HDSDP 1.0: Software for Semidefinite Programming

WENZHI GAO and DONGDONG GE, Shanghai University of Finance and Economics, China
YINYU YE, Stanford University, United States

CCS Concepts: • **Mathematics of computing → Solvers**;

Additional Key Words and Phrases: semidefinite programming, interior point methods, dual-scaling algorithm

ACM Reference Format:
Wenzhi Gao, Dongdong Ge, and Yinyu Ye. 2018. HDSDP 1.0: Software for Semidefinite Programming. 1, 1 (July 2018), 9 pages. https://doi.org/XXXXXXX.XXXXXXX

Abstract

HDSDP is a numerical software solving the semidefinite programming problems. The main framework of HDSDP resembles the dual-scaling interior point solver DSDP[Benson and Ye, 2008] and several new features, including a dual method based on the simplified homogeneous self-dual embedding, have been implemented. The embedding technique enhances stability of dual method and several new heuristics and computational techniques are designed to accelerate its convergence. HDSDP aims to show how dual-scaling algorithm benefits from the self-dual embedding and it is developed in parallel to DSDP5.8. Numerical experiments over several classical benchmark datasets exhibit its robustness and efficiency, and particularly its advantages on SDP instances featuring low-rank structure and sparsity. HDSDP is now freely available at https://github.com/COPT-Public/HDSDP.

1 INTRODUCTION

Semidefinite programming (SDP) is defined by

$$\begin{aligned} \min_{\mathbf{X}} \quad & \langle \mathbf{C}, \mathbf{X} \rangle \\ \text{subject to} \quad & \mathcal{A}\mathbf{X} = \mathbf{b} \\ & \mathbf{X} \in \mathbb{S}^n_+, \end{aligned} \tag{1}$$

where we study linear optimization subject to affine constraints over the cone of positive-semidefinite matrices. Due to its extensive modeling capability, SDP is employed by several communities including combinatorial optimization [Goemans and Williamson, 1995, Laurent and Rendl, 2005], dynamic systems [Vandenberghe and Boyd, 1996], sums of squares optimization [Laurent, 2009], quantum information [Hayashi, 2016], and distance geometry [Biswas and Ye, 2004, So and Ye, 2007]. We refer the interested readers to [Wolkowicz, 2005] for a more comprehensive review.

Authors' addresses: Wenzhi Gao, gwz@163.shufe.edu.cn; Dongdong Ge, ge.dongdong@mail.shufe.edu.cn, Shanghai University of Finance and Economics, Shanghai, Shanghai, China; Yinyu Ye, yyye@stanford.edu, Stanford University, 450 Serra Mall, Palo Alto, California, United States.

While SDP proves useful in practice, a fundamental issue is how to numerically solve it. Theoretically, SDP is a convex conic problem admitting polynomial-time algorithms, and for general SDPs, the interior point method (IPM) is known as a robust and efficient approach. Since the 1990s, high-performance SDP softwares based on the IPM have been developed, including DSDP [Benson and Ye, 2008], COPT [Ge et al., 2022], Mosek [ApS, 2019], Sedumi [Polik et al., 2007], SDPT3 [Toh et al., 2012], CSDP [Borchers, 2006], SDPA [Yamashita et al., 2012] and so forth. While most SDP codes are IPM-based, there are also several successful attempts using other methods (e.g., [Kocvara et al., 2006, Kwasniewicz and Glineur, 2021, Yang et al., 2015]) and a list of these SDP softwares is reviewed by Majumdar et al. [2020].

SDP solvers based on IPM variants enjoy competitive convergence behavior both theoretically and practically. Most softwares implement primal-dual path-following approaches using either infeasible start [Potra and Sheng, 1998b] or the embedding technique [Potra and Sheng, 1998a], with DSDP being an exception: DSDP adopts a dual IPM method based on the potential reduction framework developed by Benson et al. [1999]. Since its initial release [Benson et al., 2000], DSDP has gone a long way through several major updates, and finally evolved into one of the most popular SDP softwares [Benson and Ye, 2008]. To further improve DSDP, we make an important extension by incorporating the well-known homogeneous self-dual (HSD) embedding technique into the dual algorithm. This new implementation, named HDSDP, is presented in this manuscript.

The manuscript is organized as follows. **Section 2** describes the formulation of interest and some basic notations; **Section 3** reviews the dual-scaling algorithm and describes how embedding technique can be applied. In **Section 4** and **5**, we introduce the practical aspects for HDSDP, including software design and algorithm configuration. Last we present comprehensive numerical results on SDP problems.

