2.2]. □

The following theorem gives an important result that will be employed in establishing the iteration complexity of the proposed algorithm.

Theorem 3.1 *For any fixed $r > 0$, the function $\eta(r, \cdot)$ is $\frac{r}{(1+\beta)^2}$-self-concordant.*

Proof: Note that $\eta(r, \cdot)$ is essentially an affine transformation of $h(r, \cdot)$. The result immediately follows from Lemma 3.2 and [30, Proposition 2.1.1]. □

The parametrized functions of our problem must form a so-called self-concordant family in order to be solved in polynomial time. The following definition was introduced by Nesterov and Nemirovskii [30, Definition 3.1.1]. The definition uses $\mathbb{R}_{++}$ to denote the set of all positive real numbers.

Definition 3.2 *Let $\mathcal{E}$ be a finite-dimensional real vector space, and $\mathbb{G}$ be an open nonempty convex subset of $\mathcal{E}$. Let also $r \in \mathbb{R}_{++}$ and $f_r : \mathbb{R}_{++} \times \mathbb{G} \to \mathbb{R}$ be a family of functions indexed by r. Let $\alpha_1(r)$, $\alpha_2(r)$, $\alpha_3(r)$, $\alpha_4(r)$ and $\alpha_5(r) : \mathbb{R}_{++} \to \mathbb{R}_{++}$ be continuously differentiable functions on r. Then the family of functions $\{f_r\}_{r\in\mathbb{R}_{++}}$ is called* strongly self-concordant with the parameters *α_1, α_2, α_3, α_4, α_5, if the following conditions hold:*

(i) The function f_r is continuous on $\mathbb{R}_{++} \times \mathbb{G}$, and for fixed $r \in \mathbb{R}_{++}$, f_r is convex on $\mathbb{G}$ and has three partial derivatives on $\mathbb{G}$, which are continuous on $\mathbb{R}_{++} \times \mathbb{G}$ and continuously differentiable with respect to r on $\mathbb{R}_{++}$.

(ii) For any $r \in \mathbb{R}_{++}$, the function f_r is strongly $\alpha_1(r)$-self-concordant.

(iii) For any $(r, x) \in \mathbb{R}_{++} \times \mathbb{G}$ and any $d \in \mathcal{E}$,

$$\begin{aligned}
\left|\frac{\partial}{\partial r}\left(d^{\mathsf{T}}\nabla_x f_r(r,x)\right) - \frac{\partial}{\partial r}\left(\ln \alpha_3(r)\right)\, d^{\mathsf{T}}\nabla_x f_r(r,x)\right| &\leq \alpha_4(r)\left(\alpha_1(r)\right)^{\frac{1}{2}}\left(d^{\mathsf{T}}\nabla^2_{xx} f_r(r,x)d\right)^{\frac{1}{2}},\\
\left|\frac{\partial}{\partial r}\left(d^{\mathsf{T}}\nabla^2_{xx} f_r(r,x)d\right) - \frac{\partial}{\partial r}\left(\ln \alpha_2(r)\right)\, d^{\mathsf{T}}\nabla^2_{xx} f_r(r,x)d\right| &\leq 2\alpha_5(r)\, d^{\mathsf{T}}\nabla^2_{xx} f_r(r,x)d.
\end{aligned}$$

The following theorem presents a fundamental result that plays a central role in proving the polynomiality of the proposed algorithm.

Theorem 3.2 *The family of functions $\{\eta(r, \cdot) : r > 0\}$ is a strong self-concordant family with parameters:*

$$\alpha_1(r) = \frac{r}{(1+\beta)^2},\ \ \alpha_2(r) = 1,\ \ \alpha_3(r) = 1,\ \ \alpha_4(r) = \frac{(1+\beta)\sqrt{sk}}{r},\ \ \textit{and}\ \ \alpha_5(r) = \frac{(1+\beta)\sqrt{sk} + 1/2}{r}.$$

Proof: Theorem 3.1 shows that $\eta(r, \cdot)$ is $\alpha_1(r)$-self-concordant for any fixed $r > 0$. To establish the result, in light of Definition 3.2, it suffices to show that the following two inequalities hold for any $r > 0$ and $\lambda \in \mathbb{R}^n$:

$$\left|\nabla_\lambda \eta'(r,\lambda)^{\mathsf{T}} d\right| \leq \sqrt{\frac{sk}{r}\nabla^2_{\lambda\lambda}\eta(r,\lambda)[d,d]}, \quad \forall d \in \mathbb{R}^n, \tag{18}$$

$$\left|\nabla^2_{\lambda\lambda}\eta'(r,\lambda)[d,d]\right| \leq \frac{2(1+\beta)\sqrt{sk}+1}{r}\nabla^2_{\lambda\lambda}\eta(r,\lambda)[d,d], \quad \forall d \in \mathbb{R}^n. \tag{19}$$

First, we prove (18). Let $r > 0$ and $\lambda \in \mathbb{R}^n$. Using (13) and (11), we have

$$\nabla_\lambda \eta'(r,\lambda) = Ax'(r,\lambda) = A\left[\nabla^2_{xx} h(r, x(r,\lambda))\right]^{-1}\left(X^{-\frac{1}{2}}(r,\lambda) \otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I). \tag{20}$$

Then, for any $d \in \mathbb{R}^n$, we have

$$\begin{aligned}\left|d^\top \nabla_\lambda \eta'(r,\lambda)\right| &= \left|d^\top A\left[\nabla^2_{xx} h(r,x(r,\lambda))\right]^{-1}\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I)\right| \\ &\le \sqrt{\left[\nabla^2_{xx} h(r,x(r,\lambda))\right]^{-1}\left[\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I),\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I)\right]} \\ &\quad \times \sqrt{\left[\nabla^2_{xx} h(r,x(r,\lambda))\right]^{-1}[A^\top d, A^\top d]}. \end{aligned} \tag{21}$$

By the convexity of f, we have $\nabla^2_{xx} f(\cdot) \succeq 0$, hence

$$\nabla^2_{xx} h(r,x) = \nabla^2_{xx} f(x) - r\nabla^2_{xx}\log\det X \succeq -r\nabla^2_{xx}\log\det X = rX^{-1}\otimes X^{-1}, \tag{22}$$

and therefore

$$\left[\nabla^2_{xx} h(r,x(r,\lambda))\right]^{-1} \le \frac{1}{r}\left[X^{-1}(r,\lambda)\otimes X^{-1}(r,\lambda)\right]^{-1}.$$

It follows that

$$\begin{aligned}&\mathrm{vec}^\top(I)\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\left[\nabla^2_{xx} h(r,x(r,\lambda))\right]^{-1}\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I) \\ &\le \frac{1}{r}\left[X^{-1}(r,\lambda)\otimes X^{-1}(r,\lambda)\right]^{-1}\left[\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I),\left(X^{-\frac{1}{2}}(r,\lambda)\otimes X^{-\frac{1}{2}}(r,\lambda)\right)\mathrm{vec}(I)\right] \le \frac{sk}{r},\end{aligned} \tag{23}$$

where the last inequality was obtained by substituting $\ell(x) = -\log\det X$ in (16). As a consequence, using (21), (23), and then (14), we have

$$\left|d^\top \nabla_\lambda \eta'(r,\lambda)\right| \le \sqrt{\frac{sk}{r}}\sqrt{A\left[\nabla^2_{xx} h(r,x(r,\lambda))\right]^{-1}A^\top[d,d]} = \sqrt{\frac{sk}{r}}\sqrt{\nabla^2_{xx}\eta(r,\lambda)[d,d]},$$

which is the desired result in (18).

Now, we prove (19). Let $r > 0$ and $\lambda \in \mathbb{R}^n$, and define

$$\varphi(r) := \nabla^2_{\lambda\lambda}\eta(r,\lambda)[d,d].$$

In order to make it simple, we drop the arguments of h, x, X, and x' in the remaining part of the proof. So we simply write $h(r,x(r,\lambda))$ and $x'(r,\lambda)$, for instance, as h and x', respectively. From (14), we have

$$\varphi(r) = \left(\nabla^2_{xx} h\right)^{-1}\left[A^\top d, A^\top d\right].$$

Differentiating with respect to r, we get

$$\varphi'(r) = -\left(\nabla^2_{xx} h\right)^{-1}\left(\nabla^2_{xx} h\right)'\left(\nabla^2_{xx} h\right)^{-1}\left[A^\top d, A^\top d\right].$$

Letting $\bar{d} := (\nabla^2_{xx} h)^{-1}A^\top d$, we have

$$\varphi'(r) = -\left(\nabla^2_{xx} h\right)'\left[\bar{d},\bar{d}\right] = -\nabla^3_{xxx} h\left[x',\bar{d},\bar{d}\right] - \nabla^2_{xx}\log\det X\left[\bar{d},\bar{d}\right] = -\nabla^3_{xxx} h\left[x',\bar{d},\bar{d}\right] + \left(X^{-1}\otimes X^{-1}\right)\left[\bar{d},\bar{d}\right]. \tag{24}$$

