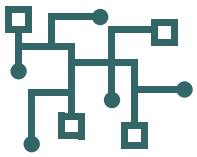

ISE

Industrial and
Systems Engineering

Generating Linear, Semidefinite, and Second-order Cone Optimization Problems for Numerical Experiments

MOHAMMADHOSSEIN MOHAMMADISIAHROUDI[1], RAMIN FAKHIMI[1], BRANDON AUGUSTINO[1], AND TAMÁS TERLAKY[1]

[1]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

ISE Technical Report 23T-010

LEHIGH
UNIVERSITY

Generating Linear, Semidefinite, and Second-order Cone Optimization Problems for Numerical Experiments

Mohammadhossein Mohammadisiahroudi[a], Ramin Fakhimi[a], Brandon Augustino[a], and Tamás Terlaky[a]

[a]Industrial and System Engineering Department, Lehigh University, Bethlehem, PA, USA

ARTICLE HISTORY
Compiled February 1, 2023

ABSTRACT
The numerical performance of algorithms can be studied using test sets or procedures that generate such problems. This paper proposes various methods for generating linear, semidefinite, and second-order cone optimization problems. Specifically, we are interested in problem instances requiring a known optimal solution, a known optimal partition, a specific interior solution, or all these together. In the proposed problem generators, different characteristics of optimization problems, including dimension, size, condition number, degeneracy, optimal partition, and sparsity, can be chosen to facilitate comprehensive computational experiments. We also develop procedures to generate instances with a maximally complementary optimal solution with predetermined optimal partition to generate challenging semidefinite and second-order cone optimization problems. Generated instances enable us to evaluate efficient interior-point methods for conic optimization problems.

KEYWORDS
Problem Generator; Conic Optimization; Linear Optimization; Semidefinite Optimization; Second-order Cone Optimization

1. Introduction

Optimization is just one of many fields in which the empirical analysis of algorithms is heavily reliant on the quality of the provided test instances. Scholars assess the strengths and weaknesses of algorithms based on these test problems, which must be unbiased, representative, and diverse in their measurable features or characteristics. However, many benchmark test problems do not possess these desired qualities, as they are often based on a limited set of real-world problems or have been reused from earlier studies that by now may be obsolete [2].

An alternative approach is using random test problem generators for experimentation in optimization. While their design must be carefully considered, one advantage of simple random generation approaches is their ability to produce problems that possess predictable characteristics. As a result, scientists have advocated for using highly parameterized generators to produce appropriately controlled data for experimentation [11]. As one of the first attempts in this area, randomly generated feasible polyhedra properties were investigated by Todd [24]. Pilcher and Rardin [17] proposed a

Corresponding author: M. M., mom219@Lehigh.edu

generator for pure integer optimization problems with a known partial polytope by introducing random cuts. Yet, this methodology is restricted to traveling salesman problems and does not explicitly consider the solution of relaxation or structural features. Lacking the ability to vary features of interest, the scope of these generators for experimentation is limited to specific problem domains.

At times, it can be challenging to develop instance generators in a way that allows properties of interest to be suitably varied. While specific characteristics, such as the density of a graph, can usually be directly controlled through the generation process, other attributes can be harder to predefine or control explicitly. Many measurable features of the same problem instance can be highly correlated, either due to interacting bounds or simply as a consequence of the random generation process. Instances with less-like feature combinations can be attained through an iterative local search, which successively modifies an instance until it possesses the desired properties. While these instance-space search techniques are more computationally intensive than parameterized generators, they provide a reliable method for producing instances with specific target characteristics [2].

The most prevalent search techniques for this application are evolutionary algorithms. Chakraborty and Choudhury [5] and Cotta and Moscato [8] applied this approach to perform statistical average- and worst-case analysis of algorithm performance. More recently, exploration in this direction has focused on improving the spectrum of instance hardness and diversity of measured features [20]. The success of these techniques in combinatorial optimization opens up questions on the use of similar approaches for linear optimization (LO) and mixed-integer optimization, adopting a more comprehensive range of search algorithms for obtaining difficult-to-design instances, and considering how to best construct the search space for efficient performance.

To develop instance generation techniques for LO test problems with controllable properties, Bowly et al. [2] presented a comparison of a naive random generator with a highly parameterized generator, showing which feature values can be effectively controlled by each method. They also investigated iterative search approaches to find instances that are difficult to design or rarely produced by the generator. These approaches allow practitioners to explore areas of interest in the space of linear optimization problems (LOPs), where challenging instances have previously been found. This would be impossible using static test sets or näive random generation methods, which provide limited feature control. Further, large-scale linear optimization problems are prevalent in economics, industry, logistics, statistics, quantum physics, and other fields. As is the case with any real-world application, the aim is to obtain high-quality solutions efficiently, a task for which high-performance computing systems and parallel algorithms are required. Thus, the development of new parallel algorithms for generating LOPs and the revision of current algorithms are considered by Sokolinsky and Sokolinskaya [21].

Developing new algorithms for solving large-scale LOPs necessitates testing them on benchmark and random problems. At times, it is sensible to construct linear and integer optimization instance generators specified for special purposes. The NETGEN generator [12] and its successor MNETGEN produce parameterized multicommodity flow, transport, and assignment problems. The parameters used are thus appropriate to the underlying network, not the feasible set. One of the well-known benchmark repositories of LOPs is Netlib-LP [10]. Yet, when debugging LO solvers, generating random LOPs with specific characteristics (such as, e.g., the sparsity, condition number of the coefficient matrix, or a known optimal partition) is often necessary.

Charnes et al. [6] suggested one of the first methods for generating random LOPs

with known solutions. This method allows one to generate test problems of arbitrary size with a wide range of numerical characteristics. The main idea of the method is as follows; take as a basis a LOP with a known solution, and then randomly modify it so that the solution does not change. The key drawback of this approach is that fixing the optimal solution in advance significantly restricts the random nature of the resulting LOP.

Arthur and Frendewey [1] described the GENGUB generator, which constructs random LOPs with a known solution and given characteristics, such as the problem size, the density of the coefficient matrix, the number of binding inequalities, or the degeneracy status. A distinctive feature of GENGUB is the ability to introduce generalized upper bound constraints, defined to be a (sub)set of constraints in which each variable appears at most once (i.e., has at most one nonzero coefficient). This method has similar drawbacks to the generator found in [6]: by fixing the optimal solution ex ante, the random nature of the resulting LOP is significantly restricted.

Castillo et al. [4] suggest a method for generating random LOPs with a preselected solution type: bounded or unbounded, unique or multiple. Each structure is generated using random vectors with integer components, whose range can be treated as given. Next, an objective function that satisfies the required conditions, i.e., leads to a solution of the desired type, is obtained. This LO problem generator is mainly used for educational purposes rather than testing new LO algorithms. Okolinsky and Sokolinskaya [21] proposed the random LOP generator FRaGenLP (Feasible Random Generator of LP), which is implemented as a parallel program for cluster computing systems. Calamai et al. [3] described a new technique for generating convex, strictly concave, and indefinite (bilinear or not) quadratic optimization problems.

In the semidefinite optimization literature, scholars were interested in complex problems. They pursued various directions for characterizing what constitutes hardness in SDO problems, e.g., not having a strictly complementary solution [16], or a solution with a nonzero duality gap [22]. Wei and Wolkowicz [25] proposed a procedure to generate SDO problems without a strictly complementary solution. We build on these ideas to develop highly parameterized generators.

1.1. Contributions

This paper reviews and proposes several procedures to generate random LOPs, semidefinite optimization problems (SDOPs), and second-order cone optimization problems (SOCOPs) with a specified optimal solution, interior solution, and both of them. We also develop SDOP and SOCOP generators with specific maximally complementary solutions to predetermine the optimal partition.

Generating SDOPs and SOCOPs with a specific interior solution ensures that Strong Duality holds for the generated problems, and the set of optimal solutions will be bounded. Access to predefined interior solutions will enable researchers to analyze the performance of optimization algorithms, such as feasible Interior Point Methods (IPMs), with respect to various initial interior solutions.

Generating problems with known optimal solutions ensures that the generated problem has a bounded optimum and helps to analyze the algorithm concerning the characteristics of the optimal solution. These procedures will serve to further scholars' ability to examine their algorithms by altering different features of input data such as dimension, sparsity, condition number, solution size (which plays an essential role in the performance of Infeasible IPMs), and many others, besides predefined properties

of the optimal solution. Another possible application of the proposed procedures is the average-case complexity analysis of algorithms.

The rest of the paper is organized as follows. In Section 2, we give a brief review of LO theory before considering several LOP generators that can generate instances with specific optimal solutions, specific interior solutions, or both. We then develop similar generators for SDO and SOCO in Sections 3 and 4, respectively. A discussion on the implementation of the proposed instance generators is provided in Section 5, and Section 6 concludes the paper.

2. Linear Optimization

In this section, we provide a gentle review of Linear Optimization theory before presenting three different algorithms for randomly generating Linear Optimization test problems.

2.1. Linear Optimization Problems

In LOPs, we seek to minimize the inner product of two n-dimensional vectors

$$c^\top x = \sum_{i=1}^{n} c_i \cdot x_i,$$

for a constant vector $c \in \mathbb{R}^n$ and variable vector $x \in \mathbb{R}^n$. In this minimization, variable x must satisfy linear constraints of the form

$$Ax = b,$$

for a given matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$. Moreover, we require that x be elementwise nonnegative, which we denote by $x \geq 0$.

We are therefore interested in randomly generating LOPs of the form

$$z_{LO}^P = \min_x \left\{ c^\top x : Ax = b, x \geq 0 \right\}, \tag{LOP-P}$$

and refer to (LOP-P) as the *primal problem*. Given the primal problem (LOP-P), we are also interested in a second problem known as the *dual problem* of (LOP-P), which we write in standard form as follows,

$$z_{LO}^D = \max_{(y,s)} \left\{ b^\top y : A^\top y + s = c, s \geq 0, y \in \mathbb{R}^m \right\}, \tag{LOP-D}$$

where $s = c - A^\top y$ is the dual slack variable.

We say that x and (y, s) are *feasible solutions* whenever they satisfy the constraints of the primal and dual problems, respectively. The set of primal-dual feasible solutions is thus defined as

$$\mathcal{PD}_{LO} = \left\{ (x, y, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n : Ax = b, A^\top y + s = c, (x, s) \geq 0 \right\}.$$

Similarly, the set of all *feasible interior solutions* is given by

$$\mathcal{PD}_{LO}^{0} = \{(x, y, s) \in \mathcal{PD}_{LO} : (x, s) > 0\}.$$

A crucial property of linear optimization is *weak duality*; any (y, s) that is feasible for (LOP-D), provides a lower bound $b^\top y$ on the value of $c^\top x$ for any x feasible for (LOP-P), i.e.:

$$b^\top y \leq c^\top x,$$

for any $(x, y, s) \in \mathcal{PD}_{LO}$. Conversely, any x that is feasible for (LOP-P) provides an upper bound $c^\top x$ on $b^\top y$ for any y that is feasible for (LOP-D), and we refer to the nonnegative quantity $c^\top x - b^\top y = x^\top s$ as the *duality gap*.

Whenever $(x, y, s) \in \mathcal{PD}$ with $c^\top x = b^\top y$, or equivalently $x^\top s = 0$, then x is optimal for (LOP-P) and (y, s) is optimal for (LOP-D). In this case, *strong duality* holds for LOPs, i.e., if both the primal and dual problems have feasible solutions, then both have optimal solution with equal objective value. Under strong duality, all optimal solutions, if there exist any, belong to the set $\mathcal{PD}_{LO}^{*}$, defined as

$$\mathcal{PD}_{LO}^{*} = \left\{(x, y, s) \in \mathcal{PD}_{LO} : x^\top s = 0\right\}.$$

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Following Roos et al. [18], LOPs admit an optimal partition $\mathcal{N} \cup \mathcal{B} = [n]$, and $\mathcal{B} \cap \mathcal{N} = \emptyset$, where

$$\mathcal{B} = \{i : \exists (x^*, y^*, s^*) \in \mathcal{PD}_{LO}^{*} \text{ with } x_i^* > 0\},$$
$$\mathcal{N} = \{i : \exists (x^*, y^*, s^*) \in \mathcal{PD}_{LO}^{*} \text{ with } s_i^* > 0\}.$$

If $(x^*, y^*, s^*) \in \mathcal{PD}_{LO}^{*}$ with $x_i^* > 0$ for all $i \in \mathcal{B}$, and $s_i^* > 0$ for all $i \in \mathcal{N}$, then we have $x^* + s^* > 0$ and the optimal solution pair (x^*, y^*, s^*) is called strictly complementary. In this section, we use $(\mathcal{B}, \mathcal{N})$ to denote the optimal partition, and (B, N) the index set partition in the algorithms. After presenting each algorithm, we clarify when the predefined partition (B, N) is equal to the optimal partition $(\mathcal{B}, \mathcal{N})$.

2.2. Instance Generators for LOPs

In the rest of this section, we review three main generators which produce LO instances given either a predefined (or randomly chosen) interior solution, a predefined (or randomly chosen) optimal solution (maybe strictly complementary or not), or both. Each LOP generator allows the user to control the characteristics of parameters (A, b, c), including but not limited to their condition number, sparsity, and norm. Further, users can alter the optimal solution's features to examine their algorithm's performance.