2 FORMULATION AND NOTATIONS

HDSDP solves both primal and dual SDPs in standard form

$$(P) \quad \begin{array}{ll} \min_{\mathbf{X}} & \langle \mathbf{C}, \mathbf{X} \rangle \\ \text{subject to} & \langle \mathbf{A}_i, \mathbf{X} \rangle = \mathbf{b}, \quad i = 1, \dots, m \\ & \mathbf{X} \in \mathbb{S}^n_+ \end{array} \qquad (D) \quad \begin{array}{ll} \max_{\mathbf{y}, \mathbf{S}} & \mathbf{b}^\top \mathbf{y} \\ \text{subject to} & \sum_{i=1}^m \mathbf{A}_i y_i + \mathbf{S} = \mathbf{C} \\ & \mathbf{S} \in \mathbb{S}^n_+, \end{array}$$

where the problem data $\{\mathbf{A}_i\}, \mathbf{C}$ are $n \times n$ symmetric matrices ($\mathbb{S}^n$) and $\mathbf{b} \in \mathbb{R}^m$ is a real vector. Matrix inner product $\langle \cdot, \cdot \rangle$ is defined by $\langle \mathbf{A}, \mathbf{X} \rangle := \sum_{i,j} c_{ij} x_{ij}$ and $\mathbb{S}^n_+$ denotes the cone of positive semidefinite matrices. For brevity we use $\mathbf{X} \succeq \mathbf{0}$ to denote the relation $\mathbf{X} \in \mathbb{S}^n_+$, and the linear map $\mathcal{A} : \mathbb{S}^n \to \mathbb{R}^m$ together with its adjoint $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{S}^n$ is respectively defined by $\mathcal{A}\mathbf{X} := (\langle \mathbf{A}, \mathbf{X}_1 \rangle, \dots, \langle \mathbf{A}_m, \mathbf{X}_m \rangle)^\top$ and $\mathcal{A}^*\mathbf{y} := \sum_{i=1}^m \mathbf{A}_i y_i$. $\|\mathbf{A}\|_F := \sqrt{\sum_{ij} a_{ij}^2}$ denotes the matrix Frobenius norm. With the above notations, we rewrite the primal and dual problems by

$$(P) \quad \begin{array}{ll} \min_{\mathbf{X}} & \langle \mathbf{C}, \mathbf{X} \rangle \\ \text{subject to} & \mathcal{A}\mathbf{X} = \mathbf{b}, \\ & \mathbf{X} \succeq \mathbf{0} \end{array} \qquad (D) \quad \begin{array}{ll} \max_{\mathbf{y}, \mathbf{S}} & \mathbf{b}^\top \mathbf{y} \\ \text{subject to} & \mathcal{A}^*\mathbf{y} + \mathbf{S} = \mathbf{C} \\ & \mathbf{S} \succeq \mathbf{0}. \end{array}$$

and primal and dual feasible regions are respectively

$$\mathcal{F}(P) := \{\mathbf{X} : \mathcal{A}\mathbf{X} = \mathbf{b}, \mathbf{X} \succeq \mathbf{0}\} \qquad \mathcal{F}(D) := \left\{(\mathbf{y}, \mathbf{S}) : \mathcal{A}^*\mathbf{y} + \mathbf{S} = \mathbf{C}, \mathbf{S} \succeq \mathbf{0}\right\}$$

$\mathbf{X} \in \mathcal{F}(P)$ is primal feasible and $(\mathbf{y}, \mathbf{S}) \in \mathcal{F}(D)$ is dual feasible. The interior of the regions are denoted by $\mathcal{F}^0(P), \mathcal{F}^0(D)$ and a solution in $\mathcal{F}^0$ is an interior point solution. HDSDP implements a dual method solving (P) and (D) through the self-dual embedding technique.

3 DUAL AND HOMOGENEOUS DUAL SCALING ALGORITHM

In this section, we briefly review the dual-scaling algorithm in DSDP, and give its interpretation through Newton's method on the KKT system. Then we show how it can be naturally generalized to the embedding formation.

3.1 Dual-scaling Algorithm

Dual-scaling method [Benson et al., 1999] works under three conditions. **1)** the data $\{\mathbf{A}_i\}$ are linearly independent. **2)** Both (P) and (D) admit an interior point solution. **3)** an interior dual feasible solution $\left(\mathbf{y}^0, \mathbf{S}^0\right) \in \mathcal{F}^0(D)$ is known. The first two conditions imply strong duality and the existence of an optimal primal-dual solution pair $(\mathbf{X}^*, \mathbf{y}^*, \mathbf{S}^*)$ satisfying complementarity $\langle \mathbf{X}^*, \mathbf{S}^* \rangle = 0$. Also, the central path $C(\mu) := \left\{(\mathbf{X}, \mathbf{y}, \mathbf{S}) \in \mathcal{F}^0(P) \times \mathcal{F}^0(D) : \mathbf{XS} = \mu\mathbf{I}\right\}$ is well-defined, which serves as a foundation of path-following approaches. Given a centrality parameter μ, dual method starts from $(\mathbf{y}, \mathbf{S}) \in \mathcal{F}^0(D)$ and takes Newton's step towards the perturbed KKT system $\mathcal{A}\mathbf{X} = \mathbf{b}, \mathcal{A}^*\mathbf{y} + \mathbf{S} = \mathbf{C}$ and $\mathbf{XS} = \mu\mathbf{I}$

$$\begin{aligned} \mathcal{A}\left(\mathbf{X} + \Delta\mathbf{X}\right) &= \mathbf{b} \\ \mathcal{A}^*\Delta\mathbf{y} + \Delta\mathbf{S} &= \mathbf{0} \\ \mu\mathbf{S}^{-1}\Delta\mathbf{S}\mathbf{S}^{-1} + \Delta\mathbf{X} &= \mu\mathbf{S}^{-1} - \mathbf{X}, \end{aligned} \tag{2}$$

