A Primal Approach to Facial Reduction for SDP Relaxations of Combinatorial Optimization Problems*

Hao Hu†, Mingming Xu‡

July 20, 2025

Abstract

We propose a novel facial reduction algorithm tailored to semidefinite programming relaxations of combinatorial optimization problems with quadratic objective functions. Our method leverages the specific structure of these relaxations, particularly the availability of feasible solutions that can often be generated efficiently in practice. By incorporating such solutions into the facial reduction process, we substantially simplify the reduction steps. On average, our facial reduction algorithm is four times faster than the standard implementation, providing significantly improved preprocessing for SDP relaxations in combinatorial optimization.

1 Introduction

Combinatorial optimization problems are central to operations research, computer science, and engineering, as they model a wide range of decision-making tasks involving discrete choices, such as routing, scheduling, and resource allocation. Traditionally, many of these problems have been studied with linear objective functions due to their mathematical tractability and the availability of well-established solution methods. However, extending the objective from linear to quadratic allows for modeling more complex interactions between decision variables, capturing dependencies that linear models cannot.

A prominent example is the Quadratic Assignment Problem (QAP), see, [3, 4], that arises in facility layout planning. In QAP, the cost depends not only on individual assignments but also on the interaction between pairs of facilities and their respective locations. This quadratic formulation provides a more accurate and realistic representation of practical problems, albeit at the cost of increased computational complexity.

Such problems can be formulated as a *mixed-integer quadratic programming (MIQP)* problem. While these problems are actually mixed-binary quadratic programming, we adopt the term MIQP for consistency with terminology such as MILP (Mixed-Integer

*Research supported by the Air Force Office of Scientific Research under award number FA9550-23-1-0508.

†School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC, USA. hhu2@clemson.edu

‡School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC, USA. mingmix@g.clemson.edu

Linear Programming). Let $Q \in \mathbb{R}^{n\times n}$ be a symmetric cost matrix and $c \in \mathbb{R}^n$ a linear cost vector. The MIQP is defined as:

$$\min \left\{ x^\top Q x + c^\top x \mid x \in P \right\}, \tag{1}$$

where the feasible region P is a mixed-binary set defined by the following constraints:

$$\begin{array}{rcll} a_i^\top x & = & b_i & \text{for } i = 1, \ldots, p, \\ a_i^\top x & \leq & b_i & \text{for } i = p+1, \ldots, m, \\ x_i & \in & \{0,1\} & \text{for } i = 1, \ldots, r, \end{array} \tag{2}$$

where $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ for $i = 1, \ldots, m$.

One of the most successful approaches for solving (1) is via *semidefinite programming (SDP)* relaxations, which provide strong convex approximations to otherwise hard combinatorial problems. Unlike traditional linear programming (LP) relaxations, SDP relaxations capture quadratic relationships and global structural properties, often leading to tighter bounds and higher-quality solutions. Landmark results, such as the Goemans–Williamson algorithm for Max-Cut [8], illustrate how SDP can yield approximation guarantees superior to those of LP-based methods. Furthermore, SDP formulations naturally appear in a variety of application domains and have achieved very good computational results, including graph partitioning [10, 36], sensor network localization [15, 16, 17], and quantum information theory [35, 27, 38, 43, 44, 37], further demonstrating their theoretical and practical relevance. The continuing development of efficient SDP solvers and scalable algorithms has made SDP a powerful tool in tackling complex combinatorial optimization problems.

Despite their advantages, SDP relaxations are computationally intensive and can be sensitive to problem regularity. We highlight two important aspects that are directly relevant to our results:

1. **Non-convex approaches.** Solving SDPs via standard interior-point methods can be prohibitively expensive for large-scale instances, due to their substantial memory and time requirements. To address this limitation, the non-convex Burer–Monteiro approach [12, 14] reformulates the SDP as a more scalable non-convex problem. This reformulation enables the use of efficient nonlinear optimization techniques. While the resulting problem is non-convex and may converge only to a local optimum, the method has shown strong empirical performance and offers a practical alternative for problems where solving the original SDP directly is computationally infeasible.

2. **Problem regularity.** SDP relaxations are known to be sensitive to the presence of strict feasibility. In particular, the failure of *Slater's condition*—the existence of a strictly feasible solution—can severely impair both the theoretical guarantees and numerical performance of SDP relaxations. In the absence of Slater's condition, the dual problem may exhibit a nonzero duality gap, and primal and dual optimal values may not coincide. This can lead to weaker bounds and unreliable results. Furthermore, numerical solvers may encounter instability or fail to converge.

   A standard remedy is *facial reduction*, a regularization procedure introduced in [1, 2], which iteratively reformulates the SDP to ensure strict feasibility. While facial reduction guarantees regularity, it typically requires solving a sequence of SDP auxiliary problems that may be as computationally demanding as the original problem. To improve practical applicability, several special *facial reduction algorithms* (FRAs) have been developed; see, e.g., [21, 26, 29, 32, 40].

In this paper, we propose a novel FRA specifically designed for SDP relaxations of MIQP problems. The guiding philosophy of our approach is akin to that of the non-convex Burer–Monteiro method: to exploit non-convex techniques in order to alleviate the computational burden of solving large-scale convex optimization problems. Our key insight is that, while MIQPs are generally difficult to solve, it is often straightforward to generate feasible solutions within the mixed-binary feasible set P in a structured manner. For instance, although the QAP is well-known for its computational difficulty, its feasible region consists of binary assignment matrices, which are simple to construct. These feasible solutions can be used to generate corresponding feasible matrix solutions for the SDP relaxation, thereby substantially simplifying the facial reduction process. We call the resulting method *primal FRA* to emphasize its key feature—leveraging the primal feasible solutions. The primal FRA is simple to implement and numerically robust in practice. We provide both theoretical and computational results related to the primal FRA in this paper. Numerical experiments demonstrate that our method substantially reduces computation time on nearly all benchmark instances and frequently restores Slater's condition without the need to solve additional SDP auxiliary problems.

Notations. For any set P, we denote its linear span by $\operatorname{span}(P)$, its affine hull by $\operatorname{aff}(P)$, its relative interior by $\operatorname{ri}(P)$, and its convex hull by $\operatorname{conv}(P)$. The all-zeros vector and matrix are denoted by $\mathbf{0}$ and $\mathbf{O}$, respectively; their dimensions will be clear from the context. If K is a closed convex cone, its *dual cone* is defined as

$$K^* := \{y \mid \langle y, x\rangle \geq 0, \ \forall x \in K\}.$$

The set of $n \times n$ symmetric matrices is denoted by $\mathbb{S}^n$. For $X, Y \in \mathbb{S}^n$, the trace inner product is defined as $\langle X, Y\rangle := \operatorname{tr}(XY)$. The set of $n \times n$ symmetric positive semidefinite matrices is denoted by $\mathbb{S}^n_+$, and the set of symmetric positive definite matrices is denoted by $\mathbb{S}^n_{++}$. The range space and the null space of a given matrix $M \in \mathbb{S}^n$ are denoted by $\operatorname{range}(M)$ and $\operatorname{null}(M)$, respectively.

2 Preliminaries

2.1 Facial Reduction

In this section, we review the theory of facial reduction. Let K be a closed convex cone. We say F is a *face* of K, denoted by $F \trianglelefteq K$, if $x, y \in K$ and $x + y \in F$ imply that $x, y \in F$. The conjugate face of F is $F^\triangle := K^* \cap F^\perp$. A face F of K is called *exposed* if it is of the form $F = K \cap w^\perp$ for some $w \in K^*$. The element w is then called an *exposing vector*. We say K is *exposed* if all of its faces are exposed. For example, the cone $\mathbb{S}^n_+$ is exposed; the non-empty faces of $\mathbb{S}^n_+$ can be characterized by linear subspaces: F is a non-empty face of $\mathbb{S}^n_+$ if and only if there exists a linear subspace $\mathcal{V} \subseteq \mathbb{R}^n$ such that

$$F = \left\{X \in \mathbb{S}^n_+ \mid \operatorname{range}(X) \subseteq \mathcal{V}\right\}.$$

Let L be an affine subspace such that $L \cap K \neq \emptyset$. We say *Slater's condition* holds for $L \cap K$ if $L \cap \operatorname{ri}(K) \neq \emptyset$, i.e., it contains a feasible solution in the relative interior of K.

Facial reduction exploits the following theorem of alternative

$$L \cap \operatorname{ri}(K) = \emptyset \quad \Leftrightarrow \quad L^\perp \cap (K^* \setminus K^\perp) \neq \emptyset.$$

Thus, $L \cap K$ fails Slater's condition if and only if there exists $w \in L^\perp \cap \mathbb{S}^n_+ \neq \emptyset$. When $K = \mathbb{S}^n_+$, w is an exposing vector of $\mathbb{S}^n_+$, and $F = \mathbb{S}^n_+ \cap w^\perp$ is a proper face of $\mathbb{S}^n_+$ containing $L \cap \mathbb{S}^n_+$. Thus, if $L \cap \mathbb{S}^n_+$ fails Slater's condition, then we can reformulate $L \cap \mathbb{S}^n_+$ as $L \cap F$ which is over a smaller cone. This procedure can be iterated for a finite number of times until $L \cap F$ satisfies Slater's condition. We outline FRA in Algorithm 1 below.

Algorithm 1 Facial Reduction Algorithm (FRA)

```
1: **Initialization:** Let $F_0 = K$, $i = 1$.
2: **while** we can pick $w_i \in L^\perp \cap (F^*_{i-1} \setminus F^\perp_{i-1})$ **do**
3:     Set $F_i \leftarrow F_{i-1} \cap w_i^\perp$.
4:     Set $i \leftarrow i + 1$.
5: **end while**
```

The finite convergence of FRA has also been established in several recent works; see [28, 22, 23]. While FRA offers a systematic framework for restoring Slater's condition for the intersection $L \cap K$, it faces a significant computational challenge: at each iteration, it requires solving the following problem,

$$w_i \in L^\perp \cap \left(F^*_{i-1} \setminus F^\perp_{i-1}\right),$$

which is itself a nontrivial optimization task. We refer to the problem of identifying such an element w_i as the *FR auxiliary problem*. For example, in the context of SDP, when applying FRA to $L \cap \mathbb{S}^n_+$, the FR auxiliary problem is to find a nonzero matrix in $L^\perp \cap \mathbb{S}^n_+$, which translates into solving an SDP feasibility problem. This implies that the FR auxiliary problem can be as difficult as the original SDP itself. Consequently, FRA may become impractical for large-scale problems or when applied directly to SDP relaxations of combinatorial optimization models.