In the following algorithms, the term "generate" should be interpreted freely. It may refer to generating the respective data randomly, or the connotation could be that the data is constructed with some specific purpose, e.g., to obtain matrices with some specific structure such as sparsity or conditioning.

2.2.1. LOPs with a Predefined Interior Solution

To study the performance of IPMs applied to LOPs, it is often helpful to have instances with specific interior solutions, and a common approach to generating LOPs with a desired interior solution is presented as Algorithm 1.

Algorithm 1 Generating a LOP with a specific interior solution

1: Choose dimensions $m < n$
2: Choose or generate (x^0, s^0) such that $x_i^0 > 0$ and $s_i^0 > 0$ for all $i \in [n]$
3: Generate $A \in \mathbb{R}^{m \times n}$
4: Generate $y^0 \in \mathbb{R}^m$
5: Calculate $b = Ax^0$ and $c = A^\top y^0 + s^0$
6: **Return** LOP (A, b, c) with interior solution (x^0, y^0, s^0)

Remark 1. Suppose we want the interior solution (x^0, s^0) to have a duality gap of ${x^0}^\top s^0 = n\mu$ for some scalar $\mu > 0$. Then, in Step 1 of Algorithm 1, we generate $x_i^0 > 0$ and calculate $s_i^0 = \frac{\mu}{x_i^0}$ for $i \in [n]$.

The above remark makes an observation relevant to IPMs, as in the context of IPMs, the constant μ, referred to as the central path parameter, plays a crucial role. IPMs begin with some initial interior solution $(x^0, s^0) \in \mathcal{PD}^0$ with

$$\frac{{x^0}^\top s^0}{n} = \mu^0 > 0,$$

and subsequently, reduce μ in each iteration as the algorithm progresses toward a solution to the LOP with desired complementarity gap. In line with our discussion on LO duality, it is easy to see that when $\mu \to 0$, we approach an optimal solution to the primal-dual pair (LOP-P)-(LOP-D).

Remark 2. Algorithm 1 facilitates the generation of a coefficient matrix A with any desired properties, e.g., sparsity, structure, or being ill-conditioned.

Remark 3. Several conditions are needed to generate a full row rank coefficient matrix A with probability one randomly [see e.g., 7].

2.2.2. LOPs with a Predefined Optimal Solution

A prevailing approach for generating LOPs with a known optimal solution is described in Algorithm 2.

Algorithm 2 Generating a LOP with a specific optimal solution

1: Choose dimensions $m < n$
2: Partition the index set $[n]$ to B and N with $B \cap N = \emptyset$ and $B \cup N = [n]$
3: Generate x^* such that $x_i^* > 0$ for $i \in B$ and $x_i^* = 0$ for $i \in N$
4: Generate s^* such that $s_i^* > 0$ for $i \in N$ and $s_i^* = 0$ for $i \in B$
5: Generate $A \in \mathbb{R}^{m \times n}$
6: Generate $y^* \in \mathbb{R}^m$
7: Calculate $b = Ax^*$ and $c = A^\top y^* + s^*$
8: **Return** LOP (A, b, c) with optimal solution (x^*, y^*, s^*)

Remark 4. Since the generated optimal solution (x^*, y^*, s^*) by Algorithm 2 is strictly complementary, the optimal partition $(\mathcal{B},\mathcal{N})$ is equal to (B, N).

Remark 5. Partition (B, N) may be generated randomly or to satisfy some desired properties, such as primal or dual degeneracy, or both, or having a unique optimal basis solution.

Remark 6. Let $A = [A_B \ A_N]$. If $|B| = m$ and A_B is nonsingular, then x^* and s^* yield the unique optimal basis solution.

Remark 7. If we modify Algorithm 2 by generating x^* such that $x_i^* \geq 0$ for $i \in B$ and $x_i^* = 0$ for $i \in N$, and s^* such that $s_i^* \geq 0$ for $i \in N$ and $s_i^* = 0$ for $i \in B$, then B and N do not necessarily give the optimal partition. While x^* and s^* are complementary solutions, they are not necessarily strictly complementary.

2.2.3. LOPs with Predefined Optimal and Interior Solutions

Charnes et al. [6] discuss procedures to generate problems with a specific optimal *or* interior solution. Here, we develop a novel procedure to generate a LOP with a specific optimal solution (x^*, y^*, s^*) *and* a specific interior solution (x^0, y^0, s^0), as presented in Algorithm 3. The general idea is first to use Algorithm 2 to generate a problem with optimal solution (x^*, y^*, s^*) before extending the problem by adding a variable and a constraint to make the interior point (x^0, y^0, s^0) feasible for the new problem. Using this scheme, we can produce LOPs for any general predefined optimal and interior solutions, where the only additional condition is

$$(x^0 - x^*)^\top (s^0 - s^*) = 0. \tag{1}$$

The condition stipulated by equation (1) is a natural property; for any feasible solution pairs, it follows that $(x^* - x^0) \in \text{Lin}^\perp(A)$ and $(s^* - s^0) \in \text{Lin}(A)$, where $\text{Lin}(A)$ denotes the lineality space of A. In other words, the difference of the predefined solutions $x^0 - x^*$ and $s^0 - s^*$ *must* be orthogonal, and steps 4 and 5 of Algorithm 3 ensure this property holds.

Algorithm 3 Generating LOP with specific optimal and interior solutions

1: Choose $m < n$ (The generated LOP has $m+1$ constraints and $n+1$ variables.)
2: Generate LOP $(\hat{A}, \hat{b}, \hat{c})$ with optimal solution $(\hat{x}, \hat{y}, \hat{s})$ and partition (B, N) using Algorithm 2
3: Generate $x^0, s^0 \in \mathbb{R}^n$ such $x^0, s^0 > 0$
4: Let $\delta = (x_B^0 - \hat{x}_B)^\top s_B^0 + (s_N^0 - \hat{s}_N)^\top x_N^0$, generate $x_{n+1}^0 > 0$ and $s_{n+1}^0 > (\frac{-\delta}{x_{n+1}^0})^+$
5: Calculate $\hat{s}_{n+1} = \frac{\delta}{x_{n+1}^0} + s_{n+1}^0$ and let $\hat{x}_{n+1} = 0$
6: Build $x^* = \begin{pmatrix} \hat{x}_B \\ 0 \\ 0 \end{pmatrix}$ and $x^0 = \begin{pmatrix} x_B^0 \\ x_N^0 \\ x_{n+1}^0 \end{pmatrix}$
7: Build $s^* = \begin{pmatrix} 0 \\ \hat{s}_N \\ \hat{s}_{n+1} \end{pmatrix}$ and $s^0 = \begin{pmatrix} s_B^0 \\ s_N^0 \\ s_{n+1}^0 \end{pmatrix}$
8: Generate $y^0 = \begin{pmatrix} y_{1:m}^0 \\ y_{m+1}^0 \end{pmatrix} \in \mathbb{R}^{m+1}$ randomly such that $y_{m+1}^0 \neq 0$
9: Build $y^* = \begin{pmatrix} \hat{y} \\ 0 \end{pmatrix}$
10: Calculate

$$\begin{aligned} \hat{a}_{n+1} &= \frac{1}{x_{n+1}^0}(\hat{A}_B(\hat{x}_B - x_B^0) - \hat{A}_N x_N^0) \\ d_B &= \frac{1}{y_{m+1}^0}(\hat{A}_B^\top(\hat{y} - y_{1:m}^0) - s_B^0) \\ d_N &= \frac{1}{y_{m+1}^0}(\hat{A}_N^\top(\hat{y} - y_{1:m}^0) + s_N^* - s_N^0) \\ d_{n+1} &= \frac{1}{x_{n+1}^0}(d_B^\top(\hat{x}_B - x_B^0) - d_N^\top x_N^0) \end{aligned}$$

11: Build $A_{(m+1)\times(n+1)} = \begin{pmatrix} \hat{A}_B & \hat{A}_N & \hat{a}_{n+1} \\ d_B^\top & d_N^\top & d_{n+1} \end{pmatrix}$
12: Calculate $b = \begin{pmatrix} \hat{b} \\ d_B^\top \hat{x}_B \end{pmatrix}$ and $c = \begin{pmatrix} \hat{c} \\ \hat{a}_{n+1}^\top \hat{y} + s_{n+1}^* \end{pmatrix}$
13: **Return** LOP (A, b, c) with optimal solution (x^*, y^*, s^*) and interior solution (x^0, y^0, s^0)

Theorem 2.2 asserts that the claimed properties of (x^0, y^0, s^0) and (x^*, y^*, s^*) are indeed correct. Before presenting and proving Theorem 2.2, we need to verify the orthogonality properties of the generated solution.

Lemma 2.1. *For (x^0, y^0, s^0) and (x^*, y^*, s^*) generated by Algorithm 3, then we have*

$$(x^0 - x^*)^\top (s^0 - s^*) = 0.$$

Proof. By construction, we have

$$\begin{aligned}(x^0 - x^*)^\top (s^0 - s^*) =& (x_B^0)^\top s_B^0 + (x_B^*)^\top s_B^* - (x_B^0)^\top s_B^* - (x_B^*)^\top s_B^0 (x_N^0)^\top s_N^0 \\ &+ (x_N^*)^\top s_N^* - (x_N^0)^\top s_N^* - (x_N^*)^\top s_N^0 (x_{n+1}^0)^\top s_{n+1}^0 \\ &+ (x_{n+1}^*)^\top s_N^* - (x_{n+1}^0)^\top s_{n+1}^* - (x_{n+1}^*)^\top s_{n+1}^0 \\ =& -\delta + (x_{n+1}^0)^\top s_{n+1}^0 - (x_{n+1}^0)^\top s_{n+1}^* = 0.\end{aligned}$$

The proof is complete. □

Using Lemma 2.1, the following theorem shows that the generated problem satisfies the desired properties.

Theorem 2.2. *Let (x^0, y^0, s^0) and (x^*, y^*, s^*) be generated by Algorithm 3. Then,*

$$x^* \geq 0,\ s^* \geq 0,\ x^0 > 0,\ s^0 > 0, \tag{2a}$$
$$(x^*)^\top s^* = 0, \tag{2b}$$
$$Ax^* = b, \tag{2c}$$
$$A^\top y^* + s^* = c, \tag{2d}$$
$$Ax^0 = b, \tag{2e}$$
$$A^\top y^0 + s^0 = c. \tag{2f}$$

That is, (x^0, y^0, s^0) and (x^*, y^*, s^*) and are, respectively, interior and optimal solutions of the generated LOP (A, b, c).

Proof. Observe that (2a) holds by construction. Equality (2b) refers to compelmentarity of (x^*, y^*, s^*), which holds due to the fact that

$$x^{*\top} s^* = \hat{x}_B^\top 0 + 0^\top \hat{s}_N + 0 s_{n+1}^* = 0.$$

To see that equation (2c) holds, i.e., the optimal solution satisfies primal feasibility, observe that

$$Ax^* = \begin{pmatrix} \hat{A}_B & \hat{A}_N & \hat{a}_{n+1} \\ d_B^\top & d_N^\top & d_{n+1} \end{pmatrix} \begin{pmatrix} \hat{x}_B \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \hat{A}_B \hat{x}_B \\ d_B^\top \hat{x}_B \end{pmatrix} = \begin{pmatrix} \hat{b} \\ d_B^\top \hat{x}_B \end{pmatrix} = b.$$

Similarly, dual feasibility is satisfied by the optimal solution, since

$$A^\top y^* + s^* = \begin{pmatrix} \hat{A}_B^\top & d_B \\ \hat{A}_N^\top & d_N \\ \hat{a}_{n+1}^\top & d_{n+1} \end{pmatrix} \begin{pmatrix} \hat{y} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \hat{s}_N \\ s_{n+1}^* \end{pmatrix} = \begin{pmatrix} \hat{c} \\ \hat{a}_{n+1}^\top \hat{y} + s_{n+1}^* \end{pmatrix} = c.$$

That is, equation (2d) holds.