where the last relation linearizes $\mathbf{X} = \mu\mathbf{S}^{-1}$ instead of $\mathbf{XS} = \mu\mathbf{I}$ and $\mathbf{S}^{-1}$ is known as a scaling matrix. By geometrically driving μ to 0, dual-scaling eliminates (primal) infeasibility, approaches optimality, and finally solves the problem to some ε-optimal solution $(\mathbf{y}_\varepsilon, \mathbf{S}_\varepsilon) \in \mathcal{F}^0(D)$ such that $\langle \mathbf{b}, \mathbf{y}_\varepsilon \rangle \geq \langle \mathbf{b}, \mathbf{y}^* \rangle - \varepsilon$. Theory of dual potential reduction shows an ε-optimal solution can be obtained in $O(\log(1/\varepsilon))$ iterations ignoring dimension dependent constants.

An important feature of dual-scaling is that $\mathbf{X}$ and $\Delta\mathbf{X}$ vanish in the Schur complement of (2)

$$\begin{pmatrix} \left\langle \mathbf{A}_1, \mathbf{S}^{-1}\mathbf{A}_1\mathbf{S}^{-1} \right\rangle & \cdots & \left\langle \mathbf{A}_1, \mathbf{S}^{-1}\mathbf{A}_m\mathbf{S}^{-1} \right\rangle \\ \vdots & \ddots & \vdots \\ \left\langle \mathbf{A}_m, \mathbf{S}^{-1}\mathbf{A}_1\mathbf{S}^{-1} \right\rangle & \cdots & \left\langle \mathbf{A}_m, \mathbf{S}^{-1}\mathbf{A}_m\mathbf{S}^{-1} \right\rangle \end{pmatrix} \Delta\mathbf{y} = \frac{1}{\mu}\mathbf{b} - \mathcal{A}\mathbf{S}^{-1}, \tag{3}$$

and the dual algorithm thereby avoids explicit reference to the primal variable $\mathbf{X}$. For brevity we sometimes denote the left-hand side matrix in (3) by $\mathbf{M}$. If $\{\mathbf{A}_i\}, \mathbf{C}$ are sparse, any feasible dual slack $\mathbf{S} = \mathbf{C} - \mathcal{A}^*\mathbf{y}$ inherits the aggregated sparsity pattern from the data and it is computationally cheaper to iterate in the dual space. Another feature of dual-scaling is the availability of primal solution by solving a projection subproblem at the end of the algorithm [Benson and Ye, 2008]. The aforementioned properties characterize the behavior of the dual-scaling method.

However, the desirable theoretical properties of the dual method is not free. First, an initial dual feasible solution is needed, while obtaining such a solution is often as difficult as solving the original problem. Second, due to a lack of information from the primal space, the dual-scaling has to be properly guided to avoid deviating from the central path. Last, dual-scaling linearizes the highly nonlinear relation $\mathbf{X} = \mu\mathbf{S}^{-1}$ and this imposes strict constraint on the step length towards the Newton's direction. To overcome the aforementioned difficulties, DSDP introduces artificial variables with big-M penalty to ensure nonempty interior and a trivial dual feasible solution. Moreover, a potential function is introduced to guide the dual iterations. These solutions work well in practice and enhance DSDP greatly. For a complete description of the theoretical aspects of DSDP and its delicate implementation, we refer the interested readers to [Benson and Ye, 2008, Benson et al., 2000].

Although DSDP proves efficient in real practice, the big-M method requires prior estimation of the optimal solution to avoid losing optimality. Also, a large penalty often leads to numerical difficulties and misclassification of infeasible problems when the problem is ill-conditioned. Taking into account of In this paper, we propose to leverage the well-known HSD embedding.

3.2 Dual-scaling Algorithm using Embedding Technique

Now we are ready to introduce the idea behind HDSDP. Given a standard form SDP, its homogeneous self-dual model is a skew symmetric system that contains the original problem data, whose non-trivial interior point solution certificates primal-dual feasibility. HDSDP adopts the following simplified embedding system [Potra and Sheng, 1998a].

$$\begin{aligned} \mathcal{A}\mathbf{X} - \mathbf{b}\tau &= \mathbf{0} \\ -\mathcal{A}^*\mathbf{y} + \mathbf{C}\tau - \mathbf{S} &= \mathbf{0} \\ \mathbf{b}^\top\mathbf{y} - \langle \mathbf{C}, \mathbf{X}\rangle - \kappa &= 0 \\ \mathbf{X}, \mathbf{S} \geq 0, \kappa, \tau &\geq 0, \end{aligned} \tag{4}$$

where $\kappa, \tau \geq 0$ are known as homogenizing variables for infeasibility detection. Given barrier parameter μ, we similarly define the central path

$$\begin{aligned} \mathcal{A}\mathbf{X} - \mathbf{b}\tau &= \mathbf{0} \\ -\mathcal{A}^*\mathbf{y} + \mathbf{C}\tau - \mathbf{S} &= \mathbf{0} \\ \mathbf{b}^\top\mathbf{y} - \langle \mathbf{C}, \mathbf{X}\rangle - \kappa &= 0 \\ \mathbf{X}\mathbf{S} = \mu\mathbf{I}, \kappa\tau &= \mu \\ \mathbf{X}, \mathbf{S} \geq 0, \kappa, \tau &\geq 0. \end{aligned} \tag{5}$$