Note that

$$\begin{aligned}\left|\nabla^3_{xxx} h\left[x',\bar{d},\bar{d}\right]\right| &\le \frac{2(1+\beta)}{\sqrt{r}}\sqrt{\nabla^2_{xx} h\left[x',x'\right]}\,\nabla^2_{xx} h\left[\bar{d},\bar{d}\right] \\ &= \frac{2(1+\beta)}{\sqrt{r}}\sqrt{\mathrm{vec}^\top(I)\left(X^{-\frac{1}{2}}\otimes X^{-\frac{1}{2}}\right)\left(\nabla^2_{xx} h\right)^{-1}\left(X^{-\frac{1}{2}}\otimes X^{-\frac{1}{2}}\right)\mathrm{vec}(I)}\,\varphi(r) \le \frac{2(1+\beta)\sqrt{sk}}{r}\,\varphi(r),\end{aligned}$$

where we used (17) and (23) to obtain the first and second inequalities, respectively, and used (11) to obtain the equality. In addition, from (22), we have

$$\left(X^{-1}\otimes X^{-1}\right)\left[\bar{d},\bar{d}\right] \le \frac{1}{r}\nabla^2_{xx} h\left[\bar{d},\bar{d}\right] = \frac{1}{r}\varphi(r).$$

Then, it follows from (24) that

$$\left|\varphi'(r)\right| \le \frac{2(1+\beta)\sqrt{sk}+1}{r}\,\varphi(r),$$

which is equivalent to the desired result in (19). The proof is complete. □

4 Efficient method for computing Newton directions

For a given barrier parameter $r > 0$, the optimality condition for the master problem (7) is

$$\nabla_\lambda \eta(r, \lambda(r)) = 0. \tag{25}$$

Finding the rate of change of the system (25) when λ changes (r is fixed), gives $\nabla^2_{\lambda\lambda}\eta(r,\lambda)\Delta\lambda + \nabla_\lambda \eta(r,\lambda) = 0$. Therefore, the solution to the system

$$\nabla^2_{\lambda\lambda}\eta(r,\lambda)\Delta\lambda = -\nabla_\lambda \eta(r,\lambda) \tag{26}$$

yields the Newton direction

$$\Delta\lambda = -\left[\nabla^2_{\lambda\lambda}\eta(r,\lambda)\right]^{-1} \nabla_\lambda \eta(r,\lambda) \tag{27}$$

used in the master problem (7), where $\nabla_\lambda \eta \in \mathbb{R}^n$ and $\nabla^2_{\lambda\lambda}\eta \in \mathbb{R}^{n\times n}$ are explicitly calculated in (13) and (15), respectively. Recall that k^2 is the dimension of x^ξ in (2), $s = |S|$ is the number of scenarios, and sk^2 is the dimension of x in Problem (4). The standard method for computing the inverse $[\nabla^2_{\lambda\lambda}\eta(r,\lambda)]^{-1}$ takes $O(k^6 s^3)$ arithmetic operations. Because s can be very large in practice, there is a need for a more efficient method for computing the Newton direction $\Delta\lambda$. This section describes a method for computing $\Delta\lambda$ that exploits the special structure of the nonanticipativity matrix A. The purpose of this section is to devise efficient methods to solve the equation (26). Using the structure of A, we will see that we can compute the Newton direction in $O(k^6 s)$ arithmetic operations.

We study the form of the nonanticipativity constraints (3). Without loss of generality, the computation of the Newton directions is illustrated in this section through a four-stage stochastic programming problem. Assume that there is a random variable ξ_t at stage t with s_t outcomes $\{\zeta_t^1, \zeta_t^2, \ldots, \zeta_t^{s_t}\}$ for each $t = 2, 3, 4$. Then $s = s_2 s_3 s_4$. The set of scenarios can be written as $S = \{\zeta^1, \zeta^2, \ldots, \zeta^s\}$ along with

$$\begin{aligned} S \;=\; \Big\{ & (\zeta_2^1, \zeta_3^1), \ldots, (\zeta_2^{s_2}, \zeta_3^1), \ldots, (\zeta_2^1, \zeta_3^{s_3}), \ldots, (\zeta_2^{s_2}, \zeta_3^{s_3}), \ldots, \\ & (\zeta_2^1, \zeta_3^1, \zeta_4^1), \ldots, (\zeta_2^{s_2}, \zeta_3^1, \zeta_4^1), \ldots, (\zeta_2^1, \zeta_3^{s_3}, \zeta_4^1), \ldots, (\zeta_2^1, \zeta_3^1, \zeta_4^{s_4}), \\ & \ldots, (\zeta_2^{s_2}, \zeta_3^{s_3}, \zeta_4^1), \ldots, (\zeta_2^{s_2}, \zeta_3^1, \zeta_4^{s_4}), \ldots, (\zeta_2^1, \zeta_3^{s_3}, \zeta_4^{s_4}), \ldots, (\zeta_2^{s_2}, \zeta_3^{s_3}, \zeta_4^{s_4}) \Big\}. \end{aligned}$$

Note that ζ_1 is disregarded because it is deterministic. Given this setting, the constraints in (3) are written as follow:

$$\begin{array}{rcrcrcrcl}
x_1^{\zeta^1} & - & x_1^{\zeta^2} & & & & & = & 0, \\
x_2^{(\zeta_2^i, \zeta_3^1)} & - & x_2^{(\zeta_2^i, \zeta_3^2)} & & & & & = & 0,\ i = 1, \ldots, s_2, \\
x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^1)} & - & x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^2)} & & & & & = & 0,\ i = 1, \ldots, s_2,\ j = 1, \ldots, s_3 - 1, \\
 & & x_1^{\zeta^2} & - & x_1^{\zeta^3} & & & = & 0, \\
 & & x_2^{(\zeta_2^i, \zeta_3^2)} & - & x_2^{(\zeta_2^i, \zeta_3^3)} & & & = & 0,\ i = 1, \ldots, s_2, \\
 & & x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^2)} & - & x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^3)} & & & = & 0,\ i = 1, \ldots, s_2,\ j = 1, \ldots, s_3 - 1, \\
 & & & & x_1^{\zeta^3} & - & x_1^{\zeta^4} & = & 0, \\
 & & & & x_2^{(\zeta_2^i, \zeta_3^3)} & - & x_2^{(\zeta_2^i, \zeta_3^4)} & = & 0,\ i = 1, \ldots, s_2, \\
 & & & & x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^3)} & - & x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^4)} & = & 0,\ i = 1, \ldots, s_2,\ j = 1, \ldots, s_3 - 1, \\
 & & & \ddots & & \ddots & & \vdots & \\
 & & & & x_1^{\zeta^{s_2 - 1}} & - & x_1^{\zeta^{s_2}} & = & 0, \\
 & & & & x_2^{(\zeta_2^i, \zeta_3^{s_3 - 1})} & - & x_2^{(\zeta_2^i, \zeta_3^{s_3})} & = & 0,\ i = 1, \ldots, s_2, \\
 & & & & x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^{s_4 - 1})} & - & x_3^{(\zeta_2^i, \zeta_3^j, \zeta_4^{s_4})} & = & 0,\ i = 1, \ldots, s_2,\ j = 1, \ldots, s_3 - 1.
\end{array}$$

Accordingly, the matrix $A \in \mathbb{R}^{n\times sk^2}$ has the following staircase structure.

$$A = \begin{bmatrix} B_1 & -B_1 & & & \\ & B_2 & -B_2 & & \\ & & \ddots & \ddots & \\ & & & B_{(s-1)} & -B_{(s-1)} \end{bmatrix}, \text{ where } B_i := [I, O] \in \mathbb{R}^{n_i \times k^2} \text{ for } i = 1, 2, \ldots, s-1.$$

Here, $\sum_{i=1}^{s-1} n_i = n$ and n_i is the number of components of x^{ζ^i} and $x^{\zeta^{i+1}}$ that are equal in the nonanticipativity constraints. For example, if $\zeta^i = (\zeta_2^1, \zeta_3^j, \zeta_4^k)$ and $\zeta^{i+1} = (\zeta_2^1, \zeta_3^{j+1}, \zeta_4^{k+1})$ with $j \le s_3 - 1$ and $k \le s_4 - 1$, then $x_1^{\zeta^i} = x_1^{\zeta^{i+1}}$, $x_2^{\zeta^i} = x_2^{\zeta^{i+1}}$, and $x_3^{\zeta^i} = x_3^{\zeta^{i+1}}$, hence $n_i = k_1^2 + k_2^2 + k_3^2$ in this case. For another example, if $\zeta^i = (\zeta_2^1, \zeta_3^j, \zeta_4^{s_4})$ and $\zeta^{i+1} = (\zeta_2^1, \zeta_3^{j+1}, \zeta_4^1)$ with $j \le s_3 - 1$, then $x_1^{\zeta^i} = x_1^{\zeta^{i+1}}$ and $x_2^{\zeta^i} = x_2^{\zeta^{i+1}}$, hence $n_i = k_1^2 + k_2^2$ in this case. For a third example, if $\zeta^i = (\zeta_2^1, \zeta_3^{s_3}, \zeta_4^{s_4})$ and $\zeta^{i+1} = (\zeta_2^1, \zeta_3^1, \zeta_4^1)$, then only $x_1^{\zeta^i} = x_1^{\zeta^{i+1}}$, hence $n_i = k_1^2$ in this case. Recall that $n_i \le k^2$ and A has full row rank (Assumption 2.1).