In our computational result, we only apply the first iteration of the FRA. There are three main reasons for this choice. First, the backward stability of a single iteration of facial reduction has been established in [21], whereas it remains an open question whether the same guarantee holds for subsequent iterations. Second, as a preprocessing algorithm, FRA should adhere to the principle of being *simple and quick*, as advocated by Andersen and Andersen [7]. Performing a second iteration of facial reduction is often too computationally expensive to be practical in this context. Third, the primal FRA is highly effective in practice, often restoring Slater's condition immediately after just one iteration for the majority of benchmark instances.

2.2 SDP Relaxations

Consider the feasible set P defined in (2). We study SDP relaxations of the following form. By lifting feasible solutions into $\mathbb{S}^{n+1}$, we define:

$$S := \left\{ \begin{bmatrix} 1 \\ x \end{bmatrix} \begin{bmatrix} 1 & x^\top \end{bmatrix} \,\middle|\, x \in P \right\}. \tag{3}$$

Note that all matrices in S are positive semidefinite. For any polyhedral set $L \subseteq \mathbb{S}^{n+1}$ such that $S \subseteq L$, the intersection

$$L \cap \mathbb{S}^{n+1}_+ \tag{4}$$

forms an outer approximation to S and is referred to as an *SDP relaxation* of S. Since there is a one-to-one correspondence between the original feasible set P and its lifted counterpart S, we also refer to $L \cap \mathbb{S}^{n+1}_+$ as an SDP relaxation of P.

We can efficiently optimize a linear function over this set, yielding a lower bound for the original MIQP problem (1):

$$\begin{array}{ll} \min & \langle \tilde{Q}, Y \rangle \\ \text{s.t.} & Y \in L \cap \mathbb{S}^{n+1}_+, \end{array} \tag{5}$$

where

$$\tilde{Q} := \begin{bmatrix} 0 & \frac{1}{2}c^\top \\ \frac{1}{2}c & Q \end{bmatrix} \in \mathbb{S}^{n+1}.$$

We describe the so-called *arrow constraints* which are implied by the binary variables. The rows and columns of the matrices in S are indexed by $\{0, 1, \ldots, n\}$. For any $Y \in S$, the binary constraints on x imply $Y_{00} = 1$ and $Y_{0i} = Y_{ii}$ for $i = 1, \ldots, r$. Define the arrow operator arrow : $\mathbb{S}^{n+1} \to \mathbb{R}^{r+1}$ as:

$$\operatorname{arrow}(Y) := \begin{bmatrix} Y_{00} \\ Y_{11} - \frac{1}{2}(Y_{01} + Y_{10}) \\ \vdots \\ Y_{rr} - \frac{1}{2}(Y_{0r} + Y_{r0}) \end{bmatrix}. \tag{6}$$

Let e_0 be the standard unit vector in $\mathbb{R}^{r+1}$ with 1 in the first coordinate. The constraint $\operatorname{arrow}(Y) = e_0$, known as the arrow constraint, holds for all $Y \in S$. Note that the arrow operator depends on the number r of binary variables, though we omit the superscript for simplicity. In Section 5, we will describe three SDP relaxations of varying strength and computational cost, all of which will be used to test our special FRA.

To simplify the presentation of applying facial reduction to the SDP relaxation $L \cap \mathbb{S}^{n+1}_+$, we assume throughout the theoretical discussion that L is an affine set. Under this assumption, the FR auxiliary problem involves the same set $L^\perp \cap \mathbb{S}^{n+1}_+$ as in Algorithm 1. If $L \cap \mathbb{S}^{n+1}_+$ does not satisfy Slater's condition, applying the first iteration of Algorithm 1 to the SDP relaxation $L \cap \mathbb{S}^{n+1}_+$, we obtain a proper face $F \trianglelefteq \mathbb{S}^{n+1}_+$ such that $L \cap F = L \cap \mathbb{S}^{n+1}_+$. Since F can have lower dimension than $\mathbb{S}^{n+1}_+$, the reformulated problem $L \cap F$ constitutes a smaller SDP relaxation. More importantly, this reformulation often satisfies Slater's condition, thereby improving numerical stability compared to the original formulation.

For any face $F \trianglelefteq \mathbb{S}^{n+1}_+$, we refer to $L \cap F$ as a *facially reduced formulation* of $L \cap \mathbb{S}^{n+1}_+$ if

$$S \subseteq L \cap F \subseteq L \cap \mathbb{S}^{n+1}_+.$$

This definition slightly generalizes the standard FRA described in Algorithm 1, which only produces faces $F \trianglelefteq \mathbb{S}^{n+1}_+$ satisfying $L \cap F = L \cap \mathbb{S}^{n+1}_+$. The generalized notion is particularly useful for unifying various specialized FRA approaches under a common framework.

3 A Primal Approach to Facial Reduction

3.1 The Primal FRA

Standard implementations of FRA require finding a nonzero element in $L^\perp \cap \mathbb{S}^n_+$ which is itself a computationally demanding task. (Note that we use $\mathbb{S}^n_+$ for general discussions, and $\mathbb{S}^{n+1}_+$ when referring to SDP relaxations.) To address this challenge, several practical strategies have been proposed. One such approach is to replace $\mathbb{S}^n_+$ with a tractable inner approximation K, and instead search for a point in the subset $L^\perp \cap K$. This idea, known as *partial facial reduction*, was introduced in [29]. Despite variations in implementation, the essential goal of all FRAs remains the same: to reduce the computational burden associated with solving the FR auxiliary problem.

The method proposed in this work follows this principle but adopts a novel strategy. Our key observation is that any primal feasible solution exposes a face of $\mathbb{S}^n_+$ that contains the entire set $L^\perp \cap \mathbb{S}^n_+$. This observation enables us to reduce the dimension of $L^\perp \cap \mathbb{S}^n_+$. Specifically, we use a primal feasible solution to define a smaller face that still contains all exposing vectors in $L^\perp \cap \mathbb{S}^n_+$. This leads to the following special FRA:

Algorithm 2 Primal Facial Reduction Algorithm (Primal FRA)

```
    Input: An SDP relaxation $L \cap \mathbb{S}^n_+$ for $S$ in (3), and a feasible solution $X^* \in L \cap \mathbb{S}^n_+$.
    Output: A facially reduced formulation of $L \cap \mathbb{S}^n_+$.
 1: Define the face $G \trianglelefteq \mathbb{S}^n_+$,
        $$G := \left\{ X \in \mathbb{S}^n_+ \mid \mathrm{range}(X) \subseteq \mathrm{range}(X^*) \right\}.  \quad (7)$$
 2: if $X^*$ has maximum rank in $\mathrm{conv}(S)$ then
 3:     return $L \cap G$
 4: else
 5:     Find an element $W^* \in L^\perp \cap G^\triangle$. Define the face
        $$F := \left\{ X \in \mathbb{S}^n_+ \mid \mathrm{range}(X) \subseteq \mathrm{null}(W^*) \right\}.  \quad (8)$$
 6:     return $L \cap F$
 7: end if
```

The description of Algorithm 2 naturally raises several questions: How can a feasible solution be obtained? How can we determine whether it has maximum rank? These practical concerns are addressed in detail in Section 3.2, but for now, we focus on the correctness of the algorithm. In particular, we show that both $L \cap G$ and $L \cap F$ are facially reduced formulations of the original feasible set $L \cap \mathbb{S}^n_+$.

Lemma 3.1. *In Algorithm 2, we have $S \subseteq L \cap G \subseteq L \cap \mathbb{S}^n_+$, and $L \cap G$ satisfies Slater's condition.*

Proof. By construction, G is a face of $\mathbb{S}^n_+$, so $L \cap G \subseteq L \cap \mathbb{S}^n_+$. To show the inclusion $S \subseteq L \cap G$, let $X \in S$ be arbitrary. Since X^* has maximum rank in $\mathrm{conv}(S)$, we must have $\mathrm{range}(X) \subseteq \mathrm{range}(X^*)$, or else the average $\frac{1}{2}(X + X^*)$ would also be in $\mathrm{conv}(S)$ and have strictly higher rank than X^*, contradicting its maximality. Hence, $X \in G$, and thus $X \in L \cap G$. This proves that $S \subseteq L \cap G$.

Moreover, since $X^* \in \mathrm{ri}(G)$, it follows that $L \cap G$ satisfies Slater's condition. $\square$

Lemma 3.2. *In Algorithm 2, we have $L \cap F = L \cap \mathbb{S}^n_+$.*

Proof. It suffices to show that $L^\perp \cap \mathbb{S}^n_+ = L^\perp \cap G^\triangle$. Since $G^\triangle \trianglelefteq \mathbb{S}^n_+$, we clearly have $L^\perp \cap \mathbb{S}^n_+ \supseteq L^\perp \cap G^\triangle$. For the reverse inclusion, let $W \in L^\perp \cap \mathbb{S}^n_+$. Then $\langle W, X\rangle = 0$ for all $X \in L \cap \mathbb{S}^n_+$, and in particular for X^*. Therefore, $\text{range}(W) \subseteq \text{null}(X^*)$, which implies $W \in G^\triangle$. Thus, $W \in L^\perp \cap G^\triangle$, completing the proof. □

We highlight some features of the primal FRA: (1) If the algorithm returns $L \cap G$, then an SDP solve is completely avoided, and Slater's condition is guaranteed to hold; (2) If the rank of X^* is high, then the dimension of $L^\perp \cap G^\triangle$ is small, making it computationally inexpensive to search for an exposing vector. We refer to the task of finding an element in $L^\perp \cap G^\triangle$ as the *reduced FR auxiliary problem*; (3) If the rank of X^* is low, then finding a non-trivial element in $L^\perp \cap G^\triangle$ remains challenging. In this case, however, we can apply existing methods such as partial FRA or sieve SDP to this smaller set $L^\perp \cap G^\triangle$, which can still lead to computational savings.

3.2 Generating Feasible Solutions

While finding a feasible solution for a general SDP can be challenging, it is often relatively straightforward in the context of SDP relaxations arising from combinatorial optimization problems.

Let P denote the feasible set as defined in (2), and let $L \cap \mathbb{S}^{n+1}_+$ be its SDP relaxation as defined in (4). In view of the lifted formulation (3), it suffices to identify a feasible solution $x \in P$; the corresponding lifted matrix

$$\begin{bmatrix} 1 \\ x \end{bmatrix} \begin{bmatrix} 1 & x^\top \end{bmatrix}$$

is then a feasible solution in $L \cap \mathbb{S}^{n+1}_+$.

