HDSDP: 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: Software for Semidefinite Programming. 1, 1 (July 2018), 17 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, especially a dual method based on the simplified homogeneous self-dual embedding, have been implemented. The embedding 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 algorithms benefit 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. The pre-built binary of HDSDP is currently freely available at https://github.com/COPT-Public/HDSDP.

1 INTRODUCTION

Semidefinite programming (SDP) is a mathematical programming problem 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 has been employed in various 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 of SDP applications.

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 many applications, a fundamental issue is how to numerically solve them. Theoretically, SDP is a convex conic problem which admits efficient polynomial-time algorithms and for general SDPs, the interior point method (IPM) is known as a most robust and efficient approach. Since the 1990s, high-performance SDPs softwares based on the IPM have been developed, including DSDP [Benson and Ye, 2008], COPT [Operations, 2022], Mosek [ApS, 2019], Sedumi [Polik et al., 2007], SDPT3 [Toh et al., 2012], CSDP [Borchers, 2006] and SDPA [Yamashita et al., 2012]. While most SDP codes implement the IPM, there also exist a number of successful attempts adopting other algorithms including [Kocvara et al., 2006, Kwasniewicz and Glineur, 2021, Yang et al., 2015] and a list of such SDP software is available from [Majumdar et al., 2020].

SDP solvers based on different IPM variants enjoy nice convergence behavior both theoretically and practically. Most SDP solvers implement a path-following primal-dual approach either using infeasible start [Potra and Sheng, 1998b] or the homogeneous self-dual embedding [Potra and Sheng, 1998a] with DSDP being an exception. DSDP implements a dual IPM method based on the potential reduction framework proposed in [Benson et al., 1999]. Since the initial release [Benson et al., 2000], DSDP has gone a long way through several major updates and evolved into an efficient and robust solver for general SDPs [Benson and Ye, 2008]. To further enhance the efficiency and robustness of DSDP, we make another extension by incorporating the well-known homogeneous self-dual (HSD) embedding into the dual algorithm. This new implementation, named HDSDP, is presented in this manuscript.

The rest of the manuscript is organized as follows. **Section 2** describes the SDP formulation of interest and basic notations. **Section 3** reviews the dual-scaling algorithm for SDP and describes how to combine it with the simplified HSD embedding. In **Section 4** and **5**, we introduce the practical aspects for HDSDP. Last we present the computational results of various SDP problems.

2 FORMULATION AND NOTATIONS

HDSDP is interested in the standard form primal SDP and its dual

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

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 $\langle \mathbf{A}, \mathbf{X}\rangle := \sum_{k,l} c_{ij} x_{ij}$ 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 $\mathcal{A} : \mathbb{S}^n \to \mathbb{R}^m$ and its adjoint $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{S}^n$ are 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 matrix Frobenius norm. With the above notations, we rewrite the primal and dual problems by

$$(P)\quad \begin{array}{ll} \min_{\mathbf{X}} & \langle \mathbf{C}, \mathbf{X}\rangle \\ \text{subject to} & \mathcal{A}\mathbf{X} = \mathbf{b}, \quad i = 1, \dots, m \\ & \mathbf{X} \succeq \mathbf{0} \end{array} \qquad (D)\quad \begin{array}{ll} \max_{\mathbf{y},\mathbf{S}} & \mathbf{b}^\top \mathbf{y} \\ \text{subject to} & \mathcal{A}^*\mathbf{y} + \mathbf{S} = \mathbf{C} \\ & \mathbf{S} \succeq \mathbf{0}. \end{array}$$

and the feasible regions for (P) and (D) are respectively $\mathcal{F}(P) := \{\mathbf{X} : \mathcal{A}\mathbf{X} = \mathbf{b}, \mathbf{X} \succeq \mathbf{0}\}$ and $\mathcal{F}(D) := \{(\mathbf{y}, \mathbf{S}) : \mathcal{A}^*\mathbf{y} + \mathbf{S} = \mathbf{C}, \mathbf{S} \succeq \mathbf{0}\}$. Any $\mathbf{X} \in \mathcal{F}(P)$ is called primal feasible and $(\mathbf{y}, \mathbf{S}) \in \mathcal{F}(D)$ is dual feasible. The interior of the feasible regions are denoted by $\mathcal{F}^0(P), \mathcal{F}^0(D)$ and any point in $\mathcal{F}^0$ is called an interior point solution.

Remark 1. *Although in this manuscript we focus on the SDP of single-block for ease of exposition, a natural generalization to multi-block SDPs applies rather straightforward.*

HDSDP implements a dual method that solves both (*P*) and (*D*) through the simplified HSD and in the next section we get down to the underlying theoretical details.