Here $(\mathbf{y}, \mathbf{S}, \tau)$ are jointly considered as dual variables. Given an (infeasible) dual point $(\mathbf{y}, \mathbf{S}, \tau)$ such that $-\mathcal{A}^*\mathbf{y}+\mathbf{C}\tau-\mathbf{S} = \mathbf{R}$, HDSDP selects a damping factor $\gamma \in (0, 1]$ and takes Newton's step towards

$$\begin{aligned} \mathcal{A}(\mathbf{X} + \Delta\mathbf{X}) - \mathbf{b}(\tau + \Delta\tau) &= \mathbf{0} \\ -\mathcal{A}^*(\mathbf{y} + \Delta\mathbf{y} + \mathbf{C}(\tau + \Delta\tau) - (\mathbf{S} + \Delta\mathbf{S}) &= -\gamma\mathbf{R} \\ \mu\mathbf{S}^{-1}\Delta\mathbf{S}\mathbf{S}^{-1} + \Delta\mathbf{X} &= \mu\mathbf{S}^{-1} - \mathbf{X}, \\ \mu\tau^{-2}\Delta\tau + \Delta\kappa &= \mu\tau^{-1} - \kappa, \end{aligned}$$

where, as in DSDP, we modify (5) and linearize $\mathbf{X} = \mu\mathbf{S}^{-1}$ and $\kappa = \mu\tau^{-1}$. We note that the damping factor can be chosen after the directions are computed and for simplicity we set $\gamma = 0$. Then the Newton's direction $(\Delta\mathbf{y}, \Delta\tau)$ is computed through the following Schur complement.

$$\begin{pmatrix} \mu\mathbf{M} & -\mathbf{b} - \mu\mathcal{A}\mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1} \\ -\mathbf{b} + \mu\mathcal{A}\mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1} & -\mu(\langle \mathbf{C}, \mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1}\rangle + \tau^{-2}) \end{pmatrix} \begin{pmatrix} \Delta\mathbf{y} \\ \Delta\tau \end{pmatrix} = \begin{pmatrix} \mathbf{b}\tau \\ \mathbf{b}^\top\mathbf{y} - \mu\tau^{-1} \end{pmatrix} - \mu \begin{pmatrix} \mathcal{A}\mathbf{S}^{-1} \\ \langle \mathbf{C}, \mathbf{S}^{-1}\rangle \end{pmatrix} + \mu \begin{pmatrix} \mathcal{A}\mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1} \\ \langle \mathbf{C}, \mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1}\rangle \end{pmatrix}$$

In practice, HDSDP solves $\Delta\mathbf{y}_1 := \mathbf{M}^{-1}\mathbf{b}, \Delta\mathbf{y}_2 := \mathbf{M}^{-1}\mathcal{A}\mathbf{S}^{-1}, \Delta\mathbf{y}_3 := \mathbf{M}^{-1}\mathcal{A}\mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1}, \Delta\mathbf{y}_4 = \mathbf{M}^{-1}\mathcal{A}\mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1}$, plugs $\Delta\mathbf{y} = \frac{\tau+\Delta\tau}{\mu}\Delta\mathbf{y}_1 - \Delta\mathbf{y}_2 + \Delta\mathbf{y}_3 + \Delta\mathbf{y}_4\Delta\tau$ to get $\Delta\tau$ and finally assembles $\Delta\mathbf{y}$.

With the above relations, $\mathbf{X}(\mu) := \mu\mathbf{S}^{-1}\left(\mathbf{S} - \mathbf{R} + \mathcal{A}^*\Delta\mathbf{y} - \mathbf{C}\Delta\tau\right)\mathbf{S}^{-1}$ satisfies $\mathcal{A}\mathbf{X}(\mu) - \mathbf{b}(\tau + \Delta\tau) = \mathbf{0}$ and $\mathbf{X}(\mu) \succeq \mathbf{0}$ if and only if $-\mathcal{A}^*\left(\mathbf{y} - \Delta\mathbf{y}\right) + \mathbf{C}(\tau - \Delta\tau) - 2\mathbf{R} \succeq \mathbf{0}$. When $-\mathcal{A}^*\left(\mathbf{y} - \Delta\mathbf{y}\right) + \mathbf{C}(\tau - \Delta\tau) - 2\mathbf{R}$ is positive definite, an objective bound follows by

$$\begin{aligned}
\bar{z} = & \langle \mathbf{C}\tau, \mathbf{X}(\mu)\rangle \\
= & \left\langle \mathbf{R} + \mathbf{S} + \mathcal{A}^*\mathbf{y}, \mu\mathbf{S}^{-1}\left(\mathbf{S} - \mathbf{R} + \mathcal{A}^*\Delta\mathbf{y} - \mathbf{C}\Delta\tau\right)\mathbf{S}^{-1}\right\rangle \\
= & (\tau + \Delta\tau)\mathbf{b}^\top\mathbf{y} + n\mu + (\mathcal{A}\mathbf{S}^{-1} + \mathcal{A}\mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1})^\top\Delta\mathbf{y} + \mu(\langle \mathbf{C}, \mathbf{S}^{-1}\rangle + \langle \mathbf{C}, \mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1}\rangle)\Delta\tau
\end{aligned}$$