Naturally, it is desirable to construct a feasible matrix with higher rank in order to simplify the set $L^\perp \cap \mathbb{S}^{n+1}_+$. To achieve this, we may generate a collection of feasible solutions $x_0, \dots, x_k \in P$ and define

$$X^* := \frac{1}{k+1} \sum_{i=0}^{k} \begin{bmatrix} 1 \\ x_i \end{bmatrix} \begin{bmatrix} 1 & x_i^\top \end{bmatrix}. \tag{9}$$

It is straightforward to verify that the vectors $x_0, \dots, x_k$ are affinely independent if and only if $\text{rank}(X^*) = k + 1$. Thus, our goal is to generate as many affinely independent feasible solutions in P as possible. In particular, if a maximal affinely independent subset of P is found, then X^* attains the maximum possible rank among all matrices in $\text{conv}(S)$. This connection between the affine structure of the original feasible set and the maximum-rank feasible matrix in $\text{conv}(S)$ was first analyzed in [11].

We now describe a method for finding a maximal affinely independent subset of feasible solutions in P. Suppose that $x_0, \dots, x_k \in P$ are known to be affinely independent. We can either certify that they span the affine hull of P, or identify a new feasible solution $x \in P$ that is affinely independent of $x_0, \dots, x_k$. By iterating this procedure, we can incrementally construct a maximal affinely independent subset of P, which in turn yields a maximal-rank matrix X^* for use in Algorithm 2.

Lemma 3.3. *Let $v \in P$, and let H be a linear subspace such that $v + H \subseteq \operatorname{aff}(P)$. Then $v + H = \operatorname{aff}(P)$ if and only if*

$$u^\top(x - v) = 0 \quad \textit{for all } x \in P \textit{ and } u \in H^\perp. \tag{10}$$

Proof. Assume $v + H = \operatorname{aff}(P)$. Then for any $x \in P$, we have $x - v \in H$, which implies $u^\top(x - v) = 0$ for all $u \in H^\perp$, since every vector in $H^\perp$ is orthogonal to H.

Conversely, suppose $v + H \subsetneq \operatorname{aff}(P)$. Then there exists $x \in P$ such that $x - v \notin H$. By the fundamental theorem of linear algebra, there exists $u \in H^\perp$ such that $u^\top(x - v) \neq 0$. This shows that (10) does not hold, completing the proof. □

We can use the characterization of the affine hull of P provided in Lemma 3.3 to compute $\operatorname{aff}(P)$ as follows. Assume $v + H \subseteq \operatorname{aff}(P)$ for some $v \in P$ and a $(k-1)$-dimensional linear subspace H. Let $u_1, \ldots, u_{n-k+1}$ be any linearly independent vectors spanning $H^\perp$. Then condition (10) holds if and only if

$$\max\{u_i^\top x \mid x \in P\} = \min\{u_i^\top x \mid x \in P\}, \quad \text{for } i = 1, \ldots, n-k+1. \tag{11}$$

Thus, we can verify whether (10) holds by solving at most $2(n - k + 1)$ optimization problems of the form (11). Moreover, if equality fails for any index i, then either the maximizer or minimizer, say x^*, satisfies $u_i^\top(x^* - v) \neq 0$. This implies $x^* - v \notin H$, and by Lemma 3.3, we can construct a strictly larger subspace $\tilde{H} = \operatorname{span}(H \cup \{x^* - v\})$, for which $v + \tilde{H} \subseteq \operatorname{aff}(P)$.

Starting with an initial feasible solution $v \in P$ and $H = \{0\}$, we iteratively construct a set of affinely independent feasible points that span $\operatorname{aff}(P)$. This process terminates in at most n iterations. In practice, however, we may terminate the algorithm earlier due to computational limitations encountered while solving the subproblems in (11). The full procedure is summarized in Algorithm 3.

Algorithm 3 Generating a Feasible Matrix Solution

```
    Input: A mixed-binary feasible set $P$ as defined in (2).
    Output: A feasible matrix solution $X^* \in \operatorname{conv}(S)$, see (3).
 1: Find $v := x_0 \in P$ and set $H = \{0\}$.
 2: for $k = 1, \ldots, n$ do
 3:     Let $u_1, \ldots, u_{n-k+1}$ be linearly independent vectors spanning $H^\perp$.
 4:     if we can find $x_k \in P \setminus (v + H)$ via (11) then
 5:         $H \leftarrow \operatorname{span}(H \cup \{x_k - v\})$.
 6:     else if we can verify $v + H = \operatorname{aff}(P)$ via (11) then
 7:         return $X^*$ as defined in (9); it has maximum rank.
 8:     else if some (11) problems are intractable then
 9:         return $X^*$ as defined in (9).
10:     end if
11: end for
```

At this point, we emphasize a key distinction: the original MIQP problem (1) involves minimizing a non-convex quadratic objective over the feasible set P, whereas the subproblems in (11) involve linear objective functions over the same set P, and thus constitute mixed-integer linear programs (MILPs). Although both MIQP and MILP problems are

NP-hard and share the same feasible region, their practical tractability differs substantially. In practice, MILPs are often significantly easier to solve. For example, MILPs with over 200 variables can typically be solved within minutes using modern solvers, whereas MIQPs of comparable size may require hours. This disparity highlights the advantage of leveraging powerful MILP solvers to tackle problems that would otherwise be computationally prohibitive—an idea also explored in [25, 31] and [33].

The optimization problems in (11) can be broadly categorized into the following classes:

1. **Polynomial-time solvable problems.** In some cases, the problems in (11) admit polynomial-time solutions. A canonical example is the classical minimum spanning tree problem. In contrast, its generalization—the quadratic minimum spanning tree problem (QMSTP)—is NP-hard and has been studied extensively; see [9, 19, 20, 24, 18]. Recent works [41, 42] explore SDP-based relaxations of the QMSTP, where our proposed facial reduction method can be directly applied. For such problems, the primal FRA is guaranteed to restore Slater's condition in polynomial time.

2. **NP-hard problems with tractable feasibility.** In many practical applications, the problems in (11) are NP-hard, yet finding feasible solutions is relatively easy. This makes our method broadly applicable, even if solving (11) to optimality is intractable in theory. In our experiments, we were able to restore Slater's condition for a wide range of benchmark instances. When a new affinely independent feasible solution cannot be found, we terminate Algorithm 3 at Line 9 and return the current feasible matrix. Although this matrix may not have maximal rank, it often results in a significant size reduction of the FR auxiliary problem and improved computational performance.

3. **Intractable problems with difficult feasibility.** In rare cases, even finding a single feasible solution in P is computationally intractable. In such instances, our method cannot be applied to perform facial reduction. Fortunately, such cases are uncommon in practice, as evidenced by our empirical results on benchmark libraries such as MIPLIB.

In summary, although the proposed approach may involve solving NP-hard subproblems, it requires only a single feasible solution to initiate facial reduction. In most practical settings, such a point can be found efficiently or generated heuristically. Consequently, the method is both effective and broadly applicable across a wide range of real-world optimization problems.

Remark 3.4. *In practice, some of the optimization problems in* (11) *may not be solvable to optimality within a reasonable timeframe. However, it is important to note that solving* (11) *to optimality is only necessary when verifying the equality condition in* (10)*. In most cases, it suffices to find a feasible solution satisfying $u_i^\top(x-v) \neq 0$. This condition can often be efficiently checked during a branch-and-bound search. Specifically, whenever a new feasible solution is found, we verify whether $u_i^\top(x-v) \neq 0$ (up to a numerical tolerance). If so, we may terminate the search early and return the feasible solution.*

3.3 Reducing the Number of Subproblems

At the k-th iteration of Algorithm 3, the orthogonal complement $H^\perp$ is a subspace of dimension $n-k+1$. Consequently, up to $2(n-k+1)$ optimization problems of the form (11) may need to be solved. In the worst case, this results in approximately $\mathcal{O}(n^2)$ subproblems across all iterations. In this section, we show that it is often possible to reduce this number significantly by exploiting the structure of the problem.

Suppose there exists a linear subspace $\bar{H}$ of dimension $n - \ell$ such that

$$v + H \subseteq \mathrm{aff}(P) \subseteq v + \bar{H} \tag{12}$$

for some feasible solution $v \in P$ and current subspace H. We first show that such a subspace $\bar{H}$ can lead to a substantial reduction in the number of subproblems required. We then demonstrate how $\bar{H}$ can often be identified directly from the linear constraints that define P.

Since $\mathrm{aff}(P) - v \subseteq \bar{H}$, it follows that

$$u^\top(x - v) = 0 \quad \text{for all } x \in P \text{ and } u \in \bar{H}^\perp.$$

As $\bar{H}^\perp \subseteq H^\perp$, if we choose a set of linearly independent vectors $u_1, \ldots, u_{n-k+1}$ spanning $H^\perp$, with the first ℓ vectors $u_1, \ldots, u_\ell$ spanning $\bar{H}^\perp$, then we only need to solve (11) for $u_{\ell+1}, \ldots, u_{n-k+1}$. This reduces the number of subproblems in the k-th iteration by 2ℓ, which can be substantial when $\ell > 0$, i.e., when $\bar{H}$ is not full-dimensional.

Furthermore, the maximum number of iterations in Algorithm 3 is reduced from n to $n - \ell$. If the algorithm reaches the $(n - \ell)$-th iteration, the current subspace H has dimension ℓ, and since $v + H \subseteq \mathrm{aff}(P) \subseteq v + \bar{H}$, and both subspaces share the same dimension, it follows that

$$\mathrm{aff}(P) = v + H = v + \bar{H}.$$

Hence, no optimization problems need to be solved in the final iteration. This not only accelerates the algorithm but also provides a practical and effective termination criterion in our implementation. We describe such an application in Remark 3.5.

Importantly, the subspace $\bar{H}$ in (12) often arises naturally from the definition of P. Specifically, consider the linear programming relaxation of the mixed-binary set P, given by:

$$\begin{array}{rcll} a_i^\top x & = & b_i & \text{for } i = 1, \ldots, p, \\ a_i^\top x & \leq & b_i & \text{for } i = p+1, \ldots, m. \end{array} \tag{13}$$

The affine hull of this polyhedron—obtained by identifying the set of (implicit) equality constraints—provides a natural candidate for the subspace $\bar{H}$. This subspace can be extracted efficiently using standard techniques in linear programming, and doing so can significantly reduce the computational cost of Algorithm 3.