Because it is separable into scenarios, the function $h(x)$ can be written as $h(x) = \sum_{\zeta\in S} h_\zeta(x^\zeta)$. We can represent the Hessian matrix $\nabla_{xx}^2 h(x)$ by the block-diagonal matrix:

$$\nabla_{xx}^2 h(x) = \begin{bmatrix} H_1 & & & \\ & H_2 & & \\ & & \ddots & \\ & & & H_s \end{bmatrix}, \text{ where } H_i := \nabla^2_{x^\zeta x^\zeta} h_\zeta\left(x^\zeta\right) \in \mathbb{R}^{k^2\times k^2} \text{ for } i = 1, 2, \ldots, s. \tag{28}$$

Note that H_i are positive definite matrices for $i = 1, 2, \ldots, s$. Apparently $\nabla_{\lambda\lambda}^2 \eta(r, \lambda) = A[\nabla_{xx}^2 h(x)]^{-1} A^\mathsf{T}$ is the block-tridiagonal matrix:

$$\nabla_{\lambda\lambda}^2 \eta(r, \lambda) = \begin{bmatrix} B_1(H_1^{-1} + H_2^{-1})B_1^\mathsf{T} & -B_1 H_2^{-1} B_2^\mathsf{T} & & & & \\ -B_2 H_2^{-1} B_1^\mathsf{T} & B_2(H_2^{-1} + H_3^{-1})B_2^\mathsf{T} & -B_2 H_3^{-1} B_3^\mathsf{T} & & & \\ \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \\ & & -B_{s-2} H_{s-2}^{-1} B_{s-3}^\mathsf{T} & B_{s-2}(H_{s-2}^{-1} + H{-}1_{s-1})B_{s-2}^\mathsf{T} & -B_{s-2} H_{s-2}^{-1} B_{s-3}^\mathsf{T} & \\ & & & -B_{s-1} H_{s-1}^{-1} B_{s-2}^\mathsf{T} & B_{s-1}(H_1^{-1} + H_s^{-1})B_{s-1}^\mathsf{T} \end{bmatrix}. \tag{29}$$

Note that every submatrix in the Hessian matrix $\nabla_{\lambda\lambda}^2 \eta(r, \lambda)$ is no larger than k^2 by k^2, and all submatrices on the main diagonal are positive definite. Therefore, using a block Cholesky decomposition, this matrix can be decomposed as:

$$\begin{bmatrix} D_1 & R_1^\mathsf{T} & & \\ R_1 & D_2 & \ddots & \\ & \ddots & \ddots & R_{s-1}^\mathsf{T} \\ & & R_{s-1} & D_s \end{bmatrix} = \begin{bmatrix} I & & & \\ T_1 & I & & \\ & \ddots & \ddots & \\ & & T_{s-1} & I \end{bmatrix} \begin{bmatrix} P_1 & & & \\ & P_2 & & \\ & & \ddots & \\ & & & P_s \end{bmatrix} \begin{bmatrix} I & T_1^\mathsf{T} & & \\ & I & \ddots & \\ & & \ddots & T_{s-1}^\mathsf{T} \\ & & & I \end{bmatrix},$$

and the matrices in the decomposition are given as:

$$\begin{array}{lcllcl} P_1 &=& D_1, & & T_1 &=& P_1^{-1} R_1, \\ P_2 &=& D_2 - T_1 P_1 T_1^\mathsf{T}, & & T_2 &=& P_2^{-1} R_2, \\ & \vdots & & & & \vdots & \\ P_{s-1} &=& D_{s-1} - T_{s-2} P_{s-2} T_{s-2}^\mathsf{T}, & & T_{s-1} &=& P_{s-1}^{-1} R_{s-1}, \\ P_s &=& D_s - T_{s-1} P_{s-1} T_{s-1}^\mathsf{T}. & & & & \end{array}$$

Table 4.1: Complexity estimates for the steps required in the computation of $\Delta\lambda$ using the method of Section 4.

Steps required in the computation of the Newton direction $\Delta\lambda$	No. of arithmetic operations
i) Compute $[\nabla^2_{xx}h(x)]^{-1}$ using (28), i.e., compute H_i^{-1} for $i = 1, \ldots, s$	$O(k^6 s)$
ii) Compute the block-tridiagonal Hessian matrix $\nabla^2_{\lambda\lambda}\eta(r, \lambda)$ using (29)	$O(k^6 s)$
iii) Carry out the block Cholesky decomposition $\nabla^2_{\lambda\lambda}\eta(r, \lambda) = TPT^{\mathsf{T}}$	$O(k^6 s)$
iv) Solve the system $TPT^{\mathsf{T}}\Delta\lambda = -\nabla_\lambda\eta(r, \lambda)$ for the Newton direction $\Delta\lambda$	$O(k^4 s)$

The following theorem estimates the number of arithmetic operations required to compute $\Delta\lambda$.

Theorem 4.1 *By utilizing the method described in this section for computing the Newton direction $\Delta\lambda$, the number of arithmetic operations to be performed in the computation is $O(k^6 s)$ at each Newton iteration.*

Proof: Table 1 lists the steps required in computing the Newton direction $\Delta\lambda$ and shows the corresponding complexity estimates for the arithmetic operations involved in each step. Note that the most expensive step runs in $O(k^6 s)$. From the data presented in Table 1, it can be seen that the total number of arithmetic operations required for computing $\Delta\lambda$ is dominated by $O(k^6 s)$. The proof is complete. □

5 Algorithm

We have seen in the previous sections that, for any $r > 0$, the master problem (7) has unique solutions $\lambda(r)$ and the subproblem (8) has unique solutions $x(r, \lambda)$. The central path is defined as $\{x(r, \lambda(r)) : r > 0\}$. Let $\delta(r, \lambda) := \sqrt{r^{-1}\Delta\lambda^{\mathsf{T}}\nabla^2_{\lambda\lambda}\eta(r, \lambda)\Delta\lambda}$. Using (14) and (27), $\delta(r, \lambda)$ can be re-written as

$$\delta(r, \lambda) = \sqrt{r^{-1}[\nabla^2_{xx}h(r, x(r, \lambda))]^{-1}[A^{\mathsf{T}}\Delta\lambda, A^{\mathsf{T}}\Delta\lambda]} = \sqrt{r^{-1}\nabla_\lambda\eta^{\mathsf{T}}(\nabla^2_{\lambda\lambda}\eta)^{-1}\nabla_\lambda\eta}. \tag{30}$$

In the proposed algorithm, δ is a measure of the proximity of the current point λ to the central path, and ϵ_0 is a threshold for this measure. We formally state the proposed algorithm for SCSDP in Algorithm 5.1.

Algorithm 5.1: *A Lagrangian dual algorithm for Problem* (4)

Initialize $k, r_0, \lambda^{(0)}, x^{(0)}, \epsilon_0, \epsilon$
Ensure: $k = 0, r_0 > 0, \gamma \in (0, 1), \epsilon_0, \epsilon > 0$
for $k = 0, 1, 2, \ldots, N$ **do**
  while $r_k > \epsilon$ **do**
    set $r_{k+1} := \gamma r_k$
    maximize $\rho(r_k, \lambda^k, \cdot)$ to obtain $x(r_k, \lambda^k)$
    compute $\nabla_\lambda\eta(r_{k+1}, \lambda^k)$ using (13)
    compute $\nabla^2_{\lambda\lambda}\eta(r_{k+1}, \lambda^k)$ and find the Newton direction $\Delta\lambda^k$ by following Steps (*i*)-(*iv*) in Table 1
    compute $\delta(r_{k+1}, \lambda^k)$ using (30)
    choose a step size $\alpha \geq 0$ that minimizes $\eta(r_k, \cdot)$ along the direction $\Delta\lambda$
    set $\lambda^+ := \lambda^k + \alpha\Delta\lambda^k$
    if $\delta(r_{k+1}, \lambda^+) \leq \epsilon_0$ **then**
      set $\lambda^{k+1} := \lambda^+$
      Break
    else
    set $\lambda^k := \lambda^+$
    end if
  end while
end for