and dividing both sides by τ. Alternatively HDSDP extracts a bound from the projection problem

$$\begin{aligned}
\min_{\mathbf{X}} \quad & \|\mathbf{S}^{1/2}\mathbf{X}\mathbf{S}^{1/2} - \mu\mathbf{I}\|_F^2 \\
\text{subject to} \quad & \mathcal{A}\mathbf{X} = \mathbf{b}\tau,
\end{aligned}$$

whose optimal solution is $\mathbf{X}'(\mu) := \mu\mathbf{S}^{-1}\left(\mathbf{C}\tau - \mathcal{A}^*\left(\mathbf{y} - \Delta'\mathbf{y}\right) - \mathbf{R}\right)\mathbf{S}^{-1}$, where $\Delta\mathbf{y}' = \frac{\tau}{\mu}\Delta\mathbf{y}_1 - \Delta\mathbf{y}_2$ and

$$z' = \langle \mathbf{C}\tau, \mathbf{X}'(\mu)\rangle = \mu\{\langle \mathbf{R}, \mathbf{S}^{-1}\rangle + (\mathcal{A}\mathbf{S}^{-1}\mathbf{R}_y\mathbf{S}^{-1} + \mathcal{A}\mathbf{S}^{-1})^\top\Delta'\mathbf{y} + n\} + \tau\mathbf{b}^\top\mathbf{y}.$$

In practice, when the explicit solution $\mathbf{X}(\mu)$ or $\mathbf{X}'(\mu)$ is needed, we can decompose $\mathbf{S}$ and solve two sets of linear systems to obtain them.

One major computationally intensive part in HDSDP is the setup of the Schur complement matrix $\mathbf{M}$. Since the objective $\mathbf{C}$ often does not enjoy the same structure as $\{\mathbf{A}_i\}$, the additional cost from $\mathcal{A}\mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1}$ and $\left\langle \mathbf{C}, \mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1}\right\rangle$ calls for more efficient techniques to set up $\mathbf{M}$. In HDSDP, a permutation $\sigma(m)$ of the rows of $\mathbf{M}$ is generated heuristically to reduce the flops to set up $\mathbf{M}$ row by row.

$$\begin{pmatrix}
\left\langle \mathbf{A}_{\sigma(1)}, \mathbf{S}^{-1}\mathbf{A}_{\sigma(1)}\mathbf{S}^{-1}\right\rangle & \cdots & \left\langle \mathbf{A}_{\sigma(1)}, \mathbf{S}^{-1}\mathbf{A}_{\sigma(m)}\mathbf{S}^{-1}\right\rangle \\
 & \ddots & \vdots \\
 & & \left\langle \mathbf{A}_{\sigma(m)}, \mathbf{S}^{-1}\mathbf{A}_{\sigma(m)}\mathbf{S}^{-1}\right\rangle
\end{pmatrix}$$

Technique **M1** and **M2** are inherited from DSDP. They exploit the low-rank structure of the problem data and an eigen-decomposition of the problem data $\mathbf{A}_i = \sum_{r=1}^{\mathrm{rank}(\mathbf{A}_i)} \lambda_{ir}\mathbf{a}_{i,r}\mathbf{a}_{i,r}^\top$ needs to be computed at the beginning of the algorithm.

KKT Technique M1

(1) **Setup** $\mathbf{B}_{\sigma(i)} = \sum_{r=1}^{\mathrm{rank}(\mathbf{A}_{\sigma(i)})} \lambda_r\left(\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}\right)\left(\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}\right)^\top$.
(2) **Compute** $M_{\sigma(i)\sigma(j)} = \langle \mathbf{B}_{\sigma(i)}, \mathbf{A}_{\sigma(j)}\rangle$, for all $j \geq i$.

KKT Technique M2

(1) **Setup** $\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}, r = 1, \ldots, r_{\sigma(i)}$.
(2) **Compute** $M_{\sigma(i)\sigma(j)} = \sum_{r=1}^{\mathrm{rank}(\mathbf{A}_{\sigma(i)})}\left(\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}\right)^\top\mathbf{A}_{\sigma(j)}\left(\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}\right)$.

Technique **M3**, **M4** and **M5** exploit sparsity and need evaluation of $\mathbf{S}^{-1}$ [Fujisawa et al., 1997].

KKT Technique M3

(1) **Setup** $\mathbf{B}_{\sigma(i)} = \mathbf{S}^{-1}\mathbf{A}_{\sigma(i)}\mathbf{S}^{-1}$

(2) **Compute** $M_{\sigma(i)\sigma(j)} = \langle \mathbf{B}_{\sigma(i)}, \mathbf{A}_{\sigma(j)} \rangle$, for all $j \geq i$

KKT Technique M4

(1) **Setup** $\mathbf{B}_{\sigma(i)} = \mathbf{S}^{-1}\mathbf{A}_{\sigma(i)}$

(2) **Compute** $M_{\sigma(i)\sigma(j)} = \left\langle \mathbf{B}_{\sigma(i)}\mathbf{S}^{-1}, \mathbf{A}_{\sigma(j)} \right\rangle$, for all $j \geq i$

KKT Technique M5

(1) **Compute** $M_{\sigma(i)\sigma(j)} = \langle \mathbf{S}^{-1}\mathbf{A}_{\sigma(i)}\mathbf{S}^{-1}, \mathbf{A}_{\sigma(j)} \rangle$, for all $j \geq i$ directly.