Remark 3.5. *We discuss an application of the linear subspace $\bar{H}$ in the implementation. Many MIPLIB problem instances are known to suffer from numerical instability due to scaling issues. To mitigate this, we restrict our attention to feasible solutions with bounded entries. Specifically, we define the restricted feasible set*

$$\bar{P} := P \cap \{x \in \mathbb{R}^n \mid x_i \in [10^{-5}, 10^5]\},$$

and search for feasible solutions within $\bar{P}$ instead of the full set P. As a consequence, we may no longer be able to certify that the affine hull of P has been fully recovered, since $\mathrm{aff}(\bar{P}) \subseteq \mathrm{aff}(P)$. Nevertheless, if we succeed in identifying $n - \ell + 1$ affinely independent feasible solutions within $\bar{P}$, we can still conclude that these solutions span the affine hull of P.

It is important to emphasize that facial reduction is still being performed on the original set P; the restriction to $\bar{P}$ is introduced solely to reduce numerical instability in practice.

3.4 A Quadratic Speedup via Randomization

While the technique discussed in the previous section reduces the number of subproblems, the total number of optimization problems remains $\mathcal{O}(n^2)$ in the worst case. In this section, we demonstrate that it suffices to solve at most two optimization problems per iteration without imposing any additional structural assumptions. This yields a significant improvement, reducing the total number of required MILP problems to $\mathcal{O}(n)$, thereby achieving a quadratic speedup.

Lemma 3.6. *Assume $v+H \subsetneq \operatorname{aff}(P)$, and let $u_1, \dots, u_{n-k+1} \in \mathbb{R}^n$ be linearly independent vectors spanning $H^\perp$. Let $\alpha_1, \dots, \alpha_{n-k+1} \in \mathbb{R}$ be i.i.d. random variables drawn from the standard normal distribution, and define $u := \sum_{i=1}^{n-k+1} \alpha_i u_i$. Let $x^*_{\min}$ and $x^*_{\max}$ be the optimal solutions of*

$$\min\{u^\top x \mid x \in P\} \quad \textit{and} \quad \max\{u^\top x \mid x \in P\},$$

respectively. Then, with high probability, either $u^\top(x^*_{\min} - v) < 0$ or $u^\top(x^*_{\max} - v) > 0$.

Proof. Since $v + H \subsetneq \operatorname{aff}(P)$, there exists some $x \in P$ such that $x - v \notin H$. Therefore, $u_i^\top(x-v) \neq 0$ for at least one i. Because the coefficients α_i are drawn independently from a continuous distribution, the linear combination $u^\top(x - v) = \sum_{i=1}^{n-k+1} \alpha_i u_i^\top(x - v) \neq 0$ with probability 1. In particular, if $u^\top(x - v) < 0$, then $u^\top(x^*_{\min} - v) < 0$; similarly, if $u^\top(x - v) > 0$, then $u^\top(x^*_{\max} - v) > 0$. □

The following example shows why the randomness assumption in Lemma 3.6 is essential. Consider the set

$$P = \{e_1, e_2, e_3\},$$

where e_i is the i-th standard basis vector in $\mathbb{R}^3$. Let $v = e_1$ and $H = \{0\}$, so that $H^\perp = \mathbb{R}^3$. Choose $u_i = e_i$ for $i = 1, 2, 3$, which clearly span $H^\perp$. If we select coefficients deterministically with $\alpha_1 = \alpha_2 = \alpha_3 = 1$, then the resulting vector $u = \sum_{i=1}^3 \alpha_i u_i$ is the all-ones vector.

Now, consider $x = e_2 \in P$. We have $x - v = e_2 - e_1 = \begin{bmatrix}-1 & 1 & 0\end{bmatrix}^T$, and hence $u^\top(x - v) = \begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}-1 \\ 1 \\ 0\end{bmatrix} = 0$. Thus, the direction u fails to detect that $x \notin v + H$. However, if the coefficients α_i are chosen randomly from a continuous distribution, then

$$u^\top(x - v) = -\alpha_1 + \alpha_2 \neq 0$$

with high probability. This highlights the necessity of using random coefficients to guarantee a correct result in Lemma 3.6.

3.5 Numerical Robustness

Numerical algorithms often struggle to distinguish between values that are truly zero and those that are merely close to zero due to rounding errors or the limitations of finite-precision arithmetic. In the context of FRAs, accurately identifying the range space of the feasible matrix is essential, as even minor misclassifications can compromise the correctness of the reduced problem.

In this section, we first clarify how numerical issues may arise in the primal FRA. (We emphasize, however, that such numerical difficulties were rarely encountered in our computational experiments.) We then propose a heuristic approach to mitigate these numerical issues when they do occur. Finally, we argue that in the rare event where numerical instability persists and the heuristic fails, one can eliminate these issues by taking a conservative inner approximation of the face G. This simple modification guarantees numerical stability, at the cost of only a marginally smaller reduction in problem size.

We first clarify the potential numerical issues in the primal FRA. In Algorithm 2, the face G is determined by the range space of the matrix $X^* = VV^\top$, where V is constructed from feasible solutions $x_0, \dots, x_k \in P$ via Algorithm 3:

$$V := \frac{1}{\sqrt{k+1}} \begin{bmatrix} 1 & \cdots & 1 \\ x_0 & \cdots & x_k \end{bmatrix}. \tag{14}$$

Assume the affine subspace spanned by $x_0, \dots, x_k$ is $\operatorname{aff}(P)$ and $k < n$. Let $\sum_{i=1}^{k+1} \sigma_i u_i v_i^\top$ be the numerical singular value decomposition of V. In practice, the range space of X^* is numerically approximated as $\operatorname{span}\{u_i \mid \sigma_i > \delta\}$, for some threshold δ. However, ambiguity arises when $\sigma_i \in (\epsilon_{\text{mach}}, \delta]$, where ϵ_{mach} denotes the machine epsilon—the smallest number such that $1 + \epsilon_{\text{mach}} \neq 1$ in floating-point arithmetic (typically around 2.22×10^{-16} for double-precision). Singular values in this range may be inconsistently treated as either zero or nonzero by numerical solvers. Misinterpreting a near-zero singular value as positive can lead to overestimating the range space, potentially yielding an incorrect reduced face $L^\perp \cap G^\triangle \subsetneq L^\perp \cap \mathbb{S}^n_+$.

To mitigate this issue, we propose augmenting the matrix V by including an additional feasible solution $x_{k+1} \in P$ that increases an ambiguous singular value σ_r. Let $u_r = \begin{bmatrix} h_r \\ g_r \end{bmatrix}$ be the left singular vector of V corresponding to σ_r. We solve:

$$\max \left\{ g_r^\top x \mid x \in P \right\}, \tag{15}$$

to find a new point x_{k+1} that aligns with direction g_r, thus improving the conditioning of the extended matrix:

$$\bar{V} := \frac{1}{\sqrt{k+2}} \begin{bmatrix} 1 & \cdots & 1 & 1 \\ x_0 & \cdots & x_k & x_{k+1} \end{bmatrix}.$$

The corresponding matrix $\bar{X} = \bar{V}\bar{V}^\top$ remains feasible for $\operatorname{conv}(S)$, and results in the same face G, but with potentially improved numerical stability.

We formalize this improvement below:

Lemma 3.7. *Let V be defined as in (14), and suppose $\sigma_r(V) < 1$ is the r-th singular value. Let $u_r = \begin{bmatrix} h_r \\ g_r \end{bmatrix}$ be the corresponding left singular vector. Define:*

$$U := \frac{1}{\sqrt{k+2}} \begin{bmatrix} 1 & \cdots & 1 & h_r \\ x_0 & \cdots & x_k & g_r \end{bmatrix}.$$

Then the r-th singular value of U satisfies $\sigma_r(U) > \sigma_r(V)$.

Proof. Let $X^* = VV^\top$ and $Y^* = UU^\top$. Then:

$$Y^* = \frac{k+1}{k+2}X^* + \frac{1}{k+2}\begin{bmatrix} h_r \\ g_r \end{bmatrix}\begin{bmatrix} h_r & g_r^\top \end{bmatrix}.$$

Let $\lambda_r(M)$ denote the r-th eigenvalue of a symmetric matrix M. Then:

$$\lambda_r(Y^*) = \frac{k+1}{k+2}\lambda_r(X^*) + \frac{1}{k+2} > \lambda_r(X^*) = \sigma_r^2(V),$$

so $\sigma_r(U) > \sigma_r(V)$. □

Of course, the vector g_r in Lemma 3.7 may not lie in P. In practice, we approximate this direction by using a feasible solution x that maximizes $g_r^T x$. This heuristic motivates the formulation in (15). Although we do not have a formal bound for (15), our experiments consistently show that this approach improves the conditioning of V—increasing the target singular value and reducing ambiguity in rank determination.

In rare cases where ambiguous singular values persist—indicating either the failure of the above strategy or an ill-conditioned feasible set P—we simply treat all such singular values as zero. This leads to the following inner approximation:

$$\bar{G} := \operatorname{span}\left\{u_i \mid \sigma_i(V) > \bar{\delta}\right\},$$

where $\bar{\delta}$ is chosen conservatively, e.g., $\bar{\delta} \in [10^{-5}, 10^{-3}]$. Since $\bar{G} \subseteq G$, it follows that $\bar{G}^\triangle \supseteq G^\triangle$, and therefore:

$$L^\perp \cap G^\triangle \subseteq L^\perp \cap \bar{G}^\triangle.$$

With a sufficiently large $\bar{\delta}$, we can confidently conclude:

$$L^\perp \cap \bar{G}^\triangle = L^\perp \cap \mathbb{S}^n_+.$$

In practice, the number of ambiguous singular values is typically zero or very small, and the proposed inner approximation $\bar{G}$ provides a robust and reliable implementation of the primal FRA, while preserving nearly all of its reduction capability.

Remark 3.8. *While $\bar{\delta}$ can improve numerical robustness in some cases, we did not enable this feature in our implementation, as we applied the same settings uniformly across all problem instances without instance-specific tuning.*

Remark 3.9. *Note that we only claim the primal FRA does not introduce additional numerical issues. If an SDP problem is ill-posed and numerical difficulties arise in the standard FRA—specifically, in Line 3 of Algorithm 1—then similar numerical issues are likely to occur when determining the face F in the primal FRA formulation (8).*

The key advantage of the primal FRA lies in its ability to compute the exposing vector more efficiently, without introducing new sources of numerical instability. However, it does not eliminate existing numerical challenges inherent to the original problem.

4 Relation to Other Approaches

In this section, we discuss the relationship between the primal FRA and several alternative FRAs. The primal FRA is highly compatible with many of these alternatives, allowing us to combine their advantages rather than choosing a single method. We illustrate this synergy using partial FRA [29] as an example. We begin with a brief overview of the key ideas behind partial FRA and then describe how it integrates with the primal FRA. We also compare the primal FRA with the affine FRA proposed in [40], highlighting important distinctions and advantages.