The interior solution (x^0, y^0, s^0) is primal feasible since

$$\begin{aligned} Ax^0 &= \begin{pmatrix} \hat{A}_B & \hat{A}_N & \hat{a}_{n+1} \\ d_B^\top & d_N^\top & d_{n+1} \end{pmatrix} \begin{pmatrix} x_B^0 \\ x_N^0 \\ x_{n+1}^0 \end{pmatrix} \\ &= \begin{pmatrix} \hat{A}_B x_B^0 + \hat{A}_N x_N^0 + \hat{a}_{n+1} x_{n+1}^0 \\ d_B^\top x_B^0 + d_N^\top x_N^0 + d_{n+1} x_{n+1}^0 \end{pmatrix} \\ &= \begin{pmatrix} \hat{A}_B x_B^0 + \hat{A}_N x_N^0 + (\hat{A}_B(\hat{x}_B - x_B^0) - \hat{A}_N x_N^0) \\ d_B^\top x_B^0 + d_N^\top x_N^0 + (d_B^\top(\hat{x}_B - x_B^0) - d_N^\top x_N^0) \end{pmatrix} \\ &= \begin{pmatrix} \hat{A}_B \hat{x}_B \\ d_B^\top \hat{x}_B \end{pmatrix} = \begin{pmatrix} \hat{b} \\ d_B^\top \hat{x}_B \end{pmatrix} = b, \end{aligned}$$

which proves (2e). We can also certify the dual feasibility of the interior solution:

$$\begin{aligned} A^\top y^0 + s^0 &= \begin{pmatrix} \hat{A}_B^\top & d_B \\ \hat{A}_N^\top & d_N \\ \hat{a}_{n+1}^\top & d_{n+1} \end{pmatrix} \begin{pmatrix} y_{1:m}^0 \\ y_{m+1}^0 \end{pmatrix} + \begin{pmatrix} s_B^0 \\ s_N^0 \\ s_{n+1}^0 \end{pmatrix} = \begin{pmatrix} \hat{A}_B^\top y_{1:m}^0 + d_B y_{m+1}^0 + s_B^0 \\ \hat{A}_n^\top y_{1:m}^0 + d_N y_{m+1}^0 + s_N^0 \\ \hat{a}_{n+1}^\top y_{1:m}^0 + d_{n+1} y_{m+1}^0 + s_{n+1}^0 \end{pmatrix} \\ &= \begin{pmatrix} \hat{A}_B^\top y_{1:m}^0 + (\hat{A}_B^\top(\hat{y} - y_{1:m}^0) - s_B^0) + s_B^0 \\ \hat{A}_n^\top y_{1:m}^0 + (\hat{A}_N^\top(\hat{y} - y_{1:m}^0) + s_N^* - s_N^0) + s_N^0 \\ \alpha \end{pmatrix} = \begin{pmatrix} \hat{A}_B^\top \hat{y} \\ \hat{A}_N^\top \hat{y} + s_N^* \\ \alpha \end{pmatrix} \\ &= \begin{pmatrix} \hat{c} \\ \hat{a}_{n+1}^\top \hat{y} + s_{n+1}^* \end{pmatrix} = c, \end{aligned}$$

where $\alpha = \hat{a}_{n+1}^\top y_{1:m}^0 + d_{n+1} y_{m+1}^0 + s_{n+1}^0$.

Finally, to prove that equation (2f) holds as well, we still need to show that $\alpha = \hat{a}_{n+1}^\top \hat{y} + s_{n+1}^*$. By straightforward calculation, we have

$$\begin{aligned} \alpha &= \frac{1}{x_{n+1}^0}(\hat{A}_B(\hat{x}_B - x_B^0) - \hat{A}_N x_N^0)^\top y_{1:m}^0 + \frac{y_{m+1}^0}{x_{n+1}^0}(d_B^\top(\hat{x}_B - x_B^0) - d_N^\top x_N^0) + s_{n+1}^0 \\ &= \frac{1}{x_{n+1}^0}\Big((\hat{A}_B(\hat{x}_B - x_B^0) - \hat{A}_N x_N^0)^\top y_{1:m}^0 + (\hat{A}_B^\top(\hat{y} - y_{1:m}^0) - s_B^0)^\top(\hat{x}_B - x_B^0) - (\hat{A}_N^\top(\hat{y} - y_{1:m}^0) + s_N^* - s_N^0)^\top x_N^0\Big) + s_{n+1}^0 \\ &= \frac{1}{x_{n+1}^0}\Big({y_{1:m}^0}^\top \hat{A}_B \hat{x}_B - {y_{1:m}^0}^\top \hat{A}_B x_B^0 - {y_{1:m}^0}^\top \hat{A}_N x_N^0 + \hat{x}_B^\top \hat{A}_B^\top \hat{y} - \hat{x}_B^\top \hat{A}_B^\top y_{1:m}^0 - \hat{x}_B^\top s_B^0 - {x_B^0}^\top \hat{A}_B^\top \hat{y} + {x_B^0}^\top \hat{A}_B^\top y_{1:m}^0 + {x_B^0}^\top s_B^0 \\ &\quad - {x_N^0}^\top \hat{A}_N^\top \hat{y} + {x_N^0}^\top \hat{A}_N^\top y_{1:m}^0 - {x_N^0}^\top s_N^* + {x_N^0}^\top s_N^0\Big) + s_{n+1}^0 \\ &= \frac{1}{x_{n+1}^0}\Big(\hat{x}_B^\top \hat{A}_B^\top \hat{y} - {x_B^0}^\top \hat{A}_B^\top \hat{y} - {x_N^0}^\top \hat{A}_N^\top \hat{y} - \hat{x}_B^\top s_B^0 + {x_B^0}^\top s_B^0 - {x_N^0}^\top s_N^* + {x_N^0}^\top s_N^0\Big) + s_{n+1}^0 \\ &= \frac{(\hat{x}_B^\top \hat{A}_B^\top - {x_B^0}^\top \hat{A}_B^\top - {x_N^0}^\top \hat{A}_N^\top)}{x_{n+1}^0}\hat{y} + \frac{-\hat{x}_B^\top s_B^0 + {x_B^0}^\top s_B^0 - {x_N^0}^\top s_N^* + {x_N^0}^\top s_N^0 + s_{n+1}^0 x_{n+1}^0}{x_{n+1}^0} \\ &= \hat{a}_{n+1}^\top \hat{y} + \frac{(x^0 - x^*)^\top(s^0 - s^*) + s_{n+1}^* x_{n+1}^0}{x_{n+1}^0} = \hat{a}_{n+1}^\top \hat{y} + s_{n+1}^*. \end{aligned}$$

The proof is complete. □

Remark 8. Since the generated optimal solution (x^*, y^*, s^*) by Algorithm 3 is strictly complementary, the optimal partition $(\mathcal{B}, \mathcal{N})$ is equal to (B, N). If we modify Algorithm 3 such that $x_i^* \geq 0$ for $i \in B$ and $s_i^* \geq 0$ for $i \in N$, then B and N do not necessarily give the optimal partition. While x^* and s^* are complementary solutions, they are not necessarily strictly complementary.

Remark 9. We can simplify Algorithm 3 by setting

$$x_B^0 = \hat{x}_B, s_N^0 = \hat{s}_N, s_{n+1}^* = s_{n+1}^0, \text{and } y_{1:m}^0 = \hat{y}.$$

It is straightforward to verify that Condition 1 is satisfied for these choices. An even simpler case arises if we choose

$$x_N^0 = e, s_B^0 = e, \text{ and } y_{m+1}^0 = x_{n+1}^0 = s_{n+1}^0 = 1.$$

In the next section, we extend these problem generators to generate SDO problems.

3. Semidefinte Optimization

Now, we turn our attention to SDO. Just as in the previous section, we begin by reviewing the problem setting and important properties before presenting the instance generators for this class of optimization problems.

3.1. Semidefinte Optimization Problems

In *semidefinite optimization*, one seeks to minimize the inner product of two $n \times n$ symmetric matrices:

$$C \bullet X = \operatorname{tr}(CX) = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij},$$

for some symmetric constant matrix $C \in \mathcal{S}^n$ and matrix variable $X \in \mathcal{S}^n$. Note that $\mathcal{S}^n$ denotes the space of $n \times n$ symmetric matrices, and we write $\mathcal{S}_+^n$ ($\mathcal{S}_{++}^n$) to represent the cone of symmetric positive semidefinite (symmetric positive definite) matrices.

Similar to the LOP studied in the previous section, variable X must satisfy linear constraints of the form

$$A_i \bullet X = b_i, \quad \forall i \in [m],$$

where $A_1, \ldots, A_m \in \mathcal{S}^n$ are given symmetric matrices and $b \in \mathbb{R}^m$. Given that $C \bullet X$ is a linear function of X, stopping here would simply yield a LOP in which the variables are given by the entries of the matrix X. Rather, we add a nonlinear (albeit convex) constraint, which stipulates that X must be a positive semidefinite matrix, which we write $X \succeq 0$. More generally, the notation $U \succeq V$ indicates that $U - V$ is symmetric positive semidefinite, and is equivalent to stating $U - V \in \mathcal{S}_+^n$. Likewise, when the inequality is strict, i.e., $U \succ V$, it follows that $U - V \in \mathcal{S}_{++}^n$, so $U - V$ is symmetric positive definite. From the above discussion, it is straightforward to observe that SDO is a generalization of LO, in which we replace the element-wise nonnegativity constraint $x \geq 0$ found in (LOP-P) by a conic inequality with respect to the cone $\mathcal{S}_+^n$.

Accordingly, in this section we are interested in generating problems of the form

$$z_{SDO}^P = \inf_X \{C \bullet X : A_i \bullet X = b_i, \ \forall i \in [m], X \succeq 0\}, \tag{SDOP-P}$$

which has an associated dual problem

$$z^D_{SDO} = \sup_{(y,S)} \left\{ b^\top y : \sum_{i=1}^m y_i A_i + S = C,\ S \succeq 0, y \in \mathbb{R}^m \right\}, \tag{SDOP-D}$$

where $S = C - \sum_{i=1}^m y_i A_i$ is the slack matrix of the dual problem. Without loss of generality, we may assume that the matrices $A_1, \ldots, A_m$ are linearly independent.

If X and (y, S) satisfy the primal and dual constraints, respectively, we say that they are feasible solutions, denoting the feasible sets of (SDOP-P) and (SDOP-D) by:

$$\mathcal{P}_{SDO} = \{X \in \mathcal{S}^n : A_i \bullet X = b_i,\ i \in [m], X \succeq 0\}$$
$$\mathcal{D}_{SDO} = \left\{ (y, S) \in \mathbb{R}^m \times \mathcal{S}^n \ : \sum_{i=1}^m y_i A_i + S = C, S \succeq 0 \right\}.$$

Accordingly, the sets of feasible interior solutions are given by

$$\mathcal{P}^0_{SDO} = \{X \in \mathcal{P}_{SDO} : X \succ 0\},$$
$$\mathcal{D}^0_{SDO} = \{(y, S) \in \mathcal{P}_{SDO} : S \succ 0\}.$$

For ease of notation, we adopt the syntax $\mathcal{PD}_{SDO} = \mathcal{P}_{SDO} \times \mathcal{D}_{SDO}$ and $\mathcal{PD}^0_{SDO} = \mathcal{P}^0_{SDO} \times \mathcal{D}^0_{SDO}$.

Just as in the case of LO, when IPMs are applied to SDOPs, it is standard to assume the existence of a strictly feasible primal-dual pair X and (y, S) with $(X, S) \succ 0$. From the existence of a strictly feasible initial solution $(X^0, S^0) \succ 0$, it follows that the Interior Point Condition (IPC) is satisfied [9], guaranteeing that the primal and dual optimal sets

$$\mathcal{P}^*_{SDO} = \left\{X \in \mathcal{P}_{SDO} : C \bullet X = z^P_{SDO}\right\},$$
$$\mathcal{D}^*_{SDO} = \left\{(y, S) \in \mathcal{D}_{SDO} : b^\top y = z^D_{SDO}\right\},$$

are nonempty and bounded, that an optimal primal-dual pair with zero duality gap exists, i.e., strong duality holds. That is, for optimal solutions $(X^*, y^*, S^*) \in \mathcal{PD}^*_{SDO}$, where $\mathcal{PD}^*_{SDO} = \mathcal{P}^*_{SDO} \times \mathcal{D}^*_{SDO}$, we have

$$C \bullet X^* - b^\top y^* = X^* \bullet S^* = 0,$$

which implies $X^* S^* = S^* X^* = 0$ as X^* and S^* are symmetric positive semidefinite matrices.

3.2. Instance Generators for SDOPs

Similar to our work on LO, we propose three generators that produce SDO instances with a predefined interior solution, optimal solution, and both. Each generator is designed such that the user can control the characteristics of parameters such as condition number, sparsity, matrix structure, and size. Additionally, users can modify the features of optimal solutions to evaluate the performance of their algorithms.

3.2.1. SDOPs with a Predefined Interior Solution

To study the performance of IPMs applied to SDO, it is helpful to have instances with a specific interior solution. Generally, some users may need to generate problems with an interior solution to ensure that Strong Duality, i.e., zero duality gap at optimality, holds. Along this line, we adapt Algorithm 1 to generate SDO instances with known interior solutions, as given in Algorithm 4.

Algorithm 4 Generating SDO problems with a specific interior solution

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$
2: Generate (X^0, S^0) such that $X^0 \succ 0$ and $S^0 \succ 0$
3: Generate $A_i \in \mathcal{S}^n$ for $i \in [m]$
4: Generate $y^0 \in \mathbb{R}^m$
5: Calculate $b_i = A_i \bullet X^0$ for $i \in [m]$ and $C = \sum_{i=1}^m y_i^0 A_i + S^0$
6: **Return** SDOP $(A_1, \ldots, A_m, b, C)$ with interior solution (X^0, y^0, S^0)

Compared to Algorithm 1, the task of generating X^0 and S^0 in a general manner such that $X^0 S^0 = \mu I$ for $\mu > 0$ is more computationally involved; we would first have to generate $X^0 \succ 0$ randomly, and subsequently calculate S^0 as $S^0 = \mu (X^0)^{-1}$. However, we can easily generate X^0 and S^0 for a specified value of μ if we make additional assumptions regarding their structure (e.g., we can assume they are diagonal). We can also generate the matrices $A_1, \ldots, A_m$ to have desired properties such as sparsity, conditioning, or to satisfy some norm bound. Several approaches for generating random positive semidefinite are discussed in Appendix A.