At the beginning of the algorithm, HDSDP estimates the number of flops using each technique and each row of $\mathbf{M}$ is associated with the cheapest technique to minimize the overall computation cost. This is a major improvement over DSDP5.8 in the computational aspect and we leave the details to the later sections.

After setting up $\mathbf{M}$, conjugate gradient (CG) method is employed to solve the linear systems. The maximum number of iterations is chosen around $50/m$ and is heuristically adjusted. Either the diagonal of $\mathbf{M}$ or its Cholesky decomposition is chosen as pre-conditioner and after a Cholesky pre-conditioner is computed, HDSDP reuses it for the consecutive solves till a heuristic determines that the current pre-conditioner is outdated. When the algorithm approaches optimality, $\mathbf{M}$ might become ill-conditioned and HDSDP switches to LDLT decomposition in case Cholesky fails.
Using the Newton's direction, HDSDP computes the maximum stepsize

$$\alpha = \max \{\alpha \in [0,1] : \mathbf{S} + \alpha\Delta\mathbf{S} \succeq \mathbf{0}, \tau + \alpha\Delta\tau \geq 0\}$$

via a Lanczos procedure [Toh, 2002]. Then to determine a proper stepsize, one line-search determines α_c such that the barrier satisfies $-\log\det(\mathbf{S} + \alpha_c\Delta\mathbf{S}) - \log\det(\tau + \alpha_c\Delta\tau) \leq -\log\det(\mathbf{S} + \alpha_c\Delta\mathbf{S}) - \log\det(\tau + \alpha_c\Delta\tau)$ and a second line-search chooses γ such that $\mathbf{S} + \alpha_c\mathcal{A}^*\Delta\mathbf{y}_2 + \alpha_c\gamma(\mathbf{R} - \mathcal{A}^*\Delta\mathbf{y}_3) \succeq \mathbf{0}$. Next a full direction $(\Delta\mathbf{y}^\gamma, \Delta\mathbf{S}^\gamma, \Delta\tau^\gamma)$ is assembled from the Newton system with damping factor γ and a third Lanczos procedure computes

$$\alpha' = \max\left\{\alpha \in [0,1] : \mathbf{S} + \alpha\Delta\mathbf{S}^r \succeq \mathbf{0}, \tau + \alpha\Delta\tau^r \geq 0\right\}$$

. Last HDSDP updates $\mathbf{y} \leftarrow \mathbf{y} + 0.95\alpha'\Delta\mathbf{y}^r$ and $\tau \leftarrow \tau + 0.95\alpha\Delta\tau^r$. To make further use of the Schur matrix and to maintain centrality, the above procedure is repeated several times in an iteration.

In HDSDP, the update of the barrier parameter μ is another critical factor. At the end of each iteration, HDSDP updates the barrier parameter μ by $\left(\bar{z} - \mathbf{b}^\top\mathbf{y} + \theta\|\mathbf{R}\|_F\right)/\rho n$, where ρ and θ are pre-defined parameters. Heuristics also adjust μ using previously computed α_c, α' and γ.

To get the best of the two worlds, HDSDP implements the same dual-scaling algorithm as in DSDP5.8 assuming a dual feasible solution. When the dual infeasibility $\|\mathcal{A}^*\mathbf{y} + \mathbf{S} - \mathbf{C}\|_F \leq \varepsilon\tau$ and μ are sufficiently small, HDSDP fixes $\tau = 1$, re-starts with $(\mathbf{y}/\tau, \mathbf{S}/\tau)$ and applies dual-scaling to guide convergence. For the detailed implementation of dual-scaling in DSDP5.8 refer to [Benson and Ye, 2008]. To sum up, HDSDP implements strategies and computational tricks tailored for the embedding and can switch to DSDP5.8 once a dual feasible solution is available.

4 INITIALIZATION AND FEASIBILITY CERTIFICATE

Internally HDSDP deals with the following problem

$$\begin{aligned} \min_{\mathbf{X}} \quad & \langle \mathbf{C}, \mathbf{X} \rangle + \mathbf{u}^\top \mathbf{x}_u + \mathbf{l}^\top \mathbf{x}_l \\ \text{subject to} \quad & \mathcal{A}\mathbf{X} + \mathbf{x}^u - \mathbf{x}^l = \mathbf{b} \\ & \mathbf{X} \succeq \mathbf{0}, \mathbf{x}_u \geq \mathbf{0}, \mathbf{x}_l \geq \mathbf{0} \end{aligned}$$