Partial FRA replaces the positive semidefinite cone $\mathbb{S}^n_+$ with an inner approximation $K \subseteq \mathbb{S}^n_+$ such that $L^\perp \cap K \subseteq L^\perp \cap \mathbb{S}^n_+$. By choosing K to be computationally simpler, one obtains a more practical method for identifying an exposing vector. A widely used choice is the cone of diagonally dominant matrices, defined by:

$$\mathcal{DD}^n := \left\{ X \in \mathbb{S}^n \;\middle|\; X_{ii} \geq \sum_{j \neq i} |X_{ij}| \right\}.$$

Since $\mathcal{DD}^n \subseteq \mathbb{S}^n_+$, searching for a nonzero matrix in $L^\perp \cap \mathcal{DD}^n$ reduces to solving a linear program. The trade-off is that $L^\perp \cap \mathcal{DD}^n$ may contain only the zero matrix, even when $L^\perp \cap \mathbb{S}^n_+$ contains a nonzero exposing vector—resulting in failure to detect a valid reduction.

In contrast, the primal FRA identifies a proper face $G^\triangle$ such that $L^\perp \cap \mathbb{S}^n_+ = L^\perp \cap G^\triangle$, thereby reducing the dimension of the FR auxiliary problem. These two methods can be effectively combined. One may first apply the primal FRA to obtain a reformulation over a smaller face $G^\triangle$, and then apply partial FRA to construct an inner approximation $\tilde{K} \subseteq G^\triangle$. Since $G^\triangle$ is a proper face of $\mathbb{S}^n_+$, the resulting $\tilde{K}$ is lower-dimensional and more manageable than an inner approximation over the full cone. For example, if

$$G^\triangle = \left\{ \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} \;\middle|\; R \in \mathbb{S}^r_+ \right\}, r < n,$$

then a natural inner approximation is

$$\tilde{K} = \left\{ \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} \;\middle|\; R \in \mathcal{DD}^r \right\}.$$

This demonstrates how primal FRA and partial FRA can be seamlessly combined, benefiting from both reduction quality and computational efficiency. The same strategy applies to other methods such as Sieve-SDP [32] and the preprocessing approach in [21].

The affine FRA proposed in [40] adopts a different perspective: it identifies implicit equalities by analyzing the affine hull of the feasible region P through its LP relaxation. This yields an affine subspace containing P, although not necessarily the minimal one. The affine FRA performs well in practice, particularly on benchmark instances from MIPLIB. It often achieves the same reduction as the primal FRA, sometimes with less computational effort. However, it does not offer guarantees regarding the satisfaction of Slater's condition for the reduced problem. In contrast, the primal FRA explicitly verifies that Slater's condition holds for most instances—a critical advantage in applications where numerical precision and stability are paramount.

5 Numerical Experiments

5.1 Settings

Experimental Settings: All experiments were conducted on a Mac Studio (2023) equipped with an Apple M2 Ultra chip, 128 GB of RAM, and running macOS 14.1.1 (build 23B81). We generated MIQP problem instances based on benchmark instances from MIPLIB 2017 [34], specifically those categorized under "collection." The instances were imported into MATLAB R2023b using Gurobi (version 10.0.1) [39]. We also used Gurobi to solve problem (11), which generates feasible solutions within Algorithm 3. The SDP problems were solved using the MOSEK solver in [30] via the MATLAB interface YALMIP [13].

Problem Generation: The MIPLIB provides MILP problems whose feasible regions are of the same form as in (2). In addition, MIPLIB specifies a linear objective function $c^T x$ for some vector $c \in \mathbb{R}^n$. To construct MIQP instances, we generate a random symmetric quadratic cost matrix $Q \in \mathbb{S}^n$, where the entries are independently drawn from a standard normal distribution. This results in an MIQP problem of the form (1). Note that the objective function does not affect facial reduction; it is used only when solving the original SDP problem in Section A.

SDP Relaxations: For small to moderate problem instances, we apply the standard FRA and the primal FRA to the following two well-known SDP relaxations:

1. **Shor's Relaxation.**

   Define the matrices:

$$\tilde{Q} := \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & Q \end{bmatrix} \in \mathbb{S}^{n+1} \text{ and } A_i := \begin{bmatrix} 0 & \frac{1}{2}a_i^\top \\ \frac{1}{2}a_i & \mathbf{O} \end{bmatrix} \in \mathbb{S}^{n+1}, \quad i = 1, \dots, m.$$

   Then the relaxation is given by:

$$\begin{aligned} \min \quad & \langle \tilde{Q}, Y \rangle \\ \text{s.t.} \quad & \langle A_i, Y \rangle = b_i, \quad i = 1, \dots, p && (16) \\ & \langle A_i, Y \rangle \leq b_i, \quad i = p+1, \dots, m && (17) \\ & \operatorname{arrow}(Y) = e_0 && (18) \\ & Y \in \mathbb{S}^{n+1}_+. && (19) \end{aligned}$$

2. **Doubly Nonnegative (DNN) Relaxation.**

   Define:

$$\bar{A}_i := \begin{bmatrix} -b_i \\ a_i \end{bmatrix} \begin{bmatrix} -b_i & a_i^\top \end{bmatrix} \in \mathbb{S}^{n+1}.$$

   In many problems, the inequality constraints include variable bounds $l \leq x \leq u$. We can derive additional constraints based on the Reformulation-Linearization

Technique (RLT) [5, 6]. This yields the following SDP relaxation:

$$\begin{array}{rl} \min & \langle \tilde{Q}, Y \rangle \\ \text{s.t.} & Y \text{ satisfies } (16), (17), (18), (19) \\ & \langle \bar{A}_i, Y \rangle = 0, \quad i = 1, \dots, p \\ & l_i l_j - Y_{0i} l_j - Y_{0j} l_i + Y_{ij} \geq 0, \quad \forall i, j \\ & u_i u_j - Y_{0i} u_j - Y_{0j} u_i + Y_{ij} \geq 0, \quad \forall i, j. \end{array}$$

When the problem contains only binary variables, the matrix Y is both positive semidefinite and entrywise nonnegative; hence, we refer to this as the *doubly nonnegative (DNN)* relaxation.

For large-scale instances, solving Shor's or DNN relaxations may be computationally prohibitive. We consider a special case in which quadratic costs appear only between binary variables, i.e.,

$$Q = \begin{bmatrix} Q_B & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \text{ for some } Q_B \in \mathbb{S}^r.$$

Here, $Q_B \in \mathbb{S}^r$ represents the quadratic cost matrix for binary variables. Let the first r variables be binary, and decompose:

$$x = \begin{bmatrix} \tilde{x} \\ \hat{x} \end{bmatrix}, \quad \tilde{x} \in \mathbb{R}^r, \quad \hat{x} \in \mathbb{R}^{n-r}.$$

Then the MIQP in (1) can be written as:

$$\min \left\{ \tilde{x}^\top Q_B \tilde{x} + c^\top x \,\middle|\, x \in P \right\}. \tag{20}$$

Let $\tilde{a}_i \in \mathbb{R}^r$, $\hat{a}_i \in \mathbb{R}^{n-r}$, $\tilde{c} \in \mathbb{R}^r$ and $\hat{c} \in \mathbb{R}^{n-r}$ be defined analogously. Define:

$$\tilde{Q} := \begin{bmatrix} 0 & \tilde{c}^T \\ \tilde{c} & Q_B \end{bmatrix} \in \mathbb{S}^{r+1} \text{ and } \tilde{A}_i := \begin{bmatrix} 0 & \frac{1}{2}\tilde{a}_i^\top \\ \frac{1}{2}\tilde{a}_i & \mathbf{O} \end{bmatrix} \in \mathbb{S}^{r+1}.$$

The following is an SDP relaxation for (20):

$$\begin{array}{rl} \min & \langle \tilde{Q}, X \rangle + \hat{c}^\top \hat{x} \\ \text{s.t.} & \langle \tilde{A}_i, X \rangle + \hat{a}_i^\top \hat{x} \leq b_i, \quad i = 1, \dots, p \\ & \langle \tilde{A}_i, X \rangle + \hat{a}_i^\top \hat{x} = b_i, \quad i = p+1, \dots, m \\ & \operatorname{arrow}(X) = e_0 \\ & X \in \mathbb{S}_+^{r+1}. \end{array} \tag{21}$$

Here, the matrix variable X is only of order $r + 1$, which is typically much smaller than $n + 1$, leading to a significant reduction in computational cost. This significantly reduces computational cost and enables the testing of larger problem instances. We will call (20) a *variant of Shor's relaxation.*

5.2 Results and Discussion

We report the numerical performance of the standard FRA and the proposed primal FRA on a diverse set of benchmark instances. We begin by presenting and discussing overall performance statistics to highlight the key advantages of the primal FRA. Detailed numerical results for individual problem instances follow, providing a deeper analysis of algorithmic behavior across different SDP relaxations.

Table 1 provides an aggregated comparison between the standard FRA and the proposed primal FRA across three types of SDP relaxations and a total of 157 problem instances. The following key performance metrics are reported:

1. **Total Time**: The total computation time in seconds required to perform FRA across all instances. (If the standard FR auxiliary problem could not be solved due to memory limitations, we exclude the computational time for both FRA methods.) The primal FRA achieves a substantial reduction in total computation time—from 31,876 seconds to 6,975 seconds—yielding a fourfold speedup.

2. **Number of Successful Solves**: The number of instances for which the FR auxiliary problem was successfully solved. The primal FRA achieves a higher success rate (120 out of 157) compared to the standard FRA (146 out of 157), demonstrating its improved robustness.

3. **Slater Condition Restored**: The number of instances in which Slater's condition was successfully restored. For the standard FRA, we apply one step of Algorithm 1. The primal FRA restores Slater's condition in 145 instances, while the standard FRA does so in only 108, further demonstrating the effectiveness of the primal approach.

4. **Average Size Ratio**: For each instance, we compute the ratio between the matrix variable size in the reduced FR auxiliary problem from the primal FRA and that from the standard FRA. The reported value is the average ratio across all instances. On average, the matrix variable in the primal FRA is approximately 9% the size of that in the standard FRA, indicating a substantial reduction in problem size.

These results show that the primal FRA not only improves computational efficiency but also enhances solver reliability and produces significantly more compact SDP formulations. This makes the primal FRA a promising alternative to the standard FRA for solving large-scale semidefinite relaxations.

Metric	**Standard FRA**	**Primal FRA**
Total Time	31876	6975
Successful Solves (Count)	120	146
Slater Condition Restored (Count)	108	145
Average Size Ratio	1	0.09179

Table 1: Summary of total time, number of successful solves, and Slater condition restorations over 157 instances.