3.2.2. SDOPs with a Predefined Block-diagonal Optimal Solution

Algorithm 5 can be seen as a generalization of Algorithm 2 to SDO problems, in which the generated optimal solution explicitly has a block-diagonal structure corresponding to the optimal partition. Before presenting the instance generator, we review the notation of the optimal partition in the context of SDO.

We are interested in problems whose optimal solution (X^*, y^*, S^*) exhibits zero duality gap, i.e., $X^* S^* = 0$. Thus, the spectral decomposition of an optimal pair X^* and S^* takes the form

$$X^* = Q \Sigma Q^\top \text{ and } S^* = Q \Lambda Q^\top,$$

where Q is orthonormal, and the matrices Σ and Λ are diagonal, containing eigenvalues of X^* and S^*, respectively. Letting $\sigma_i = \Sigma_{i,i}$ and $\lambda_i = \Lambda_{i,i}$, it follows that $X^* S^* = 0$ holds if and only if $\sigma_i \lambda_i = 0$ for all $i \in [n]$. A primal-dual optimal solution $(X^*, y^*, S^*) \in \mathcal{PD}^*_{SDO}$ is called maximally complementary if $X^* \in \text{ri}(\mathcal{P}^*_{SDO})$ and $(y^*, S^*) \in \text{ri}(\mathcal{D}^*_{SDO})$. A maximally complementary optimal solution (X^*, y^*, S^*) is called strictly complementary if $X^* + S^* \succ 0$. Let $\mathcal{B} \coloneqq \mathcal{R}(X^*)$ and $\mathcal{N} \coloneqq \mathcal{R}(S^*)$, where (X^*, y^*, S^*) is a maximally complementary optimal solution and $\mathcal{R}(.)$ denotes the range space. We define $n_\mathcal{B} \coloneqq \dim(\mathcal{B})$ and $n_\mathcal{N} \coloneqq \dim(\mathcal{N})$. Then, we have $\mathcal{R}(X) \subseteq B$ and $\mathcal{R}(S) \subseteq \mathcal{N}$ for all $(X, y, S) \in \mathcal{PD}^*_{SDO}$. By the complementarity condition, the subspaces $\mathcal{B}$ and $\mathcal{N}$ are orthogonal, and this implies that $n_\mathcal{B} + n_\mathcal{N} \leq n$, and in case of strict complementarity, $n_\mathcal{B} + n_\mathcal{N} = n$. Otherwise, a subspace $\mathcal{T}$ exists, which is the orthogonal complement to $\mathcal{B} + \mathcal{N}$. Similarly, we have $n_\mathcal{T} \coloneqq \dim(\mathcal{T})$, and so $n_\mathcal{B} + n_\mathcal{N} + n_\mathcal{T} = n$ [15]. The partition $(\mathcal{B}, \mathcal{N}, \mathcal{T})$ of $\mathbb{R}^n$ is called the optimal partition of an SDO problem.

In LOPs, we know that $\mathcal{T}$ is empty, but in general SDOPs $\mathcal{T}$ can be non-empty [9].

In Algorithm 5, we generate SDOPs with optimal solutions which exhibit a block-diagonal structure using a partition (B, N, T), which may be different from the optimal partition $(\mathcal{B}, \mathcal{N}, \mathcal{T})$ of the generated problem.

Algorithm 5 Generating SDO problems with a specific optimal solution

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$
2: Choose $n_B, n_N \in [n]$ where $n_B + n_N \leq n$
3: Generate positive definite matrix $X_B \in \mathcal{S}^{n_B}_{++}$
4: Generate positive definite matrix $S_N \in \mathcal{S}^{n_N}_{++}$
5: Build[1] $X^* = \begin{pmatrix} X_B & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and $S^* = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & S_N \end{pmatrix}$
6: Generate $A_i \in \mathcal{S}^n$ for $i \in [m]$
7: Generate $y^* \in \mathbb{R}^m$
8: Calculate $b_i = A_i \bullet X^*$ for $i \in [m]$ and $C = \sum_{i=1}^m y_i^* A_i + S^*$
9: **Return** SDOP $(A_1, \ldots, A_m, b, C)$ with optimal solution (X^*, y^*, S^*)

Remark 10. The sets (B, N, T) generated in Algorithm 5 are not necessarily the optimal partition $(\mathcal{B}, \mathcal{N}, \mathcal{T})$ for the generated SDO problem $(A_1, \ldots, A_m, b, C)$. In general, we only have

$$B \subseteq \mathcal{B}, N \subseteq \mathcal{N}, \text{ and } \mathcal{T} \subseteq T.$$

Remark 11. If an SDOP with a strictly complementary optimal solution is required, then we set $n_N = n - n_B$. In this case the optimal partition is predefined as $\mathcal{B} = B$, $\mathcal{N} = N$, and $\mathcal{T} = \emptyset$.

One can easily verify that the solution (X^*, y^*, S^*) generated by Algorithm 5 is feasible for the SDO problem $(A_1, \ldots, A_m, b, C)$, and optimal since $X^* S^* = 0$. In addition, matrices A_i and C can be generated in a way to exhibit a particular sparsity, condition number or norm, and we can also control primal and/or dual degeneracy.

3.2.3. SDOPs with Predefined Block-diagonal Optimal and Interior Solutions

By generating SDO problems with specific interior and optimal solutions, we can study the performance of various solution approaches. For example, one can analyze how efficiently feasible IPMs reduce the complementarity starting from a predefined interior solution to an optimal solution, or alternatively examine how robust performance is to the provided starting point or changes in the characteristics of the optimal solutions or partition. To accomplish this, we propose several algorithms in this paper providing an optimal solution or a maximally complementary solution.

This section is focused on the case in which the user is interested in predefining an interior solution and an optimal solution, which need not necessarily be maximally complementary. Accordingly, Algorithm 6 generalizes Algorithm 3 to SDO for the case in which the generated optimal solution has a block-diagonal structure. We similarly seek to generate an optimal solution (X^*, y^*, S^*) and interior solution (X^0, y^0, S^0) as generally as possible, but we need to impose some additional requirements. Letting $\mathcal{L} = \text{span}\{A_1, \ldots, A_m\}$, we have $X^0 - X^* \in \mathcal{L}^\perp$ and $S^0 - S^* \in \mathcal{L}$, and hence, the

generated solutions are required to satisfy the orthogonality condition

$$(X^0 - X^*) \bullet (S^0 - S^*) = 0. \tag{3}$$

In Algorithm 6, steps 7 and 8 are designed to ensure the generated solutions (X^*, y^*, S^*) and (X^0, y^0, S^0) indeed satisfy orthogonality.

Algorithm 6 Generating SDO problems with specific interior and optimal solutions

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$
2: Choose $n_B, n_N \in [n]$ where $n_B + n_N \leq n$
3: Generate SDO problem $(\hat{A}_1, \ldots, \hat{A}_m, \hat{b}, \hat{C})$ with optimal solution $(\hat{X}, \hat{y}, \hat{s})$ using Algorithm 5
4: Generate $X_B^0 \succ 0$, $X_T^0 \succ 0$,$X_N^0 \succ 0$,$X_{n+1}^0 > 0$ randomly
5: Build $X^*_{(n+1)\times(n+1)} = \begin{pmatrix} \hat{X} & 0 \\ 0 & 0 \end{pmatrix}$ and $X^0_{(n+1)\times(n+1)} = \begin{pmatrix} X_B^0 & 0 & 0 & 0 \\ 0 & X_T^0 & 0 & 0 \\ 0 & 0 & X_N^0 & 0 \\ 0 & 0 & 0 & X_{n+1}^0 \end{pmatrix}$
6: Generate $S_T^0 \succ 0$, $S_B^0 \succ 0$, $S_N^0 \succ 0$ randomly
7: Calculate $\delta = (X_B^0 - \hat{X}_B) \bullet S_B^0 + X_T^0 \bullet S_T^0 + X_N^0 \bullet (S_N^0 - \hat{S}_N)$
8: Generate $S_{n+1}^0 > (\frac{-\delta}{X_{n+1}^0})^+$ and calculate $\hat{S}_{n+1} = \frac{\delta}{X_{n+1}^0} + S_{n+1}^0$
9: Build $S^*_{(n+1)\times(n+1)} = \begin{pmatrix} \hat{S} & 0 \\ 0 & \hat{S}_{n+1} \end{pmatrix}$ and $S^0_{(n+1)\times(n+1)} = \begin{pmatrix} S_B^0 & 0 & 0 & 0 \\ 0 & S_T^0 & 0 & 0 \\ 0 & 0 & S_N^0 & 0 \\ 0 & 0 & 0 & S_{n+1}^0 \end{pmatrix}$
10: Generate $y^0 \in \mathbb{R}_{m+1}$ randomly such that $y_{m+1}^0 \neq 0$
11: Build $y^* = \begin{pmatrix} \hat{y} \\ 0 \end{pmatrix} \in \mathbb{R}_{m+1}$
12: Calculate $\alpha_i = \frac{1}{X_{n+1}^0}(\hat{A}_{i_B} \bullet (X_B - X_B^0)) - (\hat{A}_{i_N} \bullet X_N^0) - (\hat{A}_{i_T} \bullet X_T^0))$ for $i \in [m]$
13: Build $A_i = \begin{pmatrix} \hat{A}_i & 0 \\ 0 & \alpha_i \end{pmatrix}$ for $i \in [m]$
14: Build $A_{n+1} = \sum_{i=1}^{m} \frac{\hat{y}_i - y_i^0}{y_{m+1}^0} A_i + \frac{1}{y_{m+1}^0} \begin{pmatrix} -S_B^0 & 0 & 0 & 0 \\ 0 & -S_T^0 & 0 & 0 \\ 0 & 0 & \hat{S}_N - S_N^0 & 0 \\ 0 & 0 & 0 & \hat{S}_{n+1} - S_{n+1}^0 \end{pmatrix}$
15: Calculate $\theta = \hat{S}_{n+1} + \sum_{i=1}^{m} \hat{y}_i \alpha_i$ and build $C = \begin{pmatrix} \hat{C} & 0 \\ 0 & \theta \end{pmatrix}$
16: Calculate $\beta = A_{n+1} \bullet X^*$ and build $b = \begin{pmatrix} \hat{b} \\ \beta \end{pmatrix}$
17: **Return** SDOP $(A_1, \ldots, A_m, b, C)$ with optimal solution (X^*, y^*, S^*) and interior solution (X^0, y^0, S^0)

Before proving the correctness of Algorithm 6, the next result certifies the orthogonality properties of the generated solution.

Lemma 3.1. *For any (X^0, y^0, S^0) and (X^*, y^*, S^*) generated by Algorithm 6, we have*

$$(X^0 - X^*) \bullet (S^0 - S^*) = 0.$$

Proof. Similar to the proof of Lemma 2.1, it can be proved by substitution and using Steps 7 and 8. □

Using Lemma 3.1, the following theorem shows that the generated problem satisfies the desired properties.

Theorem 3.2. *Let* (X^0, y^0, S^0) *and* (X^*, y^*, S^*) *be solutions generated by Algorithm 6. Then,*

$$X^* \succeq 0,\ S^* \succeq 0,\ X^0 \succ 0,\ S^0 \succ 0, \tag{4a}$$

$$X^* \bullet S^* = 0, \tag{4b}$$

$$A_i \bullet X^* = b_i, \tag{4c}$$

$$\sum_{i=1}^{n} y_i^* A_i^\top + S^* = C, \tag{4d}$$

$$A_i \bullet X^0 = b_i, \tag{4e}$$

$$\sum_{i=1}^{n} y_i^0 A_i^\top + S^0 = C. \tag{4f}$$

Proof. Just as in the case of LO, all parts of Theorem 3.2 are easy to verify based on the steps of Algorithm 6, save for equation (4e) for $i = n + 1$. Following the proof of Theorem 2.2, the claimed result follows from the definition of α and equation (3). □

Similar to generating SDOPs with an optimal solution, we can also generate problems with both a specific strictly complementary optimal solution and a specific interior solution.

Remark 12. One special case is when $n_B + n_N = n$, $T = \emptyset$, and

$$X_B^0 = \hat{X}_B, X_N^0 = I, X_T^0 = I, S_B^0 = I, S_T^0 = I, S_N^0 = \hat{S}_N, X_{n+1}^0 = 1, S_{n+1}^0 = 1.$$

3.2.4. SDOPs with Predefined Optimal Solution (General Structure)

We are also interested in the situation where the desired optimal solution does not exhibit a block structure. Some methods can exploit the structural properties of the optimal solution, for example, when it exhibits a block-diagonal structure or is sparse. In order to generate an optimal solution that possesses certain desired qualities, we use the inverse process of eigenvalue decomposition. First, we generate diagonal matrices Σ and Λ, whose diagonal elements are the eigenvalues of X^* and S^*, respectively. Then, X^* and S^* can be calculated by masking these diagonal matrices using a randomly generated orthonormal matrix Q, and techniques for generating orthonormal matrices are discussed in Appendix B. The overall scheme is formalized below in Algorithm 7.