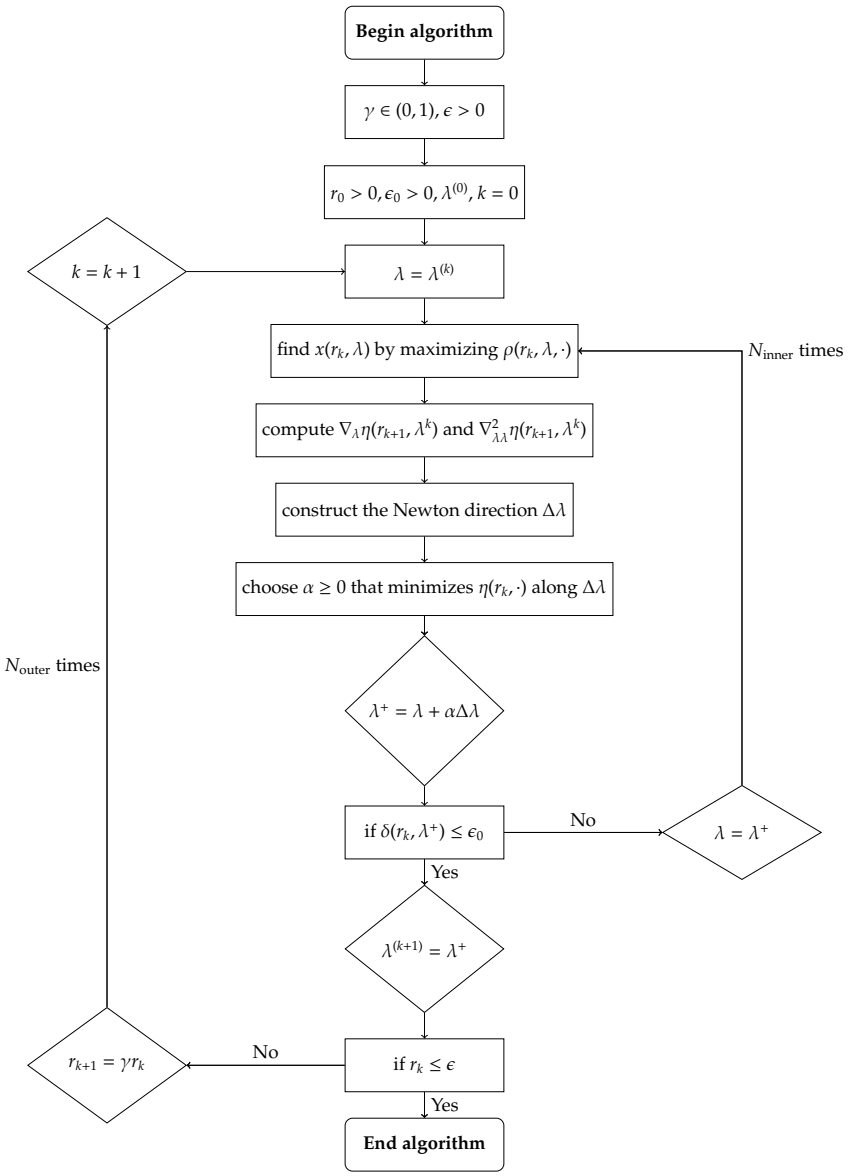

Figure 5.1: A flowchart of Algorithm 5.1.

Algorithm 5.1 is essentially a path-following algorithm. Figure 5.1 visualizes Algorithm 5.1 in a flowchart.

We use Newton method to maximize $\rho(r_k, \lambda, \cdot)$ to obtain $x(r_k, \lambda)$. Let $\{\zeta^1, \zeta^2, \ldots, \zeta^s\}$ represent the set of scenarios, and $(A^1, A^2, \ldots, A^s)$ represent the nonanticipativity matrix A accordingly. By solving the problem

$$\begin{array}{ll} \max & -f_0(x^{\zeta^j}; \zeta^j) - r_k \log\det(X^{\zeta^j}; \zeta^j) + \lambda^\mathsf{T}(A^j x^{\zeta^j} - a) \\ s.t. & x^{\zeta^j} \in \mathsf{S}^k_{++}(\zeta), \end{array} \tag{31}$$

for each $j = 1, \ldots, s$, we obtain $x^{\zeta^j}(r_k, \lambda)$. The we write

$$x(r_k, \lambda) = \begin{bmatrix} x^{\zeta^1}(r_k, \lambda) \\ x^{\zeta^2}(r_k, \lambda) \\ \vdots \\ x^{\zeta^s}(r_k, \lambda) \end{bmatrix}.$$

6 Convergence and complexity

In this section, we prove the convergence and present the time complexity of Algorithm 5.1. We show that the optimal solution can be approached by following the central path. In particular, when r approaches 0, the point $x(r, \lambda(r))$ tends to an optimal solution $x^\star$ of the multi-stage SCSDP problem (4). We will also show that the algorithm can be terminated when r is sufficiently small. In fact, we will see that the error of a solution is bounded by the barrier parameter. This explains why the stopping criterion in Algorithm 5.1 is $r \leq \epsilon$. We have the following two theorems.

Theorem 6.1 *As $r \to 0$, $x(r, \lambda(r))$ converges to an optimal solution $x^\star$ of the multi-stage SCSDP problem* (4)*.*

Proof: Consider the following barrier function problem.

$$\begin{array}{ll} \min & f(x) - r \log\det X \\ s.t. & Ax = b, \\ & x \in \mathsf{S}_{++}. \end{array} \tag{32}$$

It is known [34] that Problem (32) tends to an optimal solution $x^\star$ of Problem (4) as $r \to 0$. Denote $x(r) := x(r, \lambda(r))$. To prove the theorem, it suffices to show that $x(r)$ is the optimal solution to Problem (32).

Now, for fixed $r > 0$, $\lambda(r)$ minimizes the convex function $\eta(r, \lambda)$, and hence it satisfies the optimality condition:

$$\nabla_\lambda \eta(r, \lambda(r)) = Ax(r, \lambda(r)) - b = 0. \tag{33}$$

This proves the feasibility of $x(r)$ with respect to Problem (4).

Next, for any $r > 0$, $x(r)$ is the optimal solution to the maximization problem (8) with $\lambda = \lambda(r)$, and hence we have

$$\begin{array}{rcl} f(x(r)) + r \log\det X(r) & = & f(x(r)) - r \log\det X(r) - \lambda(r)^\mathsf{T}(Ax(r) - b) \\ & = & \min_{x \in \mathsf{S}_{++}} (f(x) - r \log\det X - \lambda(r)^\mathsf{T}(Ax - b)) \\ & \leq & \min\{f(x) - r \log\det X : Ax = b, x \in \mathsf{S}_{++}\}. \end{array}$$

This proves the optimality of $x(r)$ with respect to Problem (4). The proof is complete. □

Theorem 6.2 *Let $x^\star$ be the optimal solution to the multi-stage SCSDP problem* (4)*, and $x(r) := x(r, \lambda(\lambda))$. Let also $\varepsilon(r) := f(x(r)) - f(x^\star)$ be the error estimation. Then $0 \leq \varepsilon(r) \leq rsk$.*

Proof: By the feasibility of $x(r)$ with respect to Problem (4), we have $\varepsilon(r) \geq 0$. From Theorem 6.1, $x(r)$ tends to $x^\star$ as $r \to 0$. Therefore

$$\varepsilon(r) = \int_0^r (\nabla_x f(x(\tau)))^\mathsf{T} x'(\tau)\, d\tau. \tag{34}$$

Because $x(\tau)$ maximizes $-f(x) + \tau \log\det X + \lambda(\tau)^\mathsf{T} Ax$, we have

$$-\nabla_x f(x) + \tau \left(X^{-\frac{1}{2}}(\tau) \otimes X^{-\frac{1}{2}}(\tau)\right) \mathrm{vec}(I) + A^\mathsf{T}\lambda(\tau) = 0. \tag{35}$$

By (33), we have $Ax'(\tau) = 0$. It follows that

$$\varepsilon(r) = \int_0^r \tau\, \mathrm{vec}^\mathsf{T}(I) \left(X^{-\frac{1}{2}}(\tau) \otimes X^{-\frac{1}{2}}(\tau)\right) x'(\tau)\, d\tau. \tag{36}$$

Differentiating (35) with respect to τ again, we get

$$-\nabla^2_{xx} h(\tau, x(\tau)) x'(\tau) + \left(X^{-\frac{1}{2}}(\tau) \otimes X^{-\frac{1}{2}}(\tau)\right) \mathrm{vec}(I) + A^\mathsf{T}\lambda'(\tau) = 0,$$

which implies that

$$x'(\tau) = \left[\nabla^2_{xx} h(\tau, x(\tau))\right]^{-1} \left(X^{-\frac{1}{2}}(\tau) \otimes X^{-\frac{1}{2}}(\tau)\right) \mathrm{vec}(I) + \left[\nabla^2_{xx} h(\tau, x(\tau))\right]^{-1} A^\mathsf{T}\lambda'(\tau).$$

Using (20), we also obtain

$$\begin{aligned}\mathrm{vec}^{\mathsf{T}}(I)\left(X^{-\frac{1}{2}}(\tau)\otimes X^{-\frac{1}{2}}(\tau)\right)x'(\tau) &= \mathrm{vec}^{\mathsf{T}}(I)\left(X^{-\frac{1}{2}}(\tau)\otimes X^{-\frac{1}{2}}(\tau)\right)\left[\nabla^2_{xx}h(\tau,x(\tau))\right]^{-1}\left(X^{-\frac{1}{2}}(\tau)\otimes X^{-\frac{1}{2}}(\tau)\right)\mathrm{vec}(I)\\ &\quad+\left(\nabla_\lambda\eta'(\tau,\lambda(\tau))\right)^{\mathsf{T}}\lambda'(\tau).\end{aligned}$$