Tables 2 to 5 present detailed experimental results. For each instance, we report the following information:

1. **Size**: The order of the matrix variable in the FR auxiliary problem. For the standard FRA, this is equal to the matrix variable order in the original SDP relaxation. For the primal FRA, it corresponds to the matrix variable order in the reduced FR auxiliary problem. A value of zero indicates that the primal FRA terminated at Line 3 of Algorithm 2, restoring Slater's condition without solving the FR auxiliary problem.

2. **Time**: The total time in seconds required to perform facial reduction. For the standard FRA, this is the time to solve the FR auxiliary problem. For the primal FRA, we report two components: T_1 (time to construct a feasible solution) and T_2 (time to solve the reduced FR auxiliary problem).

3. **Solver Status**: The status returned by MOSEK. *Success* indicates that the SDP was solved correctly. *Stall* denotes limited progress due to numerical issues, although the solution may still be usable. *Fail* indicates that a reliable solution could not be obtained. *Out of memory* means the problem could not be solved due to insufficient system memory.

4. **Slater Condition**: *Yes* indicates that the facially reduced formulation satisfies Slater's condition. *No* indicates that Slater's condition was not verified.

The results demonstrate that the primal FRA substantially improves computational efficiency over the standard FRA. In nearly all cases, the size of the FR auxiliary problem is significantly reduced under the primal FRA, indicating more effective facial reduction. Moreover, the total runtime is consistently lower, with the overhead of the primal FRA (i.e., computing a feasible solution) being negligible relative to solving the reduced FR auxiliary problem.

A further key advantage is improved numerical robustness. Several instances that failed under the standard FRA due to numerical instability were successfully handled by the primal FRA. In particularly challenging cases where the standard FRA did not restore Slater's condition, the primal FRA succeeded, improving solver reliability and ensuring strong duality.

Overall, the empirical evidence strongly supports the effectiveness of the primal FRA in reducing auxiliary problem complexity, enhancing numerical stability, and restoring Slater's condition—while incurring minimal computational overhead.

6 Conclusion

We have proposed a novel facial reduction algorithm (FRA) tailored to semidefinite relaxations of combinatorial optimization problems. Unlike traditional approaches that require solving computationally intensive FR auxiliary problems, the primal FRA formulates a reduced FR auxiliary problem that is significantly more tractable and efficient.

The proposed method integrates seamlessly with other specialized FRAs, such as partial FRA and Sieve-SDP, enabling complementary preprocessing strategies. Moreover, it exhibits strong numerical robustness, successfully addressing instances where the standard FRA fails due to numerical instability or memory limitations.

Extensive computational experiments demonstrate that the primal FRA substantially reduces the size and complexity of the FR auxiliary problem, restores Slater's condition in the majority of benchmark instances, and improves solver reliability—all while incurring minimal preprocessing overhead. These results underscore the practical value of the primal FRA as an effective and scalable tool for enhancing the performance of SDP relaxations in combinatorial optimization.

Name	The standard FRA				The primal FRA					
	Size	Time	Solver Status	Slater condition	Size	T_1	T_2	Time	Solver Status	Slater condition
gr4x6	49	0.25	Success	No	0	0.21	0.00	0.21	Success	Yes
markshare-5-0	46	0.28	Success	No	0	0.28	0.00	0.28	Success	Yes
markshare-4-0	35	0.36	Success	No	0	1.18	0.00	1.18	Success	Yes
markshare1	63	1.08	Success	No	0	5.40	0.00	5.40	Success	Yes
neos5	64	1.21	Success	Yes	0	0.48	0.00	0.48	Success	Yes
markshare2	75	2.23	Success	No	0	0.48	0.00	0.48	Success	Yes
pk1	87	3.23	Success	No	0	0.86	0.00	0.86	Success	Yes
b-ball	101	5.61	Success	No	0	0.56	0.00	0.56	Success	Yes
neos-5192052-neckar	181	51.29	Success	No	0	2.03	0.00	2.03	Success	Yes
v150d30-2hopcds	151	54.32	Success	No	0	11.36	0.00	11.36	Success	Yes
mas74	152	112.76	Success	Yes	0	6.02	0.00	6.02	Success	Yes
mas76	152	122.36	Success	Yes	0	2.45	0.00	2.45	Success	Yes
gsvm2rl3	242	158.02	Success	Yes	0	6.95	0.00	6.95	Success	Yes
assign1-5-8	157	171.03	Success	No	0	4.81	0.00	4.81	Success	Yes
2club200v15p5scn	201	192.38	Success	Yes	0	12.77	0.00	12.77	Success	Yes
neos-5140963-mincio	197	250.46	Success	No	28	114.24	4.44	118.68	Success	No
glass-sc	215	320.04	Success	Yes	0	13.71	0.00	13.71	Success	Yes
iis-glass-cov	215	330.77	Success	Yes	0	13.50	0.00	13.50	Success	Yes
mad	221	479.20	Success	No	0	3.16	0.00	3.16	Success	Yes
neos-3754480-nidda	254	481.36	Success	Yes	0	41.20	0.00	41.20	Success	Yes
p0201	202	498.13	Failed	N.A.	62	3.29	32.25	35.53	Failed	N.A.
prod1	251	1027.85	Success	No	44	28.77	6.54	35.31	Success	No
prod2	302	2117.50	Success	No	39	60.37	8.64	69.01	Success	No
supportcase14	305	4509.23	Success	No	201	7.15	548.62	555.77	Success	No

Table 2: Algorithm performance on the DNN relaxation.

Name	The standard FRA				The primal FRA					
	Size	Time	Solver Status	Slater condition	Size	T_1	T_2	Time	Solver Status	Slater condition
markshare-4-0	35	0.06	Success	Yes	0	1.21	0.00	1.21	Success	Yes
markshare-5-0	46	0.09	Success	Yes	0	0.32	0.00	0.32	Success	Yes
gr4x6	49	0.13	Success	Yes	0	0.24	0.00	0.24	Success	Yes
neos5	64	0.29	Success	Yes	0	0.48	0.00	0.48	Success	Yes
markshare1	63	0.30	Success	Yes	0	5.43	0.00	5.43	Success	Yes
markshare2	75	0.57	Success	Yes	0	0.49	0.00	0.49	Success	Yes
pk1	87	1.27	Success	Yes	0	0.88	0.00	0.88	Success	Yes
b-ball	101	2.74	Success	Yes	0	0.59	0.00	0.59	Success	Yes
mas76	152	12.16	Success	Yes	0	2.49	0.00	2.49	Success	Yes
v150d30-2hopcds	151	13.33	Success	No	0	11.37	0.00	11.37	Success	Yes
mas74	152	13.39	Success	Yes	0	6.00	0.00	6.00	Success	Yes
assign1-5-8	157	25.12	Stall	Yes	0	4.94	0.00	4.94	Success	Yes
neos-5192052-neckar	181	30.52	Success	Yes	0	2.03	0.00	2.03	Success	Yes
neos-5140963-mincio	197	42.30	Success	Yes	28	113.99	1.26	115.25	Success	Yes
p0201	202	43.02	Success	Yes	62	2.90	7.83	10.73	Success	Yes
2club200v15p5scn	201	53.26	Success	Yes	0	12.74	0.00	12.74	Success	Yes
iis-glass-cov	215	58.69	Success	Yes	0	13.53	0.00	13.53	Success	Yes
glass-sc	215	67.97	Success	Yes	0	13.91	0.00	13.91	Success	Yes
mad	221	72.56	Success	Yes	0	2.67	0.00	2.67	Success	Yes
gsvm2rl3	242	124.86	Success	Yes	0	7.29	0.00	7.29	Success	Yes
prod1	251	155.15	Success	Yes	44	29.21	2.34	31.55	Failed	N.A.
supportcase14	305	354.02	Success	No	201	7.55	116.64	124.19	Success	No
neos-3754480-nidda	254	415.66	Failed	N.A.	0	41.66	0.00	41.66	Success	Yes
prod2	302	443.06	Success	Yes	39	60.52	2.43	62.95	Success	Yes
supportcase16	320	464.48	Success	No	217	8.53	137.82	146.35	Success	No
iis-hc-cov	298	475.61	Failed	N.A.	0	18.63	0.00	18.63	Success	Yes
neos-1430701	313	569.62	Success	Yes	0	5.14	0.00	5.14	Success	Yes
probportfolio	321	675.83	Success	Yes	276	102.37	255.10	357.47	Success	Yes
ran13x13	339	992.48	Success	Yes	0	9.76	0.00	9.76	Success	Yes
control30-3-2-3	333	1250.47	Failed	N.A.	171	345.81	338.36	684.17	Stall	Yes
pigeon-08	345	1378.78	Success	Yes	152	231.30	47.44	278.74	Success	Yes
nsa	389	N.A.	Out of Memory	N.A.	96	123.16	27.10	150.27	Failed	N.A.
gsvm2rl5	402	N.A.	Out of Memory	N.A.	0	41.24	0.00	41.24	Success	Yes
neos-3611689-kaihu	422	N.A.	Out of Memory	N.A.	0	14.37	0.00	14.37	Success	Yes
neos-3610040-iskar	431	N.A.	Out of Memory	N.A.	0	13.51	0.00	13.51	Success	Yes
supportcase26	437	N.A.	Out of Memory	N.A.	115	441.93	200.76	642.69	Success	Yes
rlp1	462	N.A.	Out of Memory	N.A.	0	12.09	0.00	12.09	Success	Yes
neos-3611447-jijia	473	N.A.	Out of Memory	N.A.	0	17.70	0.00	17.70	Success	Yes
k16x240b	481	N.A.	Out of Memory	N.A.	0	17.60	0.00	17.60	Success	Yes
nexp-50-20-1-1	491	N.A.	Out of Memory	N.A.	0	21.41	0.00	21.41	Success	Yes
pigeon-10	491	N.A.	Out of Memory	N.A.	230	480.00	229.21	709.21	Success	Yes
ran12x21	505	N.A.	Out of Memory	N.A.	0	22.61	0.00	22.61	Success	Yes
ran14x18-disj-8	505	N.A.	Out of Memory	N.A.	0	46.34	0.00	46.34	Success	Yes

Table 3: Algorithm performance on the Shor's relaxation.