Algorithm 7 Generating SDO problems with a specific optimal solution

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$
2: Choose $n_B, n_N \in [n]$ where $n_B + n_N \leq n$
3: Generate $\sigma_i > 0$ for $i \in [n_B]$ and $\lambda_i > 0$ for $i \in [n_N]$
4: Generate orthonormal matrix Q
5: Build $X^* = Q \begin{pmatrix} \text{diag}(\sigma) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} Q^\top$ and $S^* = Q \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \text{diag}(\lambda) \end{pmatrix} Q^\top$
6: Generate $A_i \in \mathcal{S}^n$ randomly for $i \in [m]$
7: Generate $y^* \in \mathbb{R}^m$ randomly
8: Calculate $b_i = A_i \bullet X^*$ for $i \in [m]$ and $C = \sum_{i=1}^m y_i^* A_i + S^*$
9: **Return** SDOP $(A_1, \ldots, A_m, b, C)$ with optimal solution (X^*, y^*, S^*)

While Algorithm 7 generates a more general optimal solution than Algorithm 5, it is computationally more demanding due to several matrix multiplications and generating an orthonormal matrix. In Appendix B, some procedures to generate an orthogonal matrix are discussed. One can easily verify that (X^*, y^*, S^*) is optimal since $X^*S^* = Q\Sigma\Lambda Q^\top = 0$. However, similar to Remark 10, the generated optimal solution may not be the maximally complementary solution for the SDOP $(A_1, \ldots, A_m, b, C)$. Thus, the optimal partition $(\mathcal{B}, \mathcal{N}, \mathcal{T})$ of the generated SDOP may be different from (B, N, T) such that $\dim(\mathcal{B}) \geq n_B$ and $\dim(\mathcal{N}) \geq n_N$, i.e. the set of indices such that $\sigma_i = \lambda_i = 0$ may be bigger than the partition T. The next section discusses how we can generate SDOPs with predefined optimal partition.

3.2.5. SDOPs with a Predefined Maximally Complementary Solution (General Structure)

The SDOP generated by Algorithm 7 may have an optimal partition that differs from the input partition, since the specified optimal solution may not be maximally complementary. In this section, we develop a procedure to generate SDOPs with a specific optimal partition, and by extension, a specific maximally complementary solution.

Algorithm 8 Generating SDO problems with a specific maximally complementary solution

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$
2: Choose $n_B, n_N \in [n]$ where $n_B + n_N \leq n$
3: Generate $\sigma_i > 0$ for $i \in [n_B]$ and $\lambda_i > 0$ for $i \in [n_N]$
4: Generate orthonormal matrix Q
5: Build $X^* = Q \begin{pmatrix} \text{diag}(\sigma) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} Q^\top$ and $S^* = Q \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \text{diag}(\lambda) \end{pmatrix} Q^\top$
6: Generate $A_1 = Q\Gamma Q^\top$ such that $\Gamma = \text{diag}(\gamma)$ where $\gamma_B = 0$, $\gamma_T > 0$, and $\gamma_N \in \mathbb{R}^{n_N}$
7: Generate $A_i \in \mathcal{S}^n$ $i \in [m]$ such that $A_i Q_B$ are linearly independent for $i \in [m]$
8: Generate $y^* \in \mathbb{R}^m$
9: Calculate $b_i = A_i \bullet X^*$ for $i \in [m]$ and $C = \sum_{i=1}^m y_i^* A_i + S^*$
10: **Return** SDOP $(A_1, \ldots, A_m, b, C)$ with maximally complementary solution (X^*, y^*, S^*)

As we can see, there is less freedom in generating an SDOP using Algorithm 8 when

compared to Algorithm 7. This can be attributed to the fact that the matrix A_1 is specified to ensure that the specified optimal solution is maximally complementary, and we can not alter its characteristics directly. The next theorem proves the correctness of the generator.

Theorem 3.3. *For the generated problem $(A_1, \ldots, A_m, b, C)$ by Algorithm 8, the solution (X^*, y^*, S^*) is a maximally complementary optimal solution.*

Proof. The result follows from a proof by contradiction, which is adapted from [25]. Suppose that (X^*, y^*, S^*) is not maximally complementary, and $(\tilde{X}, \tilde{y}, \tilde{S})$ is a maximally complementary solution. Since $\tilde{X}S^* = 0$, we have

$$\mathcal{R}(X^*) \subseteq \mathcal{R}(\tilde{X}) \subseteq \mathcal{R}(S^*)^{\perp}.$$

Therefore, we can write

$$\tilde{X} = Q \begin{pmatrix} D_B & 0 & 0 \\ 0 & D_T & 0 \\ 0 & 0 & 0 \end{pmatrix} Q^{\top}.$$

Since both $\tilde{X}$ and X^* are feasible, it follows

$$\begin{aligned} 0 = A_1 \bullet (\tilde{X} - X^*) = A_1 \bullet Q \begin{pmatrix} D_B - \Lambda_B & 0 & 0 \\ 0 & D_T & 0 \\ 0 & 0 & 0 \end{pmatrix} Q^{\top} &= \Gamma \bullet \begin{pmatrix} D_B - \Lambda_B & 0 & 0 \\ 0 & D_T & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ &= \Gamma_T \bullet D_T. \end{aligned}$$

Given that $\Gamma_T > 0$, it follows that $D_T = 0$, which implies $\mathcal{R}(X^*) = \mathcal{R}(\tilde{X})$.

Next, we need to show that $\mathcal{R}(S^*) = \mathcal{R}(\tilde{S})$. Again, from dual feasibility, we have

$$\sum_{i=1}^{m} A_i(y_i^* - \tilde{y}_i) = -(S^* - \tilde{S}).$$

By the orthogonality of Q_B and $S^* - \tilde{S}$, one can observe

$$\sum_{i=1}^{m} A_i Q_B (y_i^* - \tilde{y}_i) = -(S^* - \tilde{S}) Q_B = 0.$$

Since the matrices $A_i Q_B$ are linearly independent for $i \in [m]$, it follows $y_i^* = \tilde{y}_i$ and $S^* = \tilde{S}$ and thus (X^*, y^*, S^*) is maximally complementary. Therefore, we have arrived at a contradiction, and the proof is complete. □

Corollary 3.4. *For the SDOP generated by Algorithm 8, the optimal partition $(\mathcal{B}, \mathcal{N}, \mathcal{T})$ is equal to (B, N, T).*

Remark 13. Let $\Pi_i = A_i Q_B$ for $i \in [m]$. One can generate matrix Π_i for $i \in \{2, \ldots, m\}$ so that the set of matrices Π_i for $i \in [m]$ are linearly independent, and calculate $A_i = \Pi_i Q_B^{\top}$. Consequently, the matrices $A_i Q_B$ will be linearly independent with probability 1.

The framework we have described is correct when $B \neq \emptyset$ and $N \neq \emptyset$. For the cases $B = \emptyset$ and/or $N = \emptyset$, one can construct a simple procedure, such as the one presented in Algorithm 9, to generate problems with predetermined optimal partition.

Algorithm 9 Generating SDO problems with a specific maximally complementary solution when $B = \emptyset$

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$
2: Choose $n_N \in [n]$
3: Generate $\lambda_i > 0$ for $i \in [n_N]$
4: Generate orthonormal matrix Q
5: Build $X^* = 0$ and $S^* = Q \begin{pmatrix} 0 & 0 \\ 0 & \text{diag}(\lambda) \end{pmatrix} Q^\top$
6: Generate $A_i = Q\Gamma_i Q^\top$ such that $\Gamma_i = \text{diag}(\gamma_i)$ where $\gamma_{Ti} = 0$, and $\gamma_{Ni} \in \mathbb{R}^{n_N}$ for $i \in [m]$
7: Generate $y^* \in \mathbb{R}^m$
8: Let $b = 0$ and calculate $C = \sum_{i=1}^m y_i^* A_i + S^*$
9: **Return** SDO problem $(A_1, \ldots, A_m, b, C)$ with optimal solution (X^*, y^*, S^*)

3.2.6. SDOPs with Predefined Optimal and Interior Solutions (General Structure)

We can also generalize Algorithm 7 to provide SDOPs with interior solutions, and the resulting scheme is presented in Algorithm 10. Here, both the generated optimal and interior solutions have general structure by using inverse of eigenvalue decomposition and at a high level the overall scheme can be viewed as a combination of Algorithms 6 and 7.

Algorithm 10 Generating SDO problems with specific interior and optimal solutions

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$ (the dimensions of generated SDOP: $m+1, n+1$)
2: Choose $n_B, n_N \in [n]$ where $n_B + n_N \leq n$
3: Define sets

$$B = \{1, \ldots, n_B\}, T = \{n_B + 1, \ldots, n - n_N\}, \text{ and } N = \{n - n_N + 1, \ldots, n\}$$

4: Generate $\sigma_i > 0$ for $i \in B$ and build $\Sigma_B = \text{diag}(\sigma)$
5: Generate $\lambda_i > 0$ for $i \in N$ and build $\Lambda_N = \text{diag}(\lambda)$
6: Generate orthonormal matrix $\hat{Q}_{n\times n}$
7: Build $\hat{X} = \hat{Q}\begin{pmatrix}\Sigma_B & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\hat{Q}^\top$ and $\hat{S} = \hat{Q}\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & \Lambda_N\end{pmatrix}\hat{Q}^\top$
8: Generate $\hat{y} \in \mathbb{R}^m$ and $\hat{A}_i \in \mathcal{S}^n$ for $i \in [m]$
9: Calculate $\hat{b}_i = \hat{A}_i \bullet X^*$ for $i \in [m]$ and $\hat{C} = \sum_{i=1}^m \hat{y}_i \hat{A}_i + \hat{S}$
10: Build $Q_{(n+1)\times(n+1)} = \begin{pmatrix}\hat{Q} & 0\\ 0 & 1\end{pmatrix}$
11: Generate positive diagonal matrix Σ_B^0, Σ_T^0, Σ_N^0, and number $\sigma_{n+1}^0 > 0$
12: Build $X^* = Q\begin{pmatrix}\Sigma_B & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}Q^\top$ and $X^0 = Q\begin{pmatrix}\Sigma_B^0 & 0 & 0 & 0\\ 0 & \Sigma_T^0 & 0 & 0\\ 0 & 0 & \Sigma_N^0 & 0\\ 0 & 0 & 0 & \sigma_{n+1}^0\end{pmatrix}Q^\top$
13: Generate positive diagonal matrix Λ_B^0, Λ_T^0, and Λ_N^0
14: Calculate $\delta = \sum_{i\in B}(\sigma_i - \sigma_i^0)\lambda_i^0 + \sum_{i\in T}\sigma_i^0\lambda_i^0 + \sum_{i\in N}\sigma_i^0(\lambda_i^0 - \lambda_i)$
15: Generate $\lambda_{n+1}^0 > (\frac{-\delta}{\sigma_{n+1}^0})^+$, and calculate $\lambda_{n+1} = \frac{\delta}{\sigma_{n+1}^0} + \lambda_{n+1}^0$
16: Build $S^* = Q\begin{pmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & \Lambda_N & 0\\ 0 & 0 & 0 & \lambda_{n+1}\end{pmatrix}Q^\top$ and $S^0 = Q\begin{pmatrix}\Lambda_B^0 & 0 & 0 & 0\\ 0 & \Lambda_T^0 & 0 & 0\\ 0 & 0 & \Lambda_N^0 & 0\\ 0 & 0 & 0 & \lambda_{n+1}^0\end{pmatrix}Q^\top$
17: Generate $y^0 \in \mathbb{R}^{m+1}$ randomly such that $y_{m+1}^0 \neq 0$ and build $y_{(m+1)}^* = \begin{pmatrix}\hat{y}\\ 0\end{pmatrix}$
18: Calculate $\alpha_i = \frac{1}{\sigma_{n+1}^0}\text{tr}\left(\hat{A}_i\hat{Q}\begin{pmatrix}\Sigma_B - \Sigma_B^0 & 0 & 0\\ 0 & -\Sigma_T^0 & 0\\ 0 & 0 & -\Sigma_N^0\end{pmatrix}\hat{Q}^\top\right)$ for $i \in [m]$
19: Build $A_i = \begin{pmatrix}\hat{A}_i & 0\\ 0 & \alpha_i\end{pmatrix}$ for $i \in [m]$
20: $A_{m+1} = \frac{1}{y_{m+1}^0}\left(\sum_{i=1}^m(\hat{y}_i - y_i^0)\hat{A}_i + Q\begin{pmatrix}-\Lambda_B^0 & 0 & 0 & 0\\ 0 & -\Lambda_T^0 & 0 & 0\\ 0 & 0 & \Lambda_T - \Lambda_T^0 & 0\\ 0 & 0 & 0 & \Lambda_{n+1} - \Lambda_{n+1}^0\end{pmatrix}Q^\top\right)$
21: Calculate $C = \begin{pmatrix}\hat{C} & 0\\ 0 & \sum_{i=1}^m \hat{y}_i\alpha_i + \lambda_{m+1}\end{pmatrix}$
22: Calculate $b_i = \hat{b}_i$ for $i \in [m]$ and $b_{m+1} = \text{tr}\,(A_{m+1}X^*)$
23: **Return** SDOP (A_i, b, C) with optimal solution (X^*, y^*, S^*) and interior solution (X^0, y^0, S^0)

For the generated solutions (X^0, y^0, S^0) and (X^*, y^*, S^*), Steps 14 and 15 ensure that

$$\sum_{i\in B}(\sigma_i^0 - \sigma_i)\lambda_i^0 + \sum_{i\in T}\sigma_i^0\lambda_i^0 + \sum_{i\in N}\sigma_i^0(\lambda_i^0 - \lambda_i) + \sigma_{n+1}^0(\lambda_{n+1}^0 - \lambda_{n+1}) = 0, \qquad (5)$$

Consequently, the orthogonality condition (3) is satisfied. Similar to the block-diagonal case, Theorem 3.5 establishes that the generated SDO problem and its optimal and interior solutions are correct.