Now, $\nabla_\lambda\eta(\tau,\lambda(\tau))=0$ implies that $\nabla_\lambda\eta'(\tau,\lambda(\tau))+\nabla^2_{\lambda\lambda}\eta(\tau,\lambda(\tau))\lambda'(\tau)=0$. Due to the strictly convexity of $\eta(\tau,\cdot)$, for any $\tau>0$, we have $\left(\nabla_\lambda\eta'(\tau,\lambda(\tau))\right)^{\mathsf{T}}\lambda'(\tau)=-\lambda'(\tau)^{\mathsf{T}}\nabla^2_{\lambda\lambda}\eta(\tau,\lambda(\tau))\lambda'(\tau)\le 0$. Thus, using (23), we get

$$\mathrm{vec}^{\mathsf{T}}(I)\left(X^{-\frac{1}{2}}(\tau)\otimes X^{-\frac{1}{2}}(\tau)\right)x'(\tau)\le\mathrm{vec}^{\mathsf{T}}(I)\left(X^{-\frac{1}{2}}(\tau)\otimes X^{-\frac{1}{2}}(\tau)\right)\left[\nabla^2_{xx}h(\tau,x(\tau))\right]^{-1}\left(X^{-\frac{1}{2}}(\tau)\otimes X^{-\frac{1}{2}}(\tau)\right)\mathrm{vec}(I)\le\frac{sk}{\tau}.$$

The desired result immediately follows by substituting the above bound into (36). □

We have two versions of the algorithm based on the magnitude of the factor γ: The short- and long-step versions of the algorithm. If $\gamma=1-\omega/\sqrt{sk}$, where $\omega>0$ is a constant ($\omega=1$, for example), the algorithm is a short-step algorithm. If $\gamma\in(0,1)$ is an arbitrary constant and is independent of sk ($\gamma=0.1$, for example), the algorithm is a long-step algorithm. Let N_{inner} (N_{outer}) be an upper bound on the number of inner (respectively, outer) iterations performed by Algorithm 5.1 (see Figure 5.1). The short-step algorithm has small N_{inner} and large N_{outer}, while the long-step algorithm has large N_{inner} and small N_{outer}.

Now, we estimate the total number of arithmetic operations required to find an ϵ-approximate solution by each variant of algorithm. We will omit the proofs in the remaining part of this section because similar proofs for analogues results have been established in [7–9]. The total number of arithmetic operations needed by Algorithm 5.1 is bounded by the number of Newton iterations for the master problem, say N, multiplied by the number of arithmetic operations at each Newton iteration, K. Using the method described in Section 4 for computing a Newton direction for the master problem, we have found in Theorem 4.1 that the cost of this computation is $K=O(sk^6)$ at each Newton iteration.

Now, we turn to estimate N. At every complete iteration, the algorithm executes an outer iteration that updates the barrier parameter r by a factor γ, (in the sense that $r^{k+1}=\gamma r^k$), and then performs an inner loop with several inner iterations. Therefore, the total number of Newton iterations performed by Algorithm 5.1 is bounded by

$$N=N_{\text{outer}}N_{\text{inner}}.$$

Similar to (41) in [7], we can easily show that $N_{\text{outer}}=O(1)\sqrt{sk}\ln(r^{(0)}/\epsilon)$ for the short-step algorithm. Additionally, similar to (42) in [7], we can easily show that $N_{\text{outer}}=O(1)\ln(r^{(0)}/\epsilon)$ for the long-step algorithm. Furthermore, using Theorem 2.2.3 in [30] and the self-concordance results in Theorem 3.2 that we have established in this paper, we can also prove that N_{inner} is equal to one for the short-step algorithm, and that N_{inner} is $O(sk)$ for the long-step algorithm. Table 6.1 compares some criteria between the two variants of the algorithm.

Using the above estimations of N_{outer} and N_{inner}, we can present the complexity time bounds for the short- and long-step algorithms in Theorems 6.3 and 6.4, respectively. We will omit the proofs of these two theorems because the proof of Theorem 6.3 is an analogue of Theorems 4.1 in [8] (and Theorem 4 in [9] as well), and the proof of Theorem 6.4 is an analogue of Theorems 4.2 in [8] (and Theorem 5 in [9] as well).

Theorem 6.3 *Let $\epsilon_0=(2-\sqrt{3})/2$ and $\gamma=1-0.1/\sqrt{sk}$ be chosen in Algorithm 5.1. If $\delta(r^{(0)},\lambda^{(0)})\le\epsilon_0$, then the short-step algorithm terminates with at most $O(\sqrt{sk}\ln(r^{(0)}/\epsilon))$ Newton iterations.*

Theorem 6.4 *Let $\epsilon_0=1/6$ and $\gamma\in(0,1)$ be chosen in Algorithm 5.1. If $\delta(r^{(0)},\lambda^{(0)})\le\epsilon_0$, then the long-step algorithm terminates with at most $O\left(sk\ \ln(r^{(0)}/\epsilon)\right)$ Newton iterations.*

Theorems 6.3 and 6.4 demonstrate the polynomiality of the running time of Algorithm 5.1. Note that the complexity results obtained in Theorems 6.3 and 6.4 are the counterparts of the complexity results in Theorems 4.1 and 4.2 in [8] as well as in Theorems 4 and 5 in [9] for two-stage stochastic semidefinite programming.

Table 6.1: Comparison of some measures between the short- and long-step algorithms for SCSDP.

Computational measure	Short-step algorithm	Long-step algorithm
Threshold ϵ_0	$\epsilon_0 = 1 - \frac{\sqrt{3}}{2}$	$\epsilon_0 = \frac{1}{6}$
Magnitude of γ	$\gamma = 1 - \frac{\varpi}{\sqrt{sk}}, \varpi > 0$	Constant magnitude, $\gamma \in (0,1)$
Operations per iteration K	$K = O\left(sk^6\right)$	$K = O\left(sk^6\right)$
Outer iterations N_{outer}	$N_{\text{outer}} = O(1)\sqrt{sk}\ln\left(\frac{r^{(0)}}{\epsilon}\right)$	$N_{\text{outer}} = O(1)\ln\left(\frac{r^{(0)}}{\epsilon}\right)$
Inner iterations N_{inner}	Single inner iteration, $N_{\text{inner}} = 1$	$N_{\text{inner}} = O(sk)$
Total iterations N	$N = O\left(\sqrt{sk}\ln\left(\frac{r^{(0)}}{\epsilon}\right)\right)$	$N = O\left(sk\ln\left(\frac{r^{(0)}}{\epsilon}\right)\right)$
Total arithmetic operations	$O\left(s^{1.5}k^{6.5}\ln\left(\frac{r^{(0)}}{\epsilon}\right)\right)$	$O\left(s^2k^7\ln\left(\frac{r^{(0)}}{\epsilon}\right)\right)$

7 Numerical results

We consider the stochastic semidefinite programs with coefficients generated by random number generators. Writing in the abbreviated form, we consider the problem

$$\begin{array}{ll} \min & f(x) \\ & Ax \leq b, \\ \text{s.t.} & x \in \mathsf{S}_+, \end{array}$$

which has the form:

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & A_{00}x_0 & \leq b_0, \\ & A_{10}x_0 + A_{11}x_1 & \leq b_1, \\ & & \vdots \\ & A_{T0}x_0 + A_{T1}x_1 + \cdots + A_{TT}x_T & \leq b_T, \\ & l_t \leq x_t \leq u_t, \quad t = 0,1,\ldots,T, & \\ & x_t \in \mathbb{R}^{k_t^2}, \quad t = 0,1,\ldots,T, & \end{array}$$

where $b_0 \in \mathbb{R}^{n_0}, b_1 \in \mathbb{R}^{n_1}, \ldots, b_T \in \mathbb{R}^{n_T}$, $\sum_{t=0}^{T} n_t^2 = n$, $\sum_{t=0}^{T} k_t^2 = k^2$, and f is a convex function.

For simplicity, for each stage $t = 1,2,\ldots,T$, we assume $(A_{t1}, A_{t2}, \ldots, A_{tt}, b_t)$ has s_t realizations. Realizations of each entry of A_{ti} are sampled from the uniform distribution on the interval $[-0.5, 0.5]$. In order to keep consistent, realizations of b_t are generated by $b_t = A_{t0}e_0 + A_{t1}e_1 + \cdots + A_{tt}e_t + \hat{b}_t$, where $\hat{b}_t \in \mathbb{R}^{n_t}$ is a vector with all its entries chosen from a uniform distribution on the interval $[0,1]$. We also assume that all scenarios occur with the same probability, where $e_t = \text{vec}(I_t)$, is a vector with dimension k_t^2 for $t = 0,1,\cdots,T$. The vector l_t (respectively, u_t) is assumed to be a vector of entries zeros (respectively, hundreds) for $t = 0,1,\cdots,T$. We assume that all the coefficients from stage 1 onward are random variables. Therefore, $x = (\text{vec}(I_0), \text{vec}(I_2), \ldots, \text{vec}(I_T))$ is feasible for this program.

As a concrete example, Figure 7.1 illustrates in a tree this program when we have three stages and six possible scenarios. See also Figure 7.2 which illustrates this program when we generally have T stages and s possible scenarios and visualizes the problem in a tree consisting of a set of nodes, say $\mathcal{N}$.