Name	The standard FRA				The primal FRA					
	Size	Time	Solver Status	Slater condition	Size	T_1	T_2	Time	Solver Status	Slater condition
neos-5192052-neckar	25	0.02	Success	Yes	0	0.09	0.00	0.09	Success	Yes
gr4x6	25	0.03	Success	Yes	0	0.21	0.00	0.21	Success	Yes
markshare-4-0	31	0.03	Success	Yes	0	0.11	0.00	0.11	Success	Yes
bienst1	29	0.04	Success	Yes	0	0.18	0.00	0.18	Success	Yes
markshare-5-0	41	0.07	Success	Yes	0	0.17	0.00	0.17	Success	Yes
bienst2	36	0.07	Success	Yes	0	0.21	0.00	0.21	Success	Yes
nsa	37	0.08	Success	Yes	0	0.41	0.00	0.41	Success	Yes
markshare1	51	0.12	Success	Yes	0	0.22	0.00	0.22	Success	Yes
osorio-cta	8	0.14	Success	Yes	0	0.29	0.00	0.29	Success	Yes
neos5	54	0.15	Success	Yes	0	0.37	0.00	0.37	Success	Yes
pk1	56	0.17	Success	Yes	0	0.24	0.00	0.24	Success	Yes
qiu	49	0.17	Success	Yes	0	0.60	0.00	0.60	Success	Yes
newdano	57	0.24	Success	Yes	0	0.31	0.00	0.31	Success	Yes
markshare2	61	0.25	Success	Yes	0	0.29	0.00	0.29	Success	Yes
gsvm2rl3	61	0.28	Success	Yes	0	0.81	0.00	0.81	Success	Yes
fastxgemm-n2r7s4t1	57	0.30	Failed	N.A.	0	7.21	0.00	7.21	Success	Yes
neos-3754480-nidda	51	0.34	Stall	Yes	0	0.28	0.00	0.28	Success	Yes
fastxgemm-n2r6s0t2	49	0.42	Failed	N.A.	0	3.27	0.00	3.27	Success	Yes
istanbul-no-cutoff	31	0.45	Success	Yes	1	25.42	0.15	25.58	Success	Yes
neos-2978193-inde	65	0.50	Success	Yes	0	1.76	0.00	1.76	Success	Yes
danoint	57	0.55	Success	Yes	0	46.53	0.00	46.53	Success	Yes
neos-3610051-istra	77	0.83	Success	Yes	0	0.83	0.00	0.83	Success	Yes
neos-3610173-itata	78	0.96	Success	Yes	0	0.64	0.00	0.64	Success	Yes
neos-5100895-inster	57	0.97	Failed	N.A.	34	61.82	0.74	62.57	Stall	Yes
dcmulti	76	1.07	Stall	Yes	6	1.08	0.02	1.10	Stall	Yes
neos-807639	81	1.25	Success	Yes	51	10.28	0.37	10.65	Success	Yes
neos-3610040-iskar	86	1.26	Success	Yes	0	0.78	0.00	0.78	Success	Yes
neos-3611447-jijia	86	1.31	Success	Yes	0	0.74	0.00	0.74	Success	Yes
b-ball	89	1.32	Success	Yes	0	0.60	0.00	0.60	Success	Yes
neos-3611689-kaihu	89	1.57	Success	Yes	0	0.80	0.00	0.80	Success	Yes
gsvm2rl5	101	2.04	Success	Yes	0	2.20	0.00	2.20	Success	Yes
rmatr100-p10	101	2.41	Success	Yes	0	2.99	0.00	2.99	Success	Yes
rmatr100-p5	101	2.52	Success	Yes	0	4.43	0.00	4.43	Success	Yes
pg	101	2.61	Success	Yes	0	3.33	0.00	3.33	Success	Yes
pg5-34	101	2.63	Success	Yes	0	1.18	0.00	1.18	Success	Yes
control30-3-2-3	91	2.79	Failed	N.A.	0	0.94	0.00	0.94	Success	Yes
mod011	97	2.92	Stall	Yes	0	4.84	0.00	4.84	Success	Yes
app3	101	3.79	Stall	Yes	4	3.93	0.04	3.97	Stall	Yes
milo-v13-4-3d-3-0	121	5.70	Success	Yes	94	130.06	10.84	140.90	Success	Yes
assign1-5-8	131	6.24	Success	Yes	0	0.86	0.00	0.86	Success	Yes
neos-1396125	130	7.52	Success	Yes	45	132.02	5.60	137.62	Success	Yes
neos-3660371-kurow	145	9.08	Success	Yes	44	44.62	1.89	46.50	Success	Yes
mas74	151	10.76	Success	Yes	0	0.86	0.00	0.86	Success	Yes
v150d30-2hopcds	151	12.77	Success	No	0	12.48	0.00	12.48	Success	Yes
prod1	150	12.77	Success	Yes	25	9.69	0.21	9.89	Success	Yes

Table 4: Algorithm performance on the variant of Shor's relaxation (1).

Name	The standard FRA				The primal FRA					
	Size	Time	Solver Status	Slater condition	Size	T_1	T_2	Time	Solver Status	Slater condition
mas76	151	12.87	Success	Yes	0	0.91	0.00	0.91	Success	Yes
neos-1430701	157	15.42	Success	Yes	0	0.98	0.00	0.98	Success	Yes
milo-v13-4-3d-4-0	161	17.15	Success	Yes	127	170.50	33.47	203.97	Success	Yes
ran13x13	170	21.13	Success	Yes	0	1.66	0.00	1.66	Success	Yes
neos-5140963-mincio	184	28.53	Success	Yes	28	19.68	1.39	21.07	Success	Yes
binkar10-1	171	29.58	Success	Yes	0	6.02	0.00	6.02	Success	Yes
neos-480878	190	40.88	Success	Yes	0	4.06	0.00	4.06	Success	Yes
gsvm2rl9	201	43.17	Success	Yes	0	19.21	0.00	19.21	Success	Yes
neos-3372571-onahau	186	50.14	Success	Yes	75	129.53	2.55	132.08	Success	Yes
prod2	201	50.34	Success	Yes	24	26.39	0.58	26.97	Success	Yes
2club200v15p5scn	201	51.92	Success	Yes	0	12.79	0.00	12.79	Success	Yes
a2c1s1	193	53.84	Success	Yes	0	51.63	0.00	51.63	Success	Yes
rmatr200-p20	201	60.34	Success	Yes	0	44.75	0.00	44.75	Success	Yes
p0201	202	61.80	Stall	Yes	62	2.60	6.84	9.44	Stall	Yes
glass-sc	215	61.81	Success	Yes	0	13.73	0.00	13.73	Success	Yes
a1c1s1	193	63.23	Stall	Yes	0	98.01	0.00	98.01	Success	Yes
van	193	63.39	Success	Yes	0	192.59	0.00	192.59	Success	Yes
mad	201	64.73	Failed	N.A.	0	2.09	0.00	2.09	Success	Yes
iis-glass-cov	215	65.10	Success	Yes	0	13.49	0.00	13.49	Success	Yes
bg512142	241	107.76	Success	No	0	15.50	0.00	15.50	Success	Yes
k16x240b	241	111.30	Success	Yes	0	13.37	0.00	13.37	Success	Yes
exp-1-500-5-5	251	141.01	Success	Yes	0	4.73	0.00	4.73	Success	Yes
neos-2629914-sudost	257	144.15	Success	Yes	0	10.33	0.00	10.33	Success	Yes
ran12x21	253	146.42	Success	Yes	0	4.09	0.00	4.09	Success	Yes
nexp-50-20-1-1	246	149.11	Success	No	0	3.99	0.00	3.99	Success	Yes
ran14x18-disj-8	253	161.37	Success	Yes	0	6.32	0.00	6.32	Success	Yes
bc1	253	199.75	Failed	N.A.	0	134.63	0.00	134.63	Success	Yes
pigeon-08	273	220.45	Success	Yes	152	20.96	26.41	47.37	Failed	N.A.
supportcase14	305	348.73	Success	No	201	7.23	116.14	123.37	Success	No
supportcase16	320	444.53	Success	No	217	8.05	139.63	147.69	Success	No
probportfolio	301	457.84	Success	Yes	165	278.27	44.37	322.64	Success	Yes
b2c1s1	289	466.11	Success	Yes	0	180.33	0.00	180.33	Success	Yes
neos17	301	493.37	Success	Yes	0	5.45	0.00	5.45	Success	Yes
iis-hc-cov	298	508.35	Failed	N.A.	0	18.16	0.00	18.16	Success	Yes
b1c1s1	289	525.80	Stall	Yes	0	169.96	0.00	169.96	Success	Yes
neos-3665875-lesum	321	597.45	Success	Yes	157	325.58	177.87	503.46	Success	Yes
sp150x300d	301	847.32	Stall	Yes	29	39.87	0.54	40.40	Stall	Yes
neos-5188808-nattai	289	963.47	Stall	Yes	188	292.83	550.53	843.37	Stall	Yes
r50x360	361	1000.67	Success	Yes	1	19.99	0.23	20.22	Success	Yes
tr12-30	361	1052.19	Success	Yes	8	32.87	0.29	33.16	Success	Yes
neos-1442119	365	1109.93	Success	Yes	0	7.24	0.00	7.24	Success	Yes
uct-subprob	380	2093.29	Stall	Yes	0	8.13	0.00	8.13	Success	Yes
supportcase26	397	N.A.	Out of Memory	N.A.	55	400.32	24.59	424.92	Success	Yes
pigeon-10	401	N.A.	Out of Memory	N.A.	230	177.04	103.54	280.58	Success	Yes
aflow30a	422	N.A.	Out of Memory	N.A.	0	87.06	0.00	87.06	Success	Yes

Table 5: Algorithm performance on the variant of Shor's relaxation (2).

References

[1] Jon Borwein and Henry Wolkowicz. "Regularizing the abstract convex program". In: *Journal of Mathematical Analysis and Applications* 83.2 (1981), pp. 495–530.

[2] Jon M Borwein and Henry Wolkowicz. "Facial reduction for a cone-convex programming problem". In: *Journal of the Australian Mathematical Society* 30.3 (1981), pp. 369–380.

[3] Rainer E Burkard. "Quadratic assignment problems". In: *European Journal of Operational Research* 15.3 (1984), pp. 283–289.

[4] Gerd Finke, Rainer E Burkard, and Franz Rendl. "Quadratic assignment problems". In: *North-Holland Mathematics Studies.* Vol. 132. Elsevier, 1987, pp. 61–82.

[5] Hanif D Sherali and Warren P Adams. "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems". In: *SIAM Journal on Discrete Mathematics* 3.3 (1990), pp. 411–430.

[6] Hanif D Sherali and Warren P Adams. "A hierarchy of relaxations and convex hull characterizations for mixed-integer zero—one programming problems". In: *Discrete Applied Mathematics* 52.1 (1994), pp. 83–106.