3 HOMOGENEOUS DUAL SCALING ALGORITHM

In this section we review the dual-scaling algorithm in DSDP and its interpretations through Newton's method. Then we show how the simplified HSD embedding applies to the dual method leveraging this interpretation.

3.1 Dual-scaling Algorithm

The dual-scaling algorithm for SDP is initially proposed in [Benson et al., 1999] and works under three conditions. **1)** the data $\{\mathbf{A}_i\}$ are linearly independent. **2)** (*P*) and (*D*) both admit 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 a primal-dual optimal pair $(\mathbf{X}^*, \mathbf{y}^*, \mathbf{S}^*)$ satisfying complementarity condition $\langle \mathbf{X}^*, \mathbf{S}^* \rangle = 0$ and $\mathbf{XS} = \mathbf{0}$. 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 guaranteed to exist, which serves as the foundation of the path-following IPMs. Given barrier parameter μ, by the last condition, dual-scaling starts from a dual-feasible solution $(\mathbf{y}, \mathbf{S})$ and takes Newton's step towards $\mathcal{A}\mathbf{X} = \mathbf{b}, \mathcal{A}^*\mathbf{y} + \mathbf{S} = \mathbf{C}$ and $\mathbf{XS} = \mu\mathbf{I}$ by solving

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

where the last relation linearizes $\mathbf{X} = \mu\mathbf{S}^{-1}$ instead of $\mathbf{XS} = \mu\mathbf{I}$ and $\mathbf{S}^{-1}$ is called a scaling matrix. By carefully driving μ to 0, dual-scaling eliminates infeasibility, approaches optimality, and finally solves the problem.
One feature of dual-scaling is that $\mathbf{X}$ and $\Delta\mathbf{X}$ vanish in the Schur complement system 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 $\mathbf{M}$. When $\{\mathbf{A}_i\}, \mathbf{C}$ are sparse, the dual variable $\mathbf{S} = \mathbf{C} - \mathcal{A}^*\mathbf{y}$ inherits the sparsity pattern from the data and it is therefore cheaper to iterate in the dual space to exploit sparsity. Another desirable feature of dual-scaling is the availablity of primal solution by solving a projection subproblem at the end of the algorithm [Benson and Ye, 2008]. The above properties characterize the behavior of dual-scaling.

However, the nice theoretical properties of the dual method is not free. First, an initial dual feasible solution is needed but 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 property guided to avoid deviating too far away 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 damping

factor towards the Newton's direction. To overcome the aforementioned difficulties, DSDP introduces slack 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 attempts works well in practice and makes DSDP an efficient general SDP solver. 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 methods requires prior estimation of the optimal solution to retain optimality. Also, a large penalty might lead to numerical difficulties and misclassification of infeasible problems when the problem is ill-conditioned. It is natural to ask if there exists an alternative to big-M method to address these issues and in this work, we propose to leverage the well-known HSD embedding.

3.2 Homogeneous Dual-scaling Algorithm

In this section, we introduce the theoretical idea behind HDSDP. HSD embedding is a skew symmetric system whose non-trivial interior point solution certificates primal-dual feasibility. {hdsdp adopts the simplified HSD embedding [Xu et al., 1996] and its extension to SDP [Potra and Sheng, 1998a].

HSD embedding for SDP

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

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

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

Here $(\mathbf{y}, \mathbf{S}, \tau)$ are jointly considered as dual variables. Given a dual point $(\mathbf{y}, \mathbf{S}, \tau)$ such that $-\mathcal{A}^*\mathbf{y} + \mathbf{C}\tau - \mathbf{S} = \mathbf{R}$, HDSDP chooses 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.

$$\begin{aligned} &\begin{pmatrix} \mu\mathbf{M} & -\mathbf{b} - \mu\mathcal{A}\mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1} \\ -\mathbf{b} + \mu\mathcal{A}\mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1} & -\mu\left(\left\langle \mathbf{C}, \mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-1}\right\rangle + \tau^{-2}\right) \end{pmatrix} \begin{pmatrix} \Delta\mathbf{y} \\ \Delta\tau \end{pmatrix} \\ = \; &\begin{pmatrix} \mathbf{b}\tau \\ \mathbf{b}^{\top}\mathbf{y} - \mu\tau^{-1} \end{pmatrix} - \mu \begin{pmatrix} \mathcal{A}\mathbf{S}^{-1} \\ \left\langle \mathbf{C}, \mathbf{S}^{-1}\right\rangle \end{pmatrix} + \mu \begin{pmatrix} \mathcal{A}\mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1} \\ \left\langle \mathbf{C}, \mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1}\right\rangle \end{pmatrix} \end{aligned} \tag{6}$$