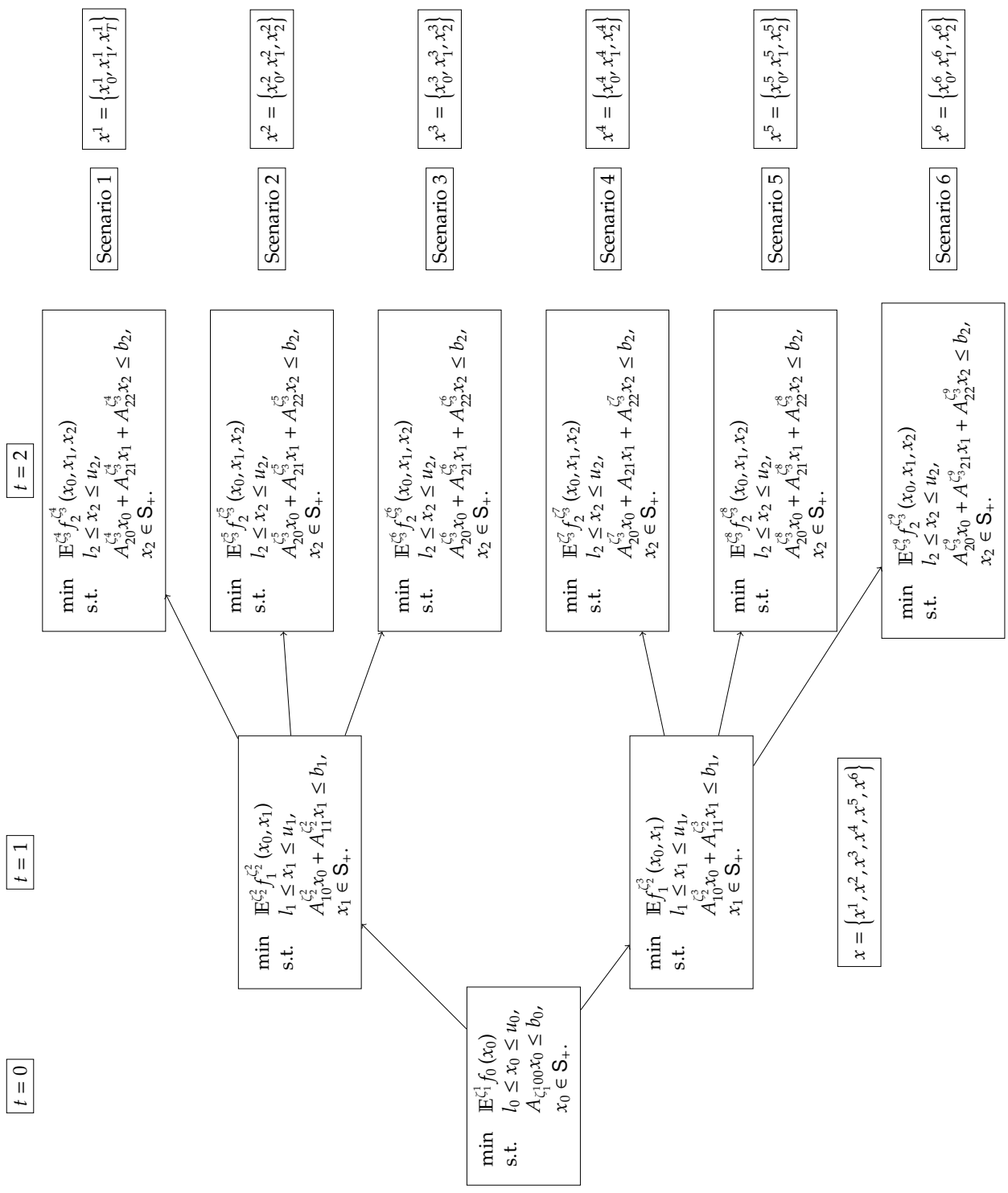

Figure 7.1: A tree analysis for three stages and six scenarios.

Table 7.1: Randomly generated multi-stage stochastic convex semidefinite programming test problems.

Problem name	No. of stages T	No. of scenarios s	No. of nonanticipativity constraints	Self-concordance parameter sk	Problem size $n \times sk^2$
SCSDP1	2	4	10	20	48×100
SCSDP2	3	4	18	24	72×144
SCSDP3	4	4	34	32	96×256
SCSDP4	2	9	20	45	120×225
SCSDP5	3	16	54	96	150×576
SCSDP6	4	16	82	128	180×1024
SCSDP7	4	36	162	216	200×1296
SCSDP8	4	64	274	384	240×2304

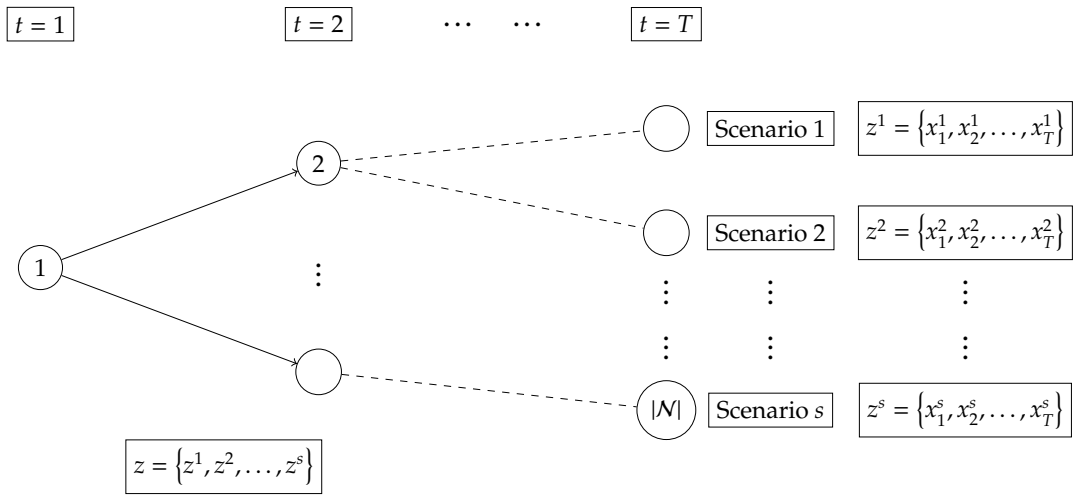

Figure 7.2: A tree analysis for T stages and s scenarios.

We implement Algorithm 5.1 on eight randomly generated multi-stage SCSDP test problems to show the algorithm effectiveness and efficiency. These randomly generated test problems are displayed in Table 7.1.

We consider two convex objective functions. We first consider the linear objective function: $f(x) = c^{\mathsf{T}}x = \sum_{t=1}^{T} c_t^{\mathsf{T}} x_t$, where the coefficient cost vector c is taken to be

$$c = \left(c_1^{\mathsf{T}}, c_2^{\mathsf{T}}, \ldots, c_T^{\mathsf{T}}\right)^{\mathsf{T}} = \left(-\mathrm{vec}^{\mathsf{T}}(I_1), -\mathrm{vec}^{\mathsf{T}}(I_2), \ldots, -\mathrm{vec}^{\mathsf{T}}(I_T)\right)^{\mathsf{T}}, \text{ and } I_t \in \mathbb{R}^{k_t \times k_t}, t = 1, 2, \ldots, T.$$

We also consider the following logarithmic objective function: $f(x) = -\log\det X$, where $X = \mathrm{Mat}(x)$.

For each objective function, we solved the corresponding system $\nabla^2_{\lambda\lambda}\eta(r,\lambda)\Delta\lambda = -\nabla_\lambda \eta(r,\lambda)$ and found the Newton direction $\Delta\lambda$ using the method described in Section 4.

Let $q(r_k, \lambda, x^{\xi^j})$ be the objective function of Problem (31). We used the Newton method to find the search directions. We have

$$\begin{aligned}
q\left(r_k, \lambda, x^{\xi^j}\right) &= -f(x^{\xi^j}; \xi^j) + r_k \log\det X^{\xi^j} + \lambda^{\mathsf{T}}\left(A^j x^{\xi^j} - b\right), \\
\nabla_{x^{\xi^j}} q\left(r_k, \lambda, x^{\xi^j}\right) &= -\nabla_{x^{\xi^j}} f(x^{\xi^j}, \xi^j) + r_k\left(\left(X^{\xi^j}\right)^{-1/2} \otimes \left(X^{\xi^j}\right)^{-1/2}\right)\mathrm{vec}(I_k) + {A^j}^{\mathsf{T}}\lambda, \\
\nabla^2_{x^{\xi^j} x^{\xi^j}} q\left(r_k, \lambda, x^{\xi^j}\right) &= -\nabla^2_{x^{\xi^j} x^{\xi^j}} f\left(x^{\xi^j}, \xi^j\right) - r_k\left(X^{\xi^j}\right)^{-1} \otimes \left(X^{\xi^j}\right)^{-1}, \\
\Delta x^{\xi^j} &= -\left\{\nabla^2_{x^{\xi^j} x^{\xi^j}} q\left(r_k, \lambda, x^{\xi^j}\right)\right\}^{-1} \nabla_x q\left(r_k, \lambda, x^{\xi^j}\right).
\end{aligned}$$

The computational results were obtained by running MATLAB version 9.8 on windows (12-core, Intel Xeon E5-2680 @ 2.50GHz, 128 G RAM). We also took the following values for the algorithm parameters: $\gamma = 0.1$, we chose the step size α using a heuristic, which usually determined α within two to three trials, and $\epsilon = 10^{-6}$, and $\epsilon_0 = 0.5$. The obtained numerical results are displayed in Tables 7.2 and 7.3 for the linear and logarithmic semidefinite test problems, respectively. In Tables 7.2 and 7.3, CPU stands for the computational time taken to obtain the ϵ-optimal solution, NIT stands for the number of iterations, and NSS stands for the number of scenario subproblems solved. The numerical results reported in Table 7.2 are also visualized in Figures 7.3 and 7.5, and the numerical results reported in Table 7.3 are also visualized in Figures 7.4 and 7.6. From these results, it is clear that the CPU time increases when scenarios and stages increase, most notably when k is very large. It can also be seen that, when s increases, the number of iteration of sub scenarios increase. This totally agrees with the theoretical findings reported in Theorems 6.3 and 6.4.