[7] Erling D Andersen and Knud D Andersen. "Presolving in linear programming". In: *Mathematical Programming* 71 (1995), pp. 221–245.

[8] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[9] Gengui Zhout and Mitsuo Gen. "An effective genetic algorithm approach to the quadratic minimum spanning tree problem". In: *Computers & Operations Research* 25.3 (1998), pp. 229–237.

[10] Henry Wolkowicz and Qing Zhao. "Semidefinite programming relaxations for the graph partitioning problem". In: *Discrete Applied Mathematics* 96 (1999), pp. 461–479.

[11] Levent Tunçel. "On the Slater condition for the SDP relaxations of nonconvex sets". In: *Operations Research Letters* 29.4 (2001), pp. 181–186.

[12] Samuel Burer and Renato DC Monteiro. "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization". In: *Mathematical programming* 95.2 (2003), pp. 329–357.

[13] Johan Lofberg. "YALMIP: A toolbox for modeling and optimization in MATLAB". In: *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508).* IEEE. 2004, pp. 284–289.

[14] Samuel Burer and Renato DC Monteiro. "Local minima and convergence in low-rank semidefinite programming". In: *Mathematical programming* 103.3 (2005), pp. 427–444.

[15] Pratik Biswas et al. "Semidefinite programming approaches for sensor network localization with noisy distance measurements". In: *IEEE transactions on automation science and engineering* 3.4 (2006), pp. 360–371.

[16] Sunyoung Kim, Masakazu Kojima, and Hayato Waki. “Exploiting sparsity in SDP relaxation for sensor network localization”. In: *SIAM Journal on Optimization* 20.1 (2009), pp. 192–215.

[17] Nathan Krislock and Henry Wolkowicz. “Explicit sensor network localization using semidefinite representations and facial reductions”. In: *SIAM Journal on Optimization* 20.5 (2010), pp. 2679–2708.

[18] Temel Öncan and Abraham P Punnen. “The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm”. In: *Computers & Operations Research* 37.10 (2010), pp. 1762–1773.

[19] Shyam Sundar and Alok Singh. “A swarm intelligence approach to the quadratic minimum spanning tree problem”. In: *Information Sciences* 180.17 (2010), pp. 3182–3191.

[20] Roberto Cordone and Gianluca Passeri. “Solving the quadratic minimum spanning tree problem”. In: *Applied Mathematics and Computation* 218.23 (2012), pp. 11597–11612.

[21] Yuen-Lam Cheung, Simon Schurr, and Henry Wolkowicz. “Preprocessing and regularization for degenerate semidefinite programs”. In: *Computational and Analytical Mathematics: In Honor of Jonathan Borwein’s 60th Birthday.* Springer. 2013, pp. 251–303.

[22] Gábor Pataki. “Strong duality in conic linear programming: facial reduction and extended duals”. In: *Computational and Analytical Mathematics: In Honor of Jonathan Borwein’s 60th Birthday.* Springer. 2013, pp. 613–634.

[23] Hayato Waki and Masakazu Muramatsu. “Facial reduction algorithms for conic optimization problems”. In: *Journal of Optimization Theory and Applications* 158 (2013), pp. 188–215.

[24] Dilson Lucas Pereira, Michel Gendreau, and Alexandre Salles Da Cunha. “Lower bounds and exact algorithms for the quadratic minimum spanning tree problem”. In: *Computers & Operations Research* 63 (2015), pp. 149–160.

[25] Dimitris Bertsimas, Angela King, and Rahul Mazumder. “Best subset selection via a modern optimization lens”. In: (2016).

[26] Henrik A Friberg. “Facial reduction heuristics and the motivational example of mixed-integer conic optimization”. In: *Optimization online* 5 (2016), p. 14.

[27] Hamza Fawzi and Omar Fawzi. “Efficient optimization of the quantum relative entropy”. In: *Journal of Physics A: Mathematical and Theoretical* 51.15 (2018), p. 154003.

[28] Minghui Liu and Gábor Pataki. “Exact duals and short certificates of infeasibility and weak infeasibility in conic linear programming”. In: *Mathematical Programming* 167 (2018), pp. 435–480.

[29] Frank Permenter and Pablo Parrilo. “Partial facial reduction: simplified, equivalent SDPs via approximations of the PSD cone”. In: *Mathematical Programming* 171 (2018), pp. 1–54.

[30] Mosek ApS. “Mosek optimization toolbox for MATLAB”. In: *User’s Guide and Reference Manual, version* 4 (2019).

[31] Dimitris Bertsimas and Jack Dunn. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC Waltham, 2019.

[32] Yuzixuan Zhu, Gábor Pataki, and Quoc Tran-Dinh. "Sieve-SDP: a simple facial reduction algorithm to preprocess semidefinite programs". In: *Mathematical Programming Computation* 11.3 (2019), pp. 503–586.

[33] Kurt M Anstreicher. "Testing copositivity via mixed–integer linear programming". In: *Linear Algebra and Its Applications* 609 (2021), pp. 218–230.

[34] Ambros Gleixner et al. "MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library". In: *Mathematical Programming Computation* (2021). DOI: 10.1007/s12532-020-00194-3. URL: https://doi.org/10.1007/s12532-020-00194-3.

[35] Hao Hu et al. "Robust interior point method for quantum key distribution rate computation". In: *Quantum* 6 (2022), p. 792.

[36] Angelika Wiegele and Shudian Zhao. "SDP-based bounds for graph partition via extended ADMM". In: *Computational Optimization and Applications* 82.1 (2022), pp. 251–291.

[37] Mateus Araújo et al. "Quantum key distribution rates from semidefinite programming". In: *Quantum* 7 (2023), p. 1019.

[38] Hamza Fawzi and James Saunderson. "Optimal self-concordant barriers for quantum relative entropies". In: *SIAM Journal on Optimization* 33.4 (2023), pp. 2858–2884.

[39] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com.

[40] Hao Hu and Boshi Yang. "Affine FR: an effective facial reduction algorithm for semidefinite relaxations of combinatorial problems". 2023.

[41] Frank de Meijer et al. "Spanning and splitting: Integer semidefinite programming for the quadratic minimum spanning tree problem". In: *arXiv preprint arXiv:2410.04997* (2024).

[42] Renata Sotirov and Zoe Verchére. "The quadratic minimum spanning tree problem: lower bounds via extended formulations". In: *Vietnam Journal of Mathematics* (2024), pp. 1–22.

[43] Armin Tavakoli et al. "Semidefinite programming relaxations for quantum correlations". In: *Reviews of Modern Physics* 96.4 (2024), p. 045006.

[44] Mehdi Karimi and Levent Tuncel. "Efficient implementation of interior-point methods for quantum relative entropy". In: *INFORMS Journal on Computing* 37.1 (2025), pp. 3–21.

A Further Results on Solving the SDP Relaxation

The primary focus of this work is the preprocessing stage—specifically, solving the FR auxiliary problem efficiently. The effectiveness of the proposed primal FRA in this regard has already been demonstrated in the previous section. In this subsection, we present supplementary numerical results comparing the DNN relaxation and its facially reduced formulation.

While the advantages of solving facially reduced SDP relaxations are well established in the literature, this approach is not without criticism. Key concerns include potential loss of sparsity and the introduction of numerical issues during the reformulation process. These challenges often necessitate problem-specific strategies and fine-tuning to fully realize the benefits of facial reduction. Indeed, most numerical experiments on SDP methods are conducted on instances where SDP formulations are known to perform well and can be regularized into stable forms—e.g., the Max-Cut problem, maximum stable set, and quadratic assignment problem. A detailed treatment of such implementations for the benchmark instances in MIPLIB lies beyond the scope of this paper.

Nevertheless, the supplementary results in Table 6 empirically confirm that facial reduction enhances numerical stability in practice. When solving the original DNN relaxations, the solver encountered numerical difficulties or failed to return reliable solutions for all eight instances. In contrast, the facially reduced DNN formulations exhibited greater robustness, successfully solving four of these cases. Although the problem size is reduced after facial reduction, the total runtime can sometimes increase due to the aforementioned loss of sparsity. Addressing this trade-off often requires customized solutions depending on the problem structure.

Table 6 also confirms that MIQP problems are significantly more expensive to solve than MILP problems. Thus, when applying the SDP approach to solve an MIQP, it is reasonable to rely on MILP solvers to help regularize the SDP relaxation. This is the key idea behind the primal FRA.

Each row in Table 6 contains the following information:

1. **Size**: the order of the matrix variable in the DNN relaxation.
2. **L.B.**: the lower bound obtained from solving the relaxation. In theory, the original relaxation and its facially reduced counterpart are equivalent and should produce the same bound, assuming both are solved correctly. In practice, however, especially for the original formulation, failure to satisfy Slater's condition may impair bound quality.
3. **Time**: the computational time required to solve the relaxation.
4. **Solver Status**: solver status as reported by MOSEK, using the same categorization as in Table 2—*Success*, *Stall*, or *Fail*.
5. **U.B.**: the best upper bound obtained by Gurobi within a 600-second time limit.
6. **Time (Gurobi)**: the computational time used by Gurobi.
7. **Status (Gurobi)**: the termination status reported by Gurobi.

	DNN Relaxation				Facially Reduced DNN Relaxation				Gurobi		
Name	Size	L.B.	Time	Status	Size	L.B.	Time	Status	U.B.	Time	Status
markshare-4-0	34	NA	0.14	Failed	31	30.46	1.33	Stall	65.85	91.29	Optimal
markshare-5-0	45	NA	0.22	Failed	41	NA	2.58	Failed	235.41	600.02	Time limit
gr4x6	48	2863.42	0.34	Stall	40	2863.83	0.77	Success	3022.17	1.39	Optimal
markshare1	62	NA	0.61	Failed	51	109.16	6.64	Stall	2097.72	924.47	Time limit
markshare2	74	NA	1.76	Failed	61	178.57	32.84	Stall	9850.66	970.81	Time limit
pk1	86	NA	2.58	Failed	72	146.58	18.54	Success	4293.04	1005.76	Time limit
b-ball	100	887.75	10.18	Stall	82	887.83	30.04	Success	986.56	978.20	Time limit
assign1-5-8	156	2076.92	70.88	Stall	127	NA	683.04	Failed	16614.17	1031.83	Time limit
neos-5192052-neckar	180	NA	100.61	Failed	161	NA	279.05	Failed	0.00	0.04	Optimal
neos-5140963-mincio	196	19.45	158.99	Stall	170	19.61	1782.07	Success	1546.53	2464.42	Time limit

Table 6: Comparison between the original DNN relaxation and its facially reduced formulation