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 $\Delta\mathbf{y} = \frac{\tau+\Delta\tau}{\mu}\Delta\mathbf{y}_1 - \Delta\mathbf{y}_2 + \Delta\mathbf{y}_3 + \Delta\mathbf{y}_4\Delta\tau$ into the second equation to solve for $\Delta\tau$ and finally assembles $\Delta\mathbf{y}$ using $\tau, \Delta\tau$ and μ. 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 $\mathbf{X}(\mu) \succeq \mathbf{0}$ iff. $-\mathcal{A}^{*}(\mathbf{y} - \Delta\mathbf{y}) + \mathbf{C}(\tau - \Delta\tau) - 2\mathbf{R} \succeq \mathbf{0}$. When $-\mathcal{A}^{*}(\mathbf{y} - \Delta\mathbf{y}) + \mathbf{C}(\tau - \Delta\tau) - 2\mathbf{R}$ is positive definite, an objective bound follows by

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

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

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

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

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

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 and $\mathbf{M}$ is then set up using one of the following five techniques 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

$$\mathbf{A}_i = \sum_{r=1}^{\text{rank}(\mathbf{A}_i)} \lambda_{ir} \mathbf{a}_{i,r} \mathbf{a}_{i,r}^\top$$

has to be computed at the beginning of the algorithm.

Technique M1

(1) **Setup** $\mathbf{B}_{\sigma(i)} = \sum_{r=1}^{\text{rank}(\mathbf{A}_{\sigma(i)})} \lambda_r \left(\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}\right)\left(\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}\right)^\top$.

(2) **Compute** $M_{\sigma(i)\sigma(j)} = \left\langle \mathbf{B}_{\sigma(i)}, \mathbf{A}_{\sigma(j)} \right\rangle, \forall j \geq i$.

Technique M2

(1) **Setup** $\mathbf{S}^{-1}\mathbf{a}_{\sigma(i),r}, r = 1, \ldots, r_{\sigma(i)}$.

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

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

Technique M3

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

(2) **Compute** $M_{\sigma(i)\sigma(j)} = \left\langle \mathbf{B}_{\sigma(i)}, \mathbf{A}_{\sigma(j)} \right\rangle, \forall j \geq i$.

Technique M4

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

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

Technique M5

(1) **Compute** $M_{\sigma(i)\sigma(j)} = \left\langle \mathbf{S}^{-1}\mathbf{A}_{\sigma(i)}\mathbf{S}^{-1}, \mathbf{A}_{\sigma(j)} \right\rangle, \forall j \geq i$ 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 $\alpha' = \max\{\alpha \in [0,1] : \mathbf{S} + \alpha\Delta\mathbf{S}^r \succeq \mathbf{0}, \tau + \alpha\Delta\tau^r \geq 0\}$.

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 $\left(\bar{z} - \mathbf{b}^\top\mathbf{y} + \theta\left\|\mathbf{R}\right\|_F\right)/\rho n$, 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 $\left\|\mathcal{A}^*\mathbf{y} + \mathbf{S} - \mathbf{C}\right\|_F \leq \varepsilon\tau$ 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

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

and together with its dual, the HSD embedding is

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

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 $\mathbf{S} \succ \mathbf{0}, \tau > 0$. 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 $\left\|\mathbf{R}\right\|_F > \varepsilon_f\tau, \tau/\kappa < \varepsilon_f$ 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.