Theorem 3.5. *Let (X^0, y^0, S^0) and (X^*, y^*, S^*) be solutions generated by Algorithm 10. Then,*

$$\begin{aligned} X^* \succeq 0,\ S^* \succeq 0,\ X^0 \succ 0,\ S^0 \succ 0,& \\ X^* \bullet S^* &= 0, \\ A_i \bullet X^* &= b_i, \\ \sum_{i=1}^{n} y_i^* A_i^\top + S^* &= C, \\ A_i \bullet X^0 &= b_i, \\ \sum_{i=1}^{n} y_i^0 A_i^\top + S^0 &= C. \end{aligned}$$

Proof. Analogous to the proof of Theorem 3.2 based on the steps of Algorithm 10 and Condition (3). □

3.2.7. SDOPs with Predefined Interior and Maximally Complementary Solutions (General Structure)

To have a predetermined optimal partition, we develop Algorithm 11 to generate SDOPs with specific interior and maximally complementary solutions as follows.

Algorithm 11 Generating SDO problems with specific interior and maximally complementary solutions

1: Choose dimensions m, n with $m < \frac{n(n+1)}{2}$ (the dimensions of generated SDOP: $m+1, n+1$)
2: Choose $n_B, n_N \in [n]$ where $n_B + n_N \leq n$
3: Define sets

$$B = \{1, \ldots, n_B\}, T = \{n_B + 1, \ldots, n - n_N\}, \text{ and } N = \{n - n_N + 1, \ldots, n\}$$

4: Generate $\sigma_i > 0$ for $i \in B$ and build $\Sigma_B = \text{diag}(\sigma)$
5: Generate $\lambda_i > 0$ for $i \in N$ and build $\Lambda_N = \text{diag}(\lambda)$
6: Generate orthonormal matrix $\hat{Q}_{n \times n}$
7: Build $\hat{X} = \hat{Q} \begin{pmatrix} \Sigma_B & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \hat{Q}^\top$ and $\hat{S} = \hat{Q} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Lambda_N \end{pmatrix} \hat{Q}^\top$
8: Generate $A_1 = Q \Gamma Q^\top$ such that $\Gamma = \text{diag}(\gamma)$ where $\gamma_B = 0$, $\gamma_T > 0$, and $\gamma_T \in \mathbb{R}^q$
9: Generate $\hat{y} \in \mathbb{R}^m$ and $\hat{A}_i \in \mathcal{S}^n$ for $i \in \{2, \ldots, m\}$
10: Calculate $\hat{b}_i = \text{tr}\,(\hat{A}_i X^*)$ for $i \in [m]$ and $\hat{C} = \sum_{i=1}^m \hat{y}_i \hat{A}_i + \hat{S}$
11: Build $Q_{(n+1)\times(n+1)} = \begin{pmatrix} \hat{Q} & 0 \\ 0 & 1 \end{pmatrix}$
12: Generate positive diagonal matrix Σ_B^0, Σ_T^0, Σ_N^0, and number $\sigma_{n+1}^0 > 0$
13: Build $X^* = Q \begin{pmatrix} \Sigma_B & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} Q^\top$ and $X^0 = Q \begin{pmatrix} \Sigma_B^0 & 0 & 0 & 0 \\ 0 & \Sigma_T^0 & 0 & 0 \\ 0 & 0 & \Sigma_N^0 & 0 \\ 0 & 0 & 0 & \sigma_{n+1}^0 \end{pmatrix} Q^\top$
14: Generate positive diagonal matrix Λ_B^0, Λ_T^0, and Λ_N^0
15: Calculate $\delta = \sum_{i \in B} (\sigma_i - \sigma_i^0)\lambda_i^0 + \sum_{i \in T} \sigma_i^0 \lambda_i^0 + \sum_{i \in N} \sigma_i^0 (\lambda_i^0 - \lambda_i)$
16: Generate $\lambda_{n+1}^0 > (\frac{-\delta}{\sigma_{n+1}^0})^+$, and calculate $\lambda_{n+1} = \frac{\delta}{\sigma_{n+1}^0} + \lambda_{n+1}^0$
17: Build $S^* = Q \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \Lambda_N & 0 \\ 0 & 0 & 0 & \lambda_{n+1} \end{pmatrix} Q^\top$ and $S^0 = Q \begin{pmatrix} \Lambda_B^0 & 0 & 0 & 0 \\ 0 & \Lambda_T^0 & 0 & 0 \\ 0 & 0 & \Lambda_N^0 & 0 \\ 0 & 0 & 0 & \lambda_{n+1}^0 \end{pmatrix} Q^\top$
18: Generate $y^0 \in \mathbb{R}^{m+1}$ such that $y_{m+1}^0 \neq 0$ and build $y_{(m+1)}^* = \begin{pmatrix} \hat{y} \\ 0 \end{pmatrix}$
19: Calculate $\alpha_i = \frac{1}{\sigma_{n+1}^0} \text{tr} \left(\hat{A}_i \hat{Q} \begin{pmatrix} \Sigma_B - \Sigma_B^0 & 0 & 0 \\ 0 & -\Sigma_T^0 & 0 \\ 0 & 0 & -\Sigma_N^0 \end{pmatrix} \hat{Q}^\top \right)$ for $i \in [m]$
20: Build $A_i = \begin{pmatrix} \hat{A}_i & 0 \\ 0 & \alpha_i \end{pmatrix}$ for $i \in [m]$
21: $A_{m+1} = \frac{1}{y_{m+1}^0} \left(\sum_{i=1}^m (\hat{y}_i - y_i^0) \hat{A}_i + Q \begin{pmatrix} -\Lambda_B^0 & 0 & 0 & 0 \\ 0 & -\Lambda_T^0 & 0 & 0 \\ 0 & 0 & \Lambda_T - \Lambda_T^0 & 0 \\ 0 & 0 & 0 & \Lambda_{n+1} - \Lambda_{n+1}^0 \end{pmatrix} Q^\top \right)$
22: Calculate $C = \begin{pmatrix} \hat{C} & 0 \\ 0 & \sum_{i=1}^m \hat{y}_i \alpha_i + \lambda_{m+1} \end{pmatrix}$
23: Calculate $b_i = \hat{b}_i$ for $i \in [m]$ and $b_{m+1} = \text{tr}\,(A_{m+1} X^*)$
24: **Return** SDOP $(A_1, \ldots, A_m, b, C)$ with optimal solution (X^*, y^*, S^*) and interior solution (X^0, y^0, S^0)

Theorem 3.6. *For Algorithm 11, solution (X^*, y^*, S^*) is the maximally complementary optimal solution of the generated problem $(A_1, \ldots, A_m, b, C)$.*

Proof. The proof closely follows the proof of Theorem 3.3; the only difference being that we expanded the matrices A_i by adding a row and column. For constructing matrix A_1, the added eigenvalue $\gamma_{n+1} = \alpha_1$ and $Q_{n+1} = (0, 0, 0, \ldots, 0, 1)^\top$ belong to partition N where we do not have any restriction. Thus, adapting the proof of Theorem 3.3 to this theorem is straightforward. □

Among all proposed SDOP generators, Algorithm 11 provides the most sophisticated SDOPs, with maximally complementary and interior solutions in a general manner and gives opportunities for altering characteristics of an optimal solution, optimal partition, matrices A_i, C, and vector b to study the performance of solution methods in a detailed and sophisticated analysis. However, this algorithm requires much more complicated computation than the other proposed generators.

4. Second-Order Cone Optimization

Before concluding, we adapt our techniques for LO and SDO to linear optimization problems over second order (or *Lorentz*) cones.

4.1. Second Order Cone Optimization Problems

A second-order cone is defined as follows

$$\left\{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n : x_1^2 - \sum_{i=2}^{n} x_i^2 \geq 0, x_1 \geq 0\right\}.$$

Observe that the above definition implies that $(x_1, x_2, \ldots, x_n)$ is a second-order cone if and only if the matrix

$$\begin{pmatrix} x_1 & x_{2:n}^\top \\ x_{2:n} & x_1 I_{n-1} \end{pmatrix}$$

is positive semidefinite, where $x_{2:n}^\top \equiv (x_2, x_3, \ldots, x_n)$ and I_{n-1} is the identity matrix of order $n-1$. Accordingly, a primal or dual second-order cone optimization problem (SOCOP) may be interpreted as a special case of SDO [19].

In SOCO problems, we seek to minimize a linear objective function over a feasible region which is defined by the intersection of an affine space and the Cartesian product of p second-order cones of dimension n_i, which is defined as

$$\mathbb{L}^n = \mathcal{L}^{n_1} \times \cdots \times \mathcal{L}^{n_p}, \quad n = \sum_{i=1}^{p} n_i,$$

where

$$\mathcal{L}^{n_i} = \{x^i = (x_1^i, \ldots, x_{n_i}^i)^\top \in \mathbb{R}^{n_i} : x_1^i \geq \|x_{2:n_i}^i\|\}, \quad i \in [p].$$

It is clear that LOPs are a special case of SOCOPs, where $n_i = 1$ for $i \in [p]$.

The primal and dual SOCO problems in standard form are represented as

$$z^P_{SOCO} = \inf_x \{c^\top x : Ax = b \ , \ x \in \mathbb{L}^n\},$$
$$z^D_{SOCO} = \sup_{(y,s)} \{b^\top y : A^\top y + s = c \ , \ s \in \mathbb{L}^n\},$$

where $b \in \mathbb{R}^m$, $A = (A_1, \ldots, A_p)$, $x = (x^1; \ldots; x^p)$, $s = (s^1; \ldots; s^p)$, and $c = (c^1; \ldots; c^p)$, in which $A_i \in \mathbb{R}^{m \times n_i}$, $s^i \in \mathbb{R}^{n_i}$, and $c^i \in \mathbb{R}^{n_i}$ for $i \in [p]$. The set of primal and dual feasible solutions is defined as

$$\mathcal{PD}_{SOCO} = \{(x, y, s) \in \mathbb{L}^n \times \mathbb{R}^m \times \mathbb{L}^n : Ax = b, A^\top y + s = c\}.$$

Let

$$\mathcal{L}^{n_i}_+ = \{x^i \in \mathcal{L}^{n_i} : x^i_1 > \|x^i_{2:n_i}\|\}, \quad i \in [p],$$

then we can define the set of primal and dual interior feasible solutions as

$$\mathcal{PD}^0_{SOCO} = \{(x, y, s) \in \mathbb{L}^n_+ \times \mathbb{R}^m \times \mathbb{L}^n_+ : Ax = b, A^\top y + s = c\}.$$

Just as in LO and SDO, it is standard practice to assume the existence of an interior feasible primal-dual solution. With the existence of a strictly feasible solution, it follows that the Interior Point Condition (IPC) is satisfied [14], guaranteeing that $z^P_{SOCO} = z^D_{SOCO}$ and the primal-dual optimal set

$$\mathcal{PD}^*_{SOCO} = \left\{(x, y, s) \in \mathcal{PD}_{SOCO} \ : \ c^\top x = z^P_{SOCO} = b^T y = z^D_{SOCO}\right\},$$

is nonempty and bounded. Therefore, there exists an optimal primal-dual pair with zero duality gap. That is, for optimal solutions x^* and (y^*, s^*), we have

$$x^* \circ s^* = (x^1 \circ s^1, \ldots, x^p \circ s^p) = 0, \tag{7}$$

where the Jordan product "$\circ$" is defined as

$$x^i \circ s^i = \begin{pmatrix} (x^i)^\top s^i \\ x^i_1 s^i_{2:n_i} + s^i_1 x^i_{2:n_i} \end{pmatrix}. \tag{8}$$

An optimal solution (x^*, y^*, s^*) is called maximally complementary if $x^* \in \mathrm{ri}(\mathcal{P}^*_{SOCO})$ and $(y^*; s^*) \in \mathrm{ri}(\mathcal{D}^*_{SOCO})$. Further, (x^*, y^*, s^*) is called strictly complementary if

$$x^* + s^* \in \mathbb{L}^n_+.$$

4.2. Instance Generators for SOCOPs

Motivated by our work on LOP and SDOP generators, we are further interested in applying these ideas to generate SOCO problems. Since SOCO can be interpreted as a special case of SDO, Sampourmahani et al. [19] studied mappings between SDOPs and SOCOPs and their optimal partitions. It is straightforward to develop SOCOP

generators using the proposed SDOP generators augmented with the appropriate mapping. However, that route is not efficient, and we alternatively propose several SOCOP generators without using their SDO representation.