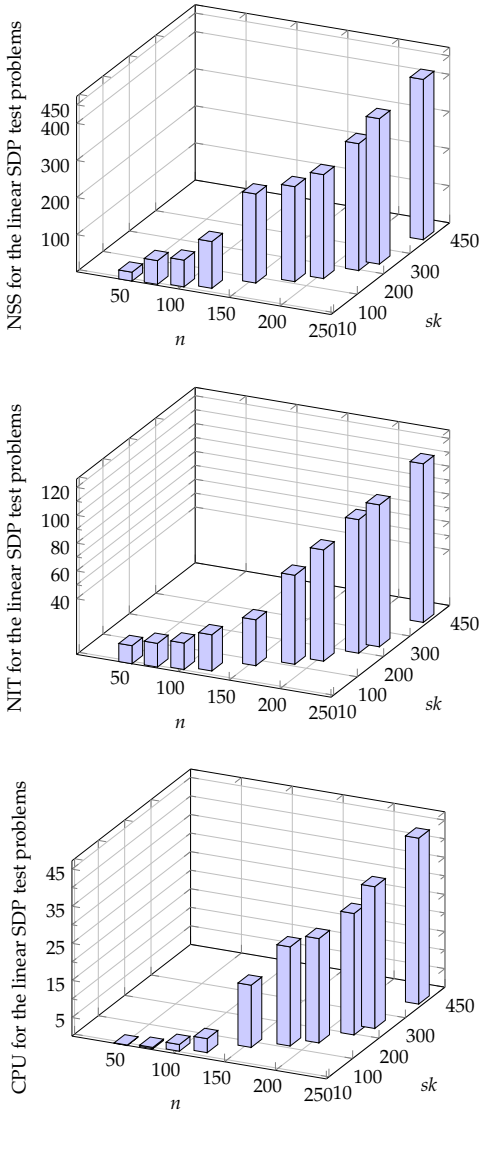

Figure 7.3: The numerical results for the linear semidefinite test problems in Table 7.2 are translated into three dimensional bar charts for comparison purposes.

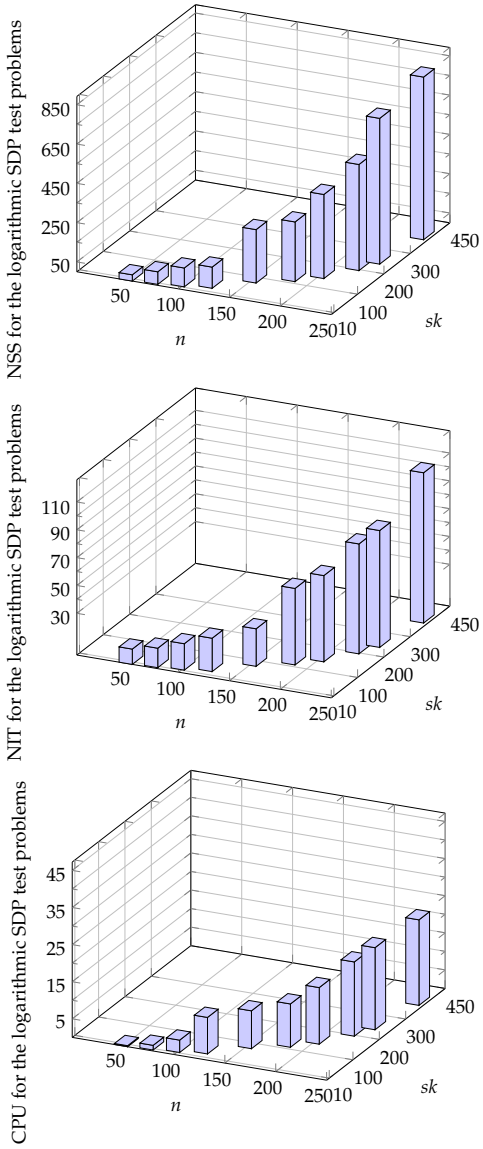

Figure 7.4: The numerical results for the logarithmic semidefinite test problems in Table 7.3 are translated into three dimensional bar charts for comparison purposes.

Table 7.2: Numerical results reported by Algorithm 5.1 for the test problems in Table 7.1 with the linear objective $f(x) = \sum_{t=1}^{T} c_t^{\mathsf{T}} x_t$.

Problem	CPU	NIT	NSS
SCSDP1	0.1092	13	24
SCSDP2	0.3588	17	64
SCSDP3	1.8096	19	72
SCSDP4	3.7800	26	126
SCSDP5	16.7679	33	256
SCSDP6	26.7160	80	360
SCSDP7	32.7160	96	342
SCSDP8	44.7160	114	432

Table 7.3: Numerical results reported by Algorithm 5.1 for the test problems in Table 7.1 with the logarithmic objective $f(x) = -\log\det X$.

Problem	CPU	NIT	NSS
SCSDP1	0.2808	11	32
SCSDP2	1.2638	14	64
SCSDP3	3.3354	19	96
SCSDP4	9.8437	24	108
SCSDP5	10.2025	27	272
SCSDP6	11.6845	55	304
SCSDP7	20.0149	79	540
SCSDP8	23.0556	108	828

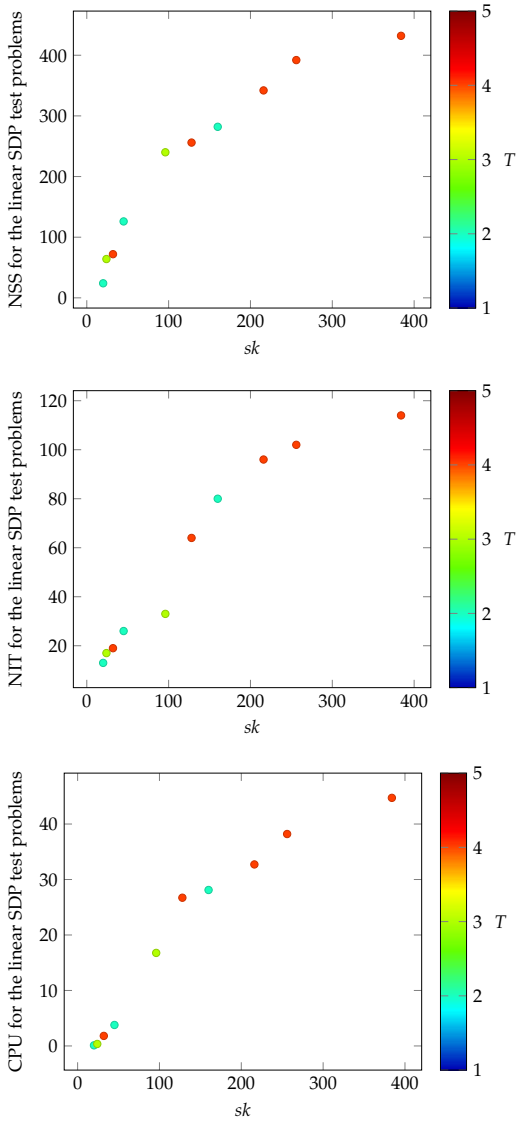

Figure 7.5: Dot plots of the numerical results obtained for the test linear semidefinite problems.

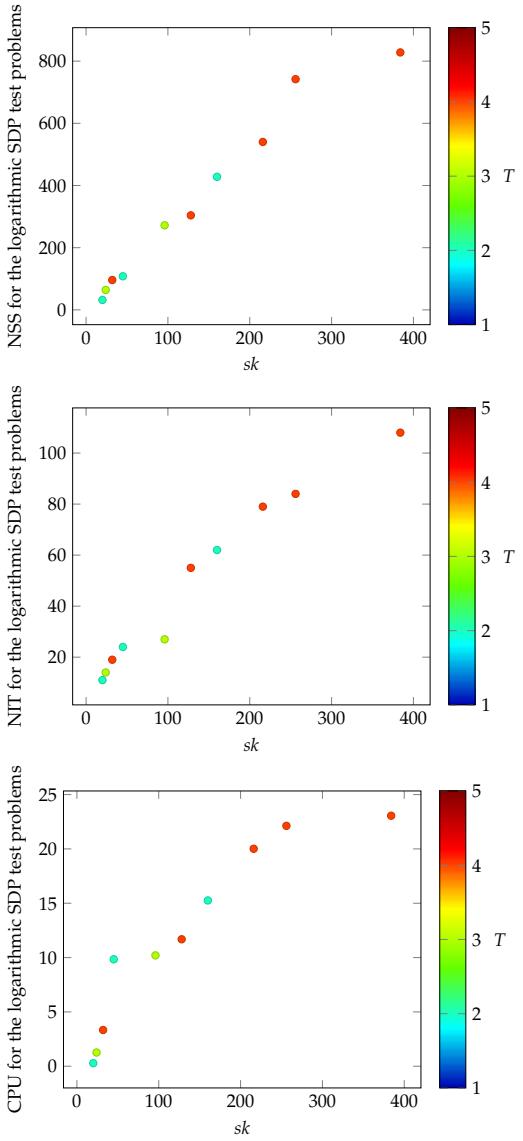

Figure 7.6: Dot plots of the numerical results obtained for the test logarithmic semidefinite problems.