and together with its dual, the HSD embedding is

$$\begin{aligned} \mathcal{A}\mathbf{X} + \mathbf{x}^u - \mathbf{x}^l - \mathbf{b}\tau &= \mathbf{0} \\ -\mathcal{A}^*\mathbf{y} + \mathbf{C}\tau - \mathbf{S} &= \mathbf{0} \\ \mathbf{b}^\top \mathbf{y} - \langle \mathbf{C}, \mathbf{X} \rangle - \mathbf{u}^\top \mathbf{x}_u - \mathbf{l}^\top \mathbf{x}_l - \kappa &= 0 \\ \mathbf{X}, \mathbf{S} \succeq 0, \kappa, \tau &\geq 0, \end{aligned}$$

where the primal problem is relaxed by two slack variables with penalty $\mathbf{l}, \mathbf{u}$ to prevent $\mathbf{y}$ going too large. Using the embedding, HDSDP needs no big-M initialization in the dual variable and starts from arbitrary $\mathbf{S} > \mathbf{0}, \tau > 0$. By default default $\mathbf{S}$ is initialized by a multiple of the identity matrix. One feature of the HSD embedding is its capability to detect infeasibility. Given infeasibility tolerance ε_f, HDSDP classifies the problem as primal unbounded, dual infeasible if $\|\mathbf{R}\|_F > \varepsilon_f \tau, \tau/\kappa < \varepsilon_f$ and $\mu/\mu_0 \leq \varepsilon_f^2$. If $\mathbf{R}$ is eliminated by HDSDP, then the embedding certificates dual feasibility. If $\mathbf{b}^\top \mathbf{y} > \varepsilon_f^{-1}$, then HDSDP begins checking if the Newton's step $(\Delta\mathbf{y}, \Delta\mathbf{S})$ is a dual improving ray. Once a ray is detected, the problem is classified as primal infeasible dual unbounded.

5 HDSDP SOFTWARE

HDSDP is written in ANSI C and provides a self-contained user interface. While DSDP serves as a sub-routine library, HDSDP is designed to be a stand-alone SDP solver and re-written to accommodate the new computational tricks and third-party packages. After the user inputs the data and invokes the optimization routine, HDSDP goes through several modules including input check, pre-solving, two-phase optimization, solution recovery and post-solving. The modules are implemented independently and are sequentially organized in a pipeline by HDSDP.

5.1 Pre-solver

One important feature of HDSDP is a special pre-solving module designed for SDPs. It inherits strategies from DSDP5.8 and adds new tricks to work jointly with the optimization module.

When the pre-solver is invoked, it first goes through the problem data $\{\mathbf{A}_i\}$ two rounds to detect the possible low-rank structure. The first round uses Gaussian elimination for rank-one structure and the second round applies eigenvalue decomposition from Lapack. Two exceptions are when the data matrix is too dense or sparse. If a matrix looks too dense to be decomposed efficiently, it is skipped and marked as full-rank; if it has very few entries, an elegant solution from DSDP5.8 is applied: **1)** a permutation gathers the non-zeros to a much smaller dense block. **2)** dense eigen routines from Lapack applies. **3)** the inverse permutation recovers the decomposition.

After detecting the hidden low-rank structures, the pre-solver moves on to the analysis of the Schur matrix $\mathbf{M}$: **1)**. the sparsity and rank information of the matrices are collected. **2)**. a permutation of $\{\mathbf{A}_i\}$ is generated in descending order of sparsity. **3)**. for each row of $\mathbf{M}$, the flops using each of **M1** to **M5** technique is computed and the cheapest technique is recorded. The recorded techniques reduces the flops to set up $\mathbf{M}$ and accelerates the convergence.

Last the pre-solver scales the coefficients and goes on to detect the following structures. **1)**. Implied trace: constraints implying $\mathrm{tr}(\mathbf{X}) = \theta$. **2)**. Implied dual bound: constraints implying $\mathbf{l} \leq \mathbf{y} \leq \mathbf{u}$. **3)**. Empty primal interior: constraints implying $\mathrm{tr}\left(\mathbf{X}\mathbf{a}\mathbf{a}^\top\right) \approx 0$. **4)**. Empty dual interior: constraints implying $\mathbf{A}^\top\mathbf{y} = \mathbf{C}$. **5)**. Feasibility problem. $\mathbf{C} = \mathbf{0}$. For each of the cases above the solver adjusts its internal strategies to enhance the numerical stability and convergence.

5.2 Two-phase Optimization

HDSDP implements two phase-algorithm which integrates HSD embedding (Phase A) and dual-scaling (Phase B). Phase A targets feasibility certificate and numerical stability, while Phase B aims to efficiently drive a dual-feasible solution to optimality using potential function.

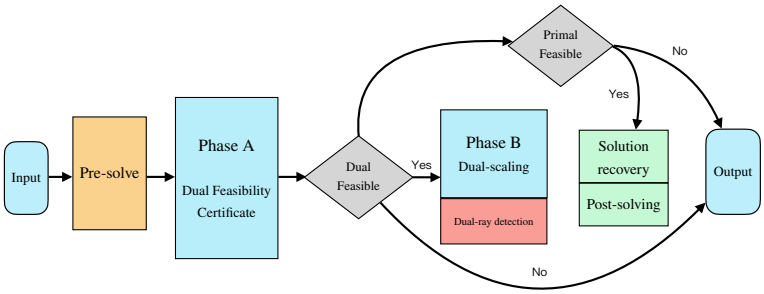

Fig. 1. Pipeline of HDSDP

In a word, the two phases share the same backend computation routines but are associated with different goals and strategies. HDSDP decides which strategy to use based on the solution behavior.