4.2.1. SOCOPs with a Predefined Interior Solution

Generating SOCOPs with an interior solution also ensures that the problem has an optimal solution with zero duality gap. Algorithm 12 is a modification of Algorithms 1 and 4 to generate SOCOPs with specific interior solutions.

Algorithm 12 Generating SOCO problems with a specific interior solution

1: Choose dimensions $m < n$
2: Choose $n_1, \ldots, n_p$ such that $n = n_1 + \cdots + n_p$
3: Generate (x^0, s^0) such that $x^0 \in \mathbb{L}_+^n$ and $s^0 \in \mathbb{L}_+^n$
4: Generate $A \in \mathbb{R}^{m \times n}$
5: Generate $y^0 \in \mathbb{R}^m$
6: Calculate $b = Ax^0$ and $c = A^\top y^0 + s^0$
7: **Return** SOCOP (A, b, c) with interior solution (x^0, y^0, s^0)

To have an interior solution x^0, we must generate $(x^0)^i$ for $i = 1, \ldots, p$, such that $((x^0)_2^i, \ldots, (x^0)_{n_i}^i) \in \mathbb{R}^{n_i - 1}$ and $(x^0)_1^i > \|(x^0)_{2:n_i}^i\|$. One way to generate such a solution is to generate $(x^0)^i \in \mathbb{R}^{n_i}$, and update it using the rule

$$(x^0)_1^i = \|(x^0)_{2:n_i}^i\| + \|(x^0)_1^i\|.$$

Similar to LO, if the matrix A is generated randomly, then the probability of that all rows of A are linearly independent is one. In addition, the user can generate a desired matrix A with specific characteristics such as sparsity, condition number, and norm.

4.2.2. SOCOPs with a Predefined Optimal Solution

For SOCOPs, the optimal partition is a bit more complicated than for LO and SDO. The index set $[p]$ is partitioned to sets $(\mathcal{B}, \mathcal{N}, \mathcal{R}, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ defined as

$$\begin{aligned}
\mathcal{B} &\coloneqq \{i : x_1^i > \|x_{2:n_i}^i\|_2, \text{ for some } x \in \mathcal{P}_{SOCO}^*\},\\
\mathcal{N} &\coloneqq \{i : x_1^i > \|x_{2:n_i}^i\|_2, \text{ for some } x \in \mathcal{D}_{SOCO}^*\},\\
\mathcal{R} &\coloneqq \{i : x_1^i = \|x_{2:n_i}^i\|_2 > 0, x_1^i = \|x_{2:n_i}^i\|_2 > 0, \text{ for some } (x, y, x) \in \mathcal{P}_{SOCO}^* \times \mathcal{D}_{SOCO}^*\},\\
\mathcal{T}_1 &\coloneqq \{i : x^i = x^i = 0, \text{ for all } (x, y, x) \in \mathcal{P}_{SOCO}^* \times \mathcal{D}_{SOCO}^*\},\\
\mathcal{T}_2 &\coloneqq \{i : x^i = 0, \text{ for all } (y, x) \in \mathcal{D}_{SOCO}^*, x_1^i = \|x_{2:n_i}^i\|_2 > 0, \text{ for some } x \in \mathcal{P}_{SOCO}^*\},\\
\mathcal{T}_3 &\coloneqq \{i : x^i = 0, \text{ for all } x \in \mathcal{P}_{SOCO}^*, x_1^i = \|x_{2:n_i}^i\|_2 > 0, \text{ for some } (y, x) \in \mathcal{D}_{SOCO}^*\}.
\end{aligned}$$

For further discussion regarding the optimal partition in SOCOPs, we refer the reader to [23]. From here, we can develop Algorithm 13 which is a generalization of Algorithm 2 for generating random SOCOPs with specific optimal solutions.

Algorithm 13 Generating SOCO problems with a specific optimal solution

1: Choose dimensions $m < n$
2: Choose $n_1, \ldots, n_p$ such that $n = n_1 + \cdots + n_p$
3: Partition the index set $[p]$ to (B, N, R, T_1, T_2, T_3)
4: For $i \in B$, $(s^*)^i = 0$ and generate $(x^*)^i \in \mathbb{R}^{n_i}$ such $(x^*)^i_1 > \|(x^*)^i_{2:n_i}\|$
5: For $i \in N$, $(x^*)^i = 0$ and generate $(s^*)^i \in \mathbb{R}^{n_i}$ such $(s^*)^i_1 > \|(s^*)^i_{2:n_i}\|$
6: For $i \in T_1$, $(s^*)^i = 0$ and $(x^*)^i = 0$
7: For $i \in T_2$, $(s^*)^i = 0$ and generate $(x^*)^i \in \mathbb{R}^{n_i}$ such $(x^*)^i_1 = \|(x^*)^i_{2:n_i}\| > 0$
8: For $i \in T_3$, $(x^*)^i = 0$ and generate $(s^*)^i \in \mathbb{R}^{n_i}$ such $(s^*)^i_1 = \|(s^*)^i_{2:n_i}\| > 0$
9: For $i \in R$, generate $(x^*)^i_{2:n_i} \in \mathbb{R}^{n_i - 1}$ and $\delta \in \mathbb{R}$ and build

$$(x^*)^i = \begin{pmatrix} \|(x^*)^i_{2:n_i}\| \\ (x^*)^i_{2:n_i} \end{pmatrix}, \text{ and } (s^*)^i = \delta \begin{pmatrix} \|(x^*)^i_{2:n_i}\| \\ -(x^*)^i_{2:n_i} \end{pmatrix}$$

10: Generate $y^* \in \mathbb{R}^m$
11: Generate $A \in \mathbb{R}^{m \times n}$
12: Calculate $b = Ax^*$ and $c = A^\top y^* + s^*$
13: **Return** SOCOP (A, b, c) with optimal solution (x^*, y^*, s^*)

Remark 14. Algorithm 13 provides a SOCOP with an optimal solution, and that optimal solution may not be maximally complementary. Thus, the optimal partition $(\mathcal{B}, \mathcal{N}, \mathcal{R}, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ of the generated problem may differ from (B, N, R, T_1, T_2, T_3), and we only have

$$B \subseteq \mathcal{B},\ N \subseteq \mathcal{N},\ R \subseteq \mathcal{R},\ \mathcal{T}_1 \subseteq T_1,\ \mathcal{T}_2 \subseteq T_2 \cap T_1, \text{ and } \mathcal{T}_3 \subseteq T_3 \cap T_1.$$

Similar to LOPs and SDOPs generators, one can generate A random in a way to have specific characteristics. The norm and properties of (x^*, y^*, s^*) are controllable directly. Also, the norm of (b, c) can be predetermined by scaling (x^*, y^*, s^*) appropriately and carefully, since determining the norm of all parameters and the optimal solution simultaneously is possible if the equations in line 11 of Algorithm 13 hold. It is easy to see that (x^*, y^*, s^*) is optimal since $x^* \circ s^* = 0$ and it is feasible by construction. In the next section, we discuss how to generate a SOCOP with a maximally complementary solution.

4.2.3. SOCOPs with a Predefined Maximally Complementary Solution

Since the optimal partition can affect the performance of algorithms to solve SOCOPs similar to SDO, we are interested in generating problems with predetermined optimal partitions. To do so, we adapt our instance generator for SDOPs with maximally complementary solution to SOCO in Algorithm 14. Let $A^p_{i,j}$ be the element in row i and column j of part (columns) of A that correspond to cone p. We also use the superscript to show the partition, e.g., A^B denotes the columns of A which correspond to partition B.

Algorithm 14 Generating SOCOPs with a specific maximally complementary solution

1: Choose dimensions $m < n$
2: Choose $n_1, \ldots, n_p$ such that $n = n_1 + \cdots + n_p$
3: Partition the index set $[p]$ to (B, N, R, T_1, T_2, T_3) such that

$$|T_2| + 1 < m \leq |B| + |R| + |T_2|$$

4: For $i \in B$, $(s^*)^i = 0$ and generate $(x^*)^i \in \mathbb{R}^{n_i}$ such $(x^*)^i_1 > \|(x^*)^i_{2:n_i}\|$
5: For $i \in N$, $(x^*)^i = 0$ and generate $(s^*)^i \in \mathbb{R}^{n_i}$ such $(s^*)^i_1 > \|(s^*)^i_{2:n_i}\|$
6: For $i \in T_1$, $(s^*)^i = 0$ and $(x^*)^i = 0$
7: For $i \in T_2$, $(s^*)^i = 0$ and generate $(x^*)^i \in \mathbb{R}^{n_i}$ such $(x^*)^i_1 = \|(x^*)^i_{2:n_i}\| > 0$
8: For $i \in T_3$, $(x^*)^i = 0$ and generate $(s^*)^i \in \mathbb{R}^{n_i}$ such $(s^*)^i_1 = \|(s^*)^i_{2:n_i}\| > 0$
9: For $i \in R$, generate $(x^*)^i_{2:n_i} \in \mathbb{R}^{n_i - 1}$ and $\delta \in \mathbb{R}$ and build

$$(x^*)^i = \begin{pmatrix} \|(x^*)^i_{2:n_i}\| \\ (x^*)^i_{2:n_i} \end{pmatrix}, \text{ and } (s^*)^i = \delta \begin{pmatrix} \|(x^*)^i_{2:n_i}\| \\ -(x^*)^i_{2:n_i} \end{pmatrix}$$

10: Generate $y^* \in \mathbb{R}^m$
11: Generate $A \in \mathbb{R}^{m \times n}$ such that

- First row:

$$\begin{aligned} A^p_{1,1} > 0, A^p_{1,j} = 0 \text{ for } j = 2, \ldots, n_p, \text{ and } p \in T_1 \cup T_3 \\ A^p_{1,j} = 0 \text{ for } j = 1, \ldots, n_p, \text{ and } p \in B \cup R \cup T_2 \\ A^p_{1,j} \in \mathbb{R} \text{ for } j = 1, \ldots, n_p, \text{ and } p \in N \end{aligned}$$

- Row 2 to $|T_2| + 1$:

$$A^p_{p,1:n_p} = \begin{bmatrix} -1 & \frac{(x^{*k}_{2:n_p})^\top}{\|x^{*k}_{2:n_p}\|} \end{bmatrix}, A^p_{k,1:n_k} = 0 \text{ for all } k \neq q, \text{ and } p \in T_2$$

- The other rows should be generated such that $\text{rank}([A^B, A^R, A^{T_2}]) = m$.

12: Calculate $b = Ax^*$ and $c = A^\top y^* + s^*$
13: **Return** SOCOP (A, b, c) with maximally complementary solution (x^*, y^*, s^*)

Compared to Algorithm 13, Algorithm 14 imposes more restrictions on how A is generated.

Theorem 4.1. *For any SOCOP (A, b, c) generated by Algorithm 14, the generated optimal solution (x^*, y^*, s^*) is maximally complementary.*

Proof. One can verify that $b_1 = 0$, and the first row of A enforces any optimal solution $\bar{x}$ to satisfy

$$\bar{x}^p = 0 \text{ for all } p \in T_1 \cup T_3.$$

From constraint 2 to $|T_2| + 1$, we add a constraint for each cone p in partition T_2 in which coefficients are zero for all variables except for variables in cone p. Since the corresponding right-hand side is zero and the coefficients are the normal vector

to the cone p at the point x^*, all feasible solutions must lie on the ray which is on the boundary of the cone p and passing through the point x^*. Thus, for any optimal solution $\bar{x}$, we have

$$\bar{x}_1^p = \|x_{2:n_p}^p\| \text{ for all } p \in T_2.$$

Up to this point, we have shown that $x^* \in ri(\mathcal{P}^*)$, and the last part of the proof is to establish that the dual problem has a unique optimal solution (y^*, s^*). To prove it, let assume that it has another optimal solution $(\bar{y}, \bar{s})$, and $X^* = \text{diag}(x^*)$. Then, we have

$$X^* A^\top (\bar{y} - y^*) = -X^* (\bar{s} - s^*) = 0.$$

Since A generated in a way that the rank of $X^* A^\top$ is m, we have $\bar{y} = y^*$. We can conclude that (x^*, y^*, s^*) is a maximal complementary solution for the generated problem. □

Corollary 4.2. *For any SOCOP generated by Algorithm 14, the optimal partition* $(\mathcal{B}, \mathcal{N}, \mathcal{R}, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$ *is equal to* (B, N, R, T_1, T_2, T_3).

As expected, generating SOCOPs with predefined optimal partitions restricts on how matrix A is generated. However, some components of A are not restricted, and enable the user to control the properties of A. This is discussed next.

4.2.4. SOCOPs with Optimal and Interior Solutions

We can extend Algorithm 13 to provide both specific interior and optimal solutions by adding one row and column to the matrix A. We aim to generate optimal and interior solutions in a general manner, but we need to enforce the orthogonality condition:

$$(x^0 - x^*)^\top (s^0 - s^*) = 0. \tag{9}$$

Note that this is a natural requirement; similar to LOPs, we have $(x^* - x^0) \in \text{Lin}^\perp(A)$ and $(s^* - s^0) \in \text{Lin}(A)$.