8 Concluding remarks and future research

The cone of symmetric positive semidefinite matrices is known to have one of the most understandable conical structures among all convex cones. Due to its broad potential applications, many in engineering, semidefinite optimization has been one of the most active research areas in mathematical programming since the nineties of the previous century, especially in the context of barrier methods for optimization.

In this paper, we have presented a polynomial-time barrier algorithm for solving multi-stage stochastic convex semidefinite optimization by using the Lagrangian dual method and relaxing the nonanticipativity constraints. We have shown that the barrier Lagrangian dual functions form self-concordant families with respect to barrier parameters. This has enabled us to prove the polynomiality of the proposed algorithm. We have also described a method for computing the Newton direction which exploits the special structure of the nonanticipativity matrix and hence it is more efficient than the standard methods. We have ended the paper by testing the derived algorithm on different test problems to demonstrate its efficiency. After this work and that in [7], it has become clear that methods based on Lagrangian dual can be more effective than methods based on the Benders decomposition or those based on the deterministic equivalence, especially in the context of multi-stage stochastic convex programming.

To the best of our knowledge, there are no published studies on the finite-dimensional semidefinite optimization. An open question in this research line is whether it is possible to establish a theory of self-concordance for infinite-dimensional semidefinite operators? If the answer is affirmative, that sounds very interesting (if one can achieve a finite barrier parameter for worthwhile instances!). One could deal with infinite-dimensional operator domains and probably use the notion of regularized determinant for establishing the self-concordance analysis. We commend the interested researchers for looking into this possibility. In fact, one of the chief attractions of semidefinite optimization is its diverse applications. So, if one can also identify a good application leading to infinite-dimensional semidefinite programs, that would be a great motivator for further study. As for references, Faybusovich's work [35–37] on the infinite-dimensional programming may be relevant.

References

1. Griffin, J., Omheni, R.: A primal–dual modified log-barrier method for inequality constrained nonlinear optimization. Optim. Lett. **14**, 2461–2477 (2020)

2. Durea, M., Strugariu, R.: A barrier method in convex vector optimization with generalized inequality constraints. Optim. Lett. **14**, 759–769 (2020)

3. Darvay, Z., Illés, T., Majoros, C.: Interior-point algorithm for sufficient LCPs based on the technique of algebraically equivalent transformation. Optim. Lett. **15**, 357–376 (2021)

4. Roy, S., Xiao, L.: On self-concordant barriers for generalized power cones. Optim. Lett. (2021)

5. Moslemi, M., Kheirfam, B.: Complexity analysis of infeasible interior-point method for semidefinite optimization based on a new trigonometric kernel function. Optim. Lett. **13**, 127–145 (2019)

6. Zhang, M., Yuan, B., Zhou, Y.,Luo, X., Huang, Z.: A primal-dual interior-point algorithm with arc-search for semidefinite programming. Optim. Lett. **13**, 1157–1175 (2019)

7. Zhao, G.: A Lagrangian dual method with self-concordant barriers for multi-stage stochastic convex programming. Math. Program. **102**, 1–24 (2005)

8. Mehrotra, S., Ozevin, M.G.: Decomposition-based interior point methods for two-stage stochastic semidefinite programming. SIAMJ. Optim. **18**, 206–222 (2007)

9. Ariyawansa, K. A., Zhu, Y.A.: Class of polynomial volumetric barrier decomposition algorithms for stochastic semidefinite programming. Math. Comp. **80**, 1639–1661 (2010)

10. Gassmann, H.I.: MSLiP: A computer code for the multi-stage stochastic linear programming problem. Math. Program. **47**, 407–423 (1990)

11. Pereira, M.V.F., Pinto, L.M.V.G.: Multi-stage stochastic optimization applied to energy planning. Math. Program. **52**, 359–375 (1991)

12. Liu, X., Toh, K.C., Zhao, G.: On the implementation of a log-barrier progressive hedging method for multi-stage stochastic programs. J. Comput. Appl. Math . **234**, 579–592 (2010)

13. Zhu, Y., Ariyawansa, K.A.: A preliminary set of applications leading to stochastic semidefinite programs and chance-constrained semidefinite programs. Appl. Math. Model. **35**, 2425–2442 (2011)

14. Kwon, R., Li, J.: A stochastic semidefinite programming approach for bounds on option pricing under regime switching. Ann. Oper. Res. **237**, 41–75 (2016)

15. Esmaeili, M., Saad, H., Nosratinia, A.: Exact recovery by semidefinite programming in the binary stochastic block model with partially revealed side information. 2019 IEEE Int. Conf. Acoust. Speech Signal Process. 3477–3481 (2019)

16. Jin, S., Ariyawansa, K.A., Zhu, Y.: Homogeneous Self-dual Algorithms for Stochastic Semidefinite Programming. J. Optim. Theory Appl. **155**, 1073–1083 (2012)

17. Alzalg, B.: Homogeneous Self-dual Algorithms for Stochastic Semidefinite Programming. J. Optim. Theory Appl. **163**, 148–164 (2014).

18. Alzalg, B., Badarneh, K., Ababneh, A.: An infeasible interior-point algorithm for stochastic second-order cone optimization. J. Optim. Theory Appl. **181**, 148–164 (2019).

19. Zhao, G.: A log-barrier method with Benders decomposition for solving two-stage stochastic programs. Math. Program. **90**, 507–536 (2001)

20. Chen, M., Mehrotra, S.: Self-concordance and decomposition based interior point methods for the two stage stochastic convex optimization problem. SIAM J. Optim. **21**, 1667–1687 (2011).

21. Alzalg, B.: Decomposition-based interior-point methods for stochastic quadratic second order cone programming. Appl. Math. Comput. **249**, 1–18 (2014).

22. Alzalg, B., Ariyawansa, K.A.: Logarithmic barrier decomposition-based interior-point methods for stochastic symmetric programming. J. Math. Anal. Appl. **409**, 973–995 (2014).

23. Alzalg, B.: Volumetric barrier decomposition algorithms for stochastic quadratic second-order cone programming. Appl. Math. Comput. **256**, 494–508 (2015).

24. Alzalg, B., Gafour, A., Alzaleq, L.: Volumetric barrier cutting plane algorithms for stochastic linear semi-infinite optimization. IEEE Access. **8**, 4995–5008 (2020).

25. Alzalg, B.: Logarithmic-barrier decomposition interior-point methods for stochastic linear optimization in a Hilbert space. Numer. Func. Anal. Optim. **41**, 901–928 (2020).

26. Rockafellar, R.T., Wets, R.J.B.: Scenarios and policy aggregation in optimization under uncertainty. Math. Oper. Res. **16**, 119–147 (1991)

27. Mulvey, J.M., Ruszczynski, A.: A new scenario decomposition method for large scale stochastic optimization. Oper. Res. **43**, 477–490 (1995)

28. Mulvey, J.M., Vladimirou, H.: Applying the progressive hedging algorithm to stochastic generalized networks. Ann. Oper. Res. **31**, 399–424 (1991)

29. Ruszczynski, A.: On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. Math. Oper. Res. **20**, 634–656 (1995)

30. Nesterov,Y., Nemirovskii, A.: Interior-point polynomial algorithms in convex programming. SIAM, Philadelphia. (1994)

31. Todd, M.J.: Semidefinite optimization. Acta Num. **10**, 515–560 (2001)

32. Hertog, Den, D., Jarre, F., Roos, C., Terlaky, T.: A sufficient condition for self-concordance, with application to some classes of structured convex programming problems. Math. Program. **69**, 75–88 (1995)

33. Jarre, F.: Interior-point methods for convex programming. Appl. Math. Optim. **26**, 287–311 (1992)

34. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms, 2nd Edition. John Wiley and Sons. (1993)

35. Faybusovich, L., Moore, J.B.: Infinite-dimensional quadratic optimization: Interior-point methods and control applications. Appl. Math. Optim. **36**, 43–66 (1997)

36. Faybusovich, L., Moore, J.B.: Long-step path-following algorithm for convex quadratic programming problems in a Hilbert space. J. Optim. Theory Appl. **95**, 615–635 (1997)

37. Faybusovich, L., Tsuchiya, T.: Primal-dual algorithms and infinite-dimensional Jordan algebras of finite rank, Math. Program. **97**, 471–493 (2003)