5.3 KKT Solver and conic interface

5.4 Linear system interface

6 CONCLUSIONS

We propose an extension of the dual-scaling algorithm based on the embedding technique. The resultant solver, HDSDP, is presented to demonstrate how dual method can be effectively integrated with the embedding. HDSDP is developed in parallel to DSDP5.8 and is entailed with several newly added features, especially an advanced conic KKT solver. The solver exhibits promising performance on several benchmark datasets and is under active development. Users are welcome to try the solver and provide valuable suggestions.

7 ACKNOWLEDGEMENT

We thank Dr. Qi Huangfu and Dr. Joachim Dahl from COPT development team [Ge et al., 2022] in the solver design and implementation. We also appreciate Hans Mittelmann's efforts in benchmarking the solver. Finally, we sincerely respect the developers of DSDP for their precious suggestions [Benson and Ye, 2008] and their invaluable efforts getting DSDP through all along the way. It is the efficient and elegant implementation from DSDP5.8 that guides HDSDP to where it is.

REFERENCES

Mosek ApS. 2019. Mosek optimization toolbox for matlab. *UserâĂŹs Guide and Reference Manual, Version* 4 (2019).

Steven J Benson and Yinyu Ye. 2008. Algorithm 875: DSDP5–software for semidefinite programming. *ACM Transactions on Mathematical Software (TOMS)* 34, 3 (2008), 1–20.

Steven J Benson, Yinyu Ye, and Xiong Zhang. 2000. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM Journal on Optimization* 10, 2 (2000), 443–461.

Steven J Benson, Yinyu Yeb, and Xiong Zhang. 1999. Mixed linear and semidefinite programming for combinatorial and quadratic optimization. *Optimization Methods and Software* 11, 1-4 (1999), 515–544.

Pratik Biswas and Yinyu Ye. 2004. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*. 46–54.

Brian Borchers. 2006. CSDP UserâĂŹs Guide.

Katsuki Fujisawa, Masakazu Kojima, and Kazuhide Nakata. 1997. Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. *Mathematical Programming* 79, 1 (1997), 235–253.

Dongdong Ge, Qi Huangfu, Zizhuo Wang, Jian Wu, and Yinyu Ye. 2022. Cardinal Optimizer (COPT) User Guide. *arXiv preprint arXiv:2208.14314* (2022).

Michel X Goemans and David P Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)* 42, 6 (1995), 1115–1145.

Masahito Hayashi. 2016. *Quantum information theory*. Springer.

Michal Kocvara, Michael Stingl, and PENOPT GbR. 2006. Pensdp users guide (version 2.2). *PENOPT GbR* 1435 (2006), 1436.

Tomasz Kwasniewicz and François Glineur. 2021. Implementation of a semidefinite optimization solver in the Julia programming language. (2021).

Monique Laurent. 2009. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*. Springer, 157–270.

Monique Laurent and Franz Rendl. 2005. Semidefinite programming and integer programming. *Handbooks in Operations Research and Management Science* 12 (2005), 393–514.

Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. 2020. Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics. *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), 331–360.

Imre Polik, Tamas Terlaky, and Yuriy Zinchenko. 2007. SeDuMi: a package for conic optimization. In *IMA workshop on Optimization and Control, Univ. Minnesota, Minneapolis*. Citeseer.

Florian A Potra and Rongqin Sheng. 1998a. On homogeneous interrior-point algorithms for semidefinite programming. *Optimization Methods and Software* 9, 1-3 (1998), 161–184.

Florian A Potra and Rongqin Sheng. 1998b. A superlinearly convergent primal-dual infeasible-interior-point algorithm for semidefinite programming. *SIAM Journal on Optimization* 8, 4 (1998), 1007–1028.

Anthony Man-Cho So and Yinyu Ye. 2007. Theory of semidefinite programming for sensor network localization. *Mathematical Programming* 109, 2 (2007), 367–384.

Kim-Chuan Toh. 2002. A note on the calculation of step-lengths in interior-point methods for semidefinite programming. *Computational Optimization and Applications* 21, 3 (2002), 301–310.

Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. 2012. On the implementation and usage of SDPT3–a Matlab software package for semidefinite-quadratic-linear programming, version 4.0. In *Handbook on semidefinite, conic and polynomial optimization*. Springer, 715–754.

Lieven Vandenberghe and Stephen Boyd. 1996. Semidefinite programming. *SIAM review* 38, 1 (1996), 49–95.

Henry Wolkowicz. 2005. Semidefinite and cone programming bibliography/comments. *http://orion. uwaterloo. ca/˜ hwolkowi/henry/book/fronthandbk. d/sdpbibliog. pdf* (2005).

Makoto Yamashita, Katsuki Fujisawa, Mituhiro Fukuda, Kazuhiro Kobayashi, Kazuhide Nakata, and Maho Nakata. 2012. Latest developments in the SDPA family for solving large-scale SDPs. In *Handbook on semidefinite, conic and polynomial optimization*. Springer, 687–713.

Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. 2015. SDPNAL++: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation* 7, 3 (2015), 331–366.