Algorithm 15 Generating SOCOPs with specific interior and optimal solutions

1: Choose dimensions $m < n$ (the dimension of generated SOCOP: $m+1, n+1$)
2: Generate $(\hat{A}, \hat{b}, \hat{c})$ with optimal solution $(\hat{x}, \hat{y}, \hat{s})$ by Algorithm 13 (dimension: $p \times m$)
3: Generate $x^0 \in \mathbb{L}_+^{n_1,\ldots,n_p+1}$ and let $\hat{x}^0 = ((x^0)^1; \ldots; (x^0)^p_{1:n_p})$
4: Build $x^* \in \mathbb{L}_+^{n_1,\ldots,n_p+1}$ such that

$$(x^*)^{1:p-1} = (\hat{x})^{1:p-1},\ (x^*)^p_{1:n_p} = (\hat{x})^p_{1:n_p}, \text{ and } (x^*)^p_{n_p+1} = 0$$

5: Calculate vector $\alpha = \frac{1}{(x^0)^p_{n_p+1}} \hat{A}(\hat{x} - \hat{x}^0)$
6: Build $\tilde{A} = (\hat{A}, \alpha)$ (concatenation of a column to matrix)
7: Generate $y^0 \in \mathbb{R}^{m+1}$ such $y^0_{m+1} \neq 0$, and build $y^* = (\hat{y}, 0)^\top \in \mathbb{R}^{m+1}$
8: Generate $\hat{s}^0 \in \mathbb{L}_+^{n_1,\ldots,n_p}$
9: Let $\delta = (\hat{x}^0 - \hat{x})^\top (\hat{s}^0 - \hat{s})$, and generate $(s^0)^p_{n_p+1} > (\frac{-\delta}{(x^0)^p_{n_p+1}})^+$
10: Calculate $\hat{s}^p_{n_p+1} = \frac{\delta}{(x^0)^p_{n_p+1}} + (s^0)^p_{n_p+1}$
11: Build $s^* = (\hat{s}, \hat{s}^p_{n_p+1})^\top$, $s^0 = (\hat{s}^0, (s^0)^p_{n_p+1})^\top$;
12: Calculate vector $\beta = \frac{1}{y^0_{m+1}}(\tilde{A}^\top(\hat{y} - (y^0)_{1:m}) + s^* - s^0)$
13: Build $A = \begin{pmatrix} \tilde{A} \\ \beta^\top \end{pmatrix}$ (concatenation of a row to matrix)
14: Calculate $b = Ax^*$ and $c = A^\top y^* + s^*$
15: **Return** SOCOP (A, b, c) with interior solution (x^0, y^0, s^0) and optimal solution (x^*, y^*, s^*)

Theorem 4.4 shows that the claimed properties of (x^0, y^0, s^0) and (x^*, y^*, s^*) are indeed correct. Before presenting Theorem 4.4, we need to verify the orthogonality properties of the generated solution.

Lemma 4.3. *For any (x^0, y^0, s^0) and (x^*, y^*, s^*) generated by Algorithm 15, we have*

$$(x^0 - x^*)^\top (s^0 - s^*) = 0.$$

Proof. Similar to the proof of Lemma 2.1, Steps 9 and 11 of Algorithm 15 ensure that the orthogonality condition holds. □

Using Lemma 4.3, the following theorem shows that the generated problem satisfies the desired properties.

Theorem 4.4. *Let (x^0, y^0, s^0) and (x^*, y^*, s^*) be generated by Algorithm 15. Then,*

$$\begin{aligned} x^* \circ s^* &= 0, \\ Ax^* &= b, \\ A^\top y^* + s^* &= c, \\ Ax^0 &= b, \\ A^\top y^0 + s^0 &= c, \\ x^* \in \mathbb{L}^{n+1},\ s^* \in \mathbb{L}^{n+1},\ x^0 &\in \mathbb{L}_+^{n+1},\ s^0 \in \mathbb{L}_+^{n+1}. \end{aligned}$$

That is, (x^0, y^0, s^0) and (x^*, y^*, s^*) are interior and optimal solutions, respectively, for the generated SOCOP (A, b, c).

Proof. The proof is similar to the proof of Theorem 2.2. □

Compared to the SDOP generators, the SOCOP generators are computationally simpler since they do not require generating random orthonormal or positive semidefinite matrices. Let t_R be the amount of arithmetic operations required to generate a number randomly. To generate orthonormal or positive semidefinite matrices, we need to use a decomposition method, which requires $\mathcal{O}(n^3)$ arithmetic operations as discussed in the appendix. In the general case, the LOP and SOCOP generators require $\mathcal{O}(n^2 t_r)$ arithmetic operations, while the SDO generators require $\mathcal{O}(n^3 t_r)$ arithmetic operations. It should be mentioned that if we want to generate a random matrix A in LOPs and SOCOPs with specific condition numbers, then we need to use decomposition methods and the complexity of the LOP and SOCOP generators increases to $\mathcal{O}(n^3 t_r)$ arithmetic operations.

4.2.5. SOCOPs with Predefined Interior and Maximally Complementary Solutions

To generate a SOCOPs with both interior and maximally complementary solutions, we can use Algorithm 15 and in its first step, use Algorithm 14 which provide a SOCOP with maximally complementary solution. The only difference is that we should choose the partition such that the last cone is in the partition N. By this modification, the added column in Step 6 of Algorithm 15 will be in partition N, which satisfies all the restrictions needed to keep (x^*, y^*, s^*) maximally complementary. In this way, we can generate a SOCOP with interior solution and predetermined optimal partition.

5. Implementation

All mentioned generators are implemented in a `python` package, which is available in open source at `https://github.com/qcol-lu/qipm`. This package gives the option of prescribing the norm of vectors, condition numbers, and sparsity of the matrices. In addition, several versions of interior point methods, such as feasible/infeasible, exact/inexact, and long-step/short-step/predictor-corrector, are implemented and available for the experiment. There is also an option to choose the solver of the Newton system. One may choose classical or quantum linear system algorithms.

6. Conclusion

We develop and implement several random instance generators for LO, SDO, and SOCO with specific optimal and/or interior solutions. Because of high level of controllability, these generators enable users to analyze different features of the problem, such as sparsity and condition number, to study the performance of different algorithms smartly. In addition, we proposed SDOP and SOCOP generators with predefined optimal partition, which can be used to generate computationally challenging instances.

The proposed generators can also be used to study the average performance of algorithms for solving LO, SDO, and SOCO problems with different probability distributions for input data, optimal and interior solutions. Future research directions include

expanding the construction of these generators for other classes of conic, polynomial, and nonlinear optimization problems.

A useful direction for extending the proposed generators is to generate hard problems which are challenging for algorithms and solvers, e.g., LOPs which are primal or dual unbounded. For SDO and SOCO, it is worth exploring to develop generators which produce instances that have zero-duality gap, but with an optimal solution that is not attainable or instances with non-zero duality gap.

7. Acknowledgement

This work is supported by Defense Advanced Research Projects Agency as part of the project W911NF2010022: *The Quantum Computing Revolution and Optimization: Challenges and Opportunities*.

References

[1] J.L. Arthur and J.O. Frendewey, *GENGUB: A generator for linear programs with generalized upper bound constraints*, Computers & Operations Research 20 (1993), pp. 565–573.

[2] S. Bowly, K. Smith-Miles, D. Baatar, and H. Mittelmann, *Generation techniques for linear programming instances with controllable properties*, Mathematical Programming Computation 12 (2020), pp. 389–415.

[3] P.H. Calamai, L.N. Vicente, and J.J. Júdice, *A new technique for generating quadratic programming test problems*, Mathematical Programming 61 (1993), pp. 215–231.

[4] E. Castillo, R.E. Pruneda, and M. Esquivel IV, *Automatic generation of linear programming problems for computer aided instruction*, International Journal of Mathematical Education in Science and Technology 32 (2001), pp. 209–232.

[5] S. Chakraborty and P.P. Choudhury, *A statistical analysis of an algorithm's complexity*, Applied Mathematics Letters 13 (2000), pp. 121–126.

[6] A. Charnes, W.M. Raike, J.D. Stutz, and A.S. Walters, *On generation of test problems for linear programming codes*, Communications of the ACM 17 (1974), pp. 583–586.

[7] A. Coja-Oghlan, P. Gao, M. Hahn-Klimroth, J. Lee, N. Müller, and M. Rolvien, *The full rank condition for sparse random matrices*, arXiv preprint arXiv:2112.14090 (2021).

[8] C. Cotta and P. Moscato, *A mixed evolutionary-statistical analysis of an algorithm's complexity*, Applied Mathematics Letters 16 (2003), pp. 41–47.

[9] E. De Klerk, *Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications*, Vol. 65, Springer Science & Business Media, 2006.

[10] D.M. Gay, *Electronic mail distribution of linear programming test problems*, Mathematical Programming Society COAL Newsletter 13 (1985), pp. 10–12.

[11] J.N. Hooker, *Needed: An empirical science of algorithms*, Operations Research 42 (1994), pp. 201–212.

[12] D. Klingman, A. Napier, and J. Stutz, *NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems*, Management Science 20 (1974), pp. 814–821.

[13] F. Mezzadri, *How to generate random matrices from the classical compact groups*, arXiv preprint math-ph/0609050 (2006).

[14] A. Mohammad-Nezhad, *Conic Optimization: Optimal Partition, Parametric, and Stability Analysis*, Ph.D. diss., Lehigh University, 2019.

[15] A. Mohammad-Nezhad and T. Terlaky, *On the identification of the optimal partition for semidefinite optimization*, INFOR: Information Systems and Operational Research 58 (2020), pp. 225–263.

[16] G. Pataki, *Bad semidefinite programs: they all look the same*, SIAM Journal on Optimization 27 (2017), pp. 146–172.
[17] M.G. Pilcher and R.L. Rardin, *Partial polyhedral description and generation of discrete optimization problems with known optima*, Naval Research Logistics (NRL) 39 (1992), pp. 839–858.
[18] C. Roos, T. Terlaky, and J.P. Vial, *Theory and Algorithms for Linear Optimization: An Interior Point Approach*, Wiley Chichester, 1997.
[19] P. Sampourmahani, M. Mohammadisiahroudi, and T. Terlaky, *On semidefinite representations of second-order conic optimization problems*, arXiv preprint arXiv:2301.12007 (2023).
[20] K. Smith-Miles and S. Bowly, *Generating new test instances by evolving in instance space*, Computers & Operations Research 63 (2015), pp. 102–113.
[21] L.B. Sokolinsky and I.M. Sokolinskaya, *FRaGenLP: A Generator of Random Linear Programming Problems for Cluster Computing Systems*, in *International Conference on Parallel Computational Technologies*. Springer, 2021, pp. 164–177.
[22] S. Sremac, H.J. Woerdeman, and H. Wolkowicz, *Error bounds and singularity degree in semidefinite programming*, SIAM Journal on Optimization 31 (2021), pp. 812–836.
[23] T. Terlaky and Z. Wang, *On the identification of the optimal partition of second order cone optimization problems*, SIAM Journal on Optimization 24 (2014), pp. 385–414.
[24] M.J. Todd, *Probabilistic models for linear programming*, Mathematics of Operations Research 16 (1991), pp. 671–693.
[25] H. Wei and H. Wolkowicz, *Generating and measuring instances of hard semidefinite programs*, Mathematical Programming 125 (2010), pp. 31–45.

8. Appendices

Here, we review some basic procedures to generate random positive semidefinite matrices and orthogonal matrices, which can be used in the proposed SDOP and SOCOP generators.

Appendix A. Generating Random Positive Semidefinite Matrices

There are several approaches to generating a positive semidefinite matrix $P \in \mathcal{S}^n_+$.

(1) Generate a random matrix $A \in \mathbb{R}^{n\times n}$ and calculate the target matrix $P = AA^\top$. If A has full rank with probability 1, the matrix P is positive semidefinite with probability 1.
(2) A more efficient way is to generate a lower triangular random matrix $L \in \mathbb{R}^{n\times n}$ and calculate the target matrix $P = LL^\top$. If the diagonal elements of L are non-zero, then P is positive definite. If some of the diagonal elements of L are zero, then P is positive semidefinite.
(3) Generate an orthonormal matrix $Q \in \mathbb{R}^{n\times n}$ and a positive diagonal matrix Λ. Then calculate $P = Q\Lambda Q^\top$. If the diagonal elements of Λ are greater than zero, then P is positive definite. If diagonal elements of Λ are greater than or equal to zero, then P is positive semidefinite.
(4) Since generating a random orthonormal matrix is computationally expensive, we can generate a lower triangular random matrix $L \in \mathbb{R}^{n\times n}$ in which all diagonal elements are one instead. Then we can compute the target matrix as $P = LDL^\top$ where D is a diagonal matrix with non-negative elements.

The second one requires the fewest arithmetic operations among the four mentioned approaches. However, the third one gives the option of determining the range of eigenvalues of the matrix P, which is helpful in predetermining the condition number of matrix P.

Appendix B. Generating Random Orthogonal Matrices

A general approach is to generate a random matrix $A \in \mathbb{R}^{n\times n}$ and orthogonalize it by the Gram–Schmidt process or other methods in QR decomposition, such as Modified Gram–Schmidt and Householder methods. Generating random orthogonal (or unitary) matrices is an active research area and there are many efficient procedures to generate such matrices, e.g., see [13].