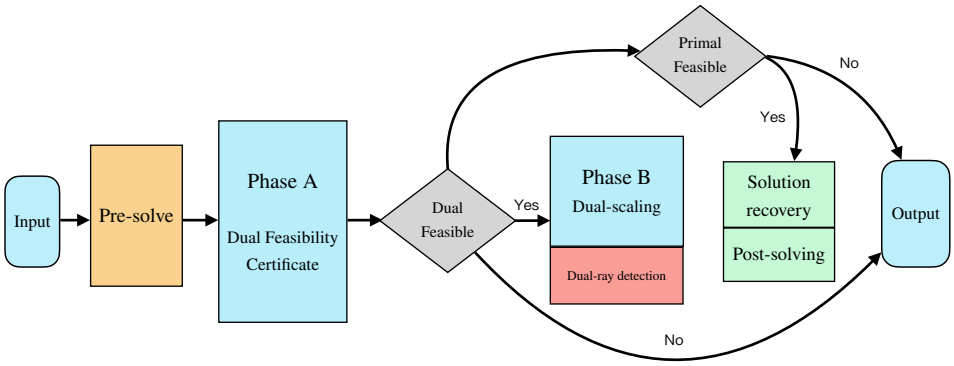

Fig. 1. Pipeline of HDSDP

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 Iteration Monitor

To help the users capture the progress of the solver, HDSDP prints related information to the screen at different stages of optimization.

```
-------------------------------------------------------------------------------
| Start presolving
| - XXX completes in 0.001 seconds
| - Matrix statistics ready in 0.000 seconds
|    Schur Re-ordering: M1: 0  M2: 3000  M3: 0  M4: 1  M5: 0
| - Special structures found
|    tr(X) = 3.00e+03 : Bound of X fixed
| Presolve Ends. Elapsed Time: 0.371 seconds
-------------------------------------------------------------------------------
```

While pre-solving, HDSDP prints time statistics when certain operation is done. Specially, the row

```
Schur Re-ordering: M1: 0  M2: 3000  M3: 0  M4: 1  M5: 0
```

indicates how many times each Schur complement technique is applied and if special SDP structures are detected, they are also printed to the screen.

```
tr(X) = 3.00e+03 : Bound of X fixed.
```

When the pre-solving ends, HDSDP prints matrix statistics including type and sum of norm. Also, the solver internally adjusts its parameters based on the collected information and display them to the user.

```
-------------------------------------------------------------------------------
| Matrix statistics [Including C]:
-------------------------------------------------------------------------------
|     Zero |   Sparse |    Dense |   Rank-1 |      |A|    |      |b|    |      |C|     |
-----------------------------------------------|-------------------------------------------
|        0 |        1 |        0 |     3000 |  3.0000e+03 |  3.0000e+03 |  1.2000e+04
-------------------------------------------------------------------------------
| Parameter Summary:
-------------------------------------------------------------------------------
| Rhon [1.0, 10.0]: 5
| Golden linesearch \{0, 1\}: 0
| Primal relaxation penalty (0.0, inf): 1e+07
| Time limit (0.0, inf): 15000
| Corrector A: 4  Corrector B: 0
-------------------------------------------------------------------------------
| DSDP is initialized with Ry = -1.225e+05 * I
| DSDP Phase A starts. Eliminating dual infeasibility
-------------------------------------------------------------------------------
```

After pre-solving, HDSDP enters optimization, invokes HSD embedding (Phase A), and prints logs.

```
-------------------------------------------------------------------------------------------
| Iter |     pObj |     dObj |     dInf |     k/t |      mu |   step |   Pnorm |   E |
-------------------------------------------------------------------------------------------
|    1 |  1.0e+05 |  0.0e+00 | 6.71e+06 | 1.0e+00 | 4.0e+04 |   0.00 | 1.0e+20 |     |
|    2 |  2.4e+08 | -7.3e+08 | 0.00e+00 | 1.0e+00 | 3.8e+04 |   1.00 | 1.1e+02 |   * |
-------------------------------------------------------------------------------------------
```

Iter	the iteration number
pObj	the primal objective bound
dObj	the dual objective
k/t	κ/τ from the embedding
mu	the current barrier parameter
step	stepsize α taken
Pnorm	the proximity to the central path
E	event monitor

Table 1. Iteration monitor of Phase A

When Phase A finds a dual-feasible solution, HDSDP collection the solution statistics to adjust the parameters for dual-scaling in Phase B, prints related information and performs re-start.

```
-------------------------------------------------------------------------------
| DSDP Phase A ends with status: DSDP_PRIMAL_DUAL_FEASIBLE
| Elapsed Time: 0.535 seconds
-------------------------------------------------------------------------------
| DSDP Phase A certificates primal-dual feasibility
| Primal relaxation penalty is set to  2.449e+06
| Perturbing dual iterations by  0.000e+00
| DSDP Phase B starts. Restarting dual-scaling
| Heuristic start: mu:  2.177e+04 pObj:  2.450e+08  dObj: -7.346e+08
-------------------------------------------------------------------------------
```

The log from Phase B is similar to Phase A but dInf is now replaced by pInf to characterize primal infeasibility. Also, k/t is dropped from log since the embedding is not applied.

```
-------------------------------------------------------------------------------
| Iter |        pObj |        dObj |       pInf |       mu |   step |    Pnorm |   E |
-------------------------------------------------------------------------------
|    1 |  2.4500e+08 | -7.3462e+08 |  3.001e+03 | 2.18e+04 |   1.00 |  1.1e+02 |     |
|    2 |  2.4500e+08 | -3.6742e+07 |  3.001e+03 | 2.18e+04 |   0.09 |  5.6e+02 |     |
|    3 |  1.3061e+08 | -1.8487e+06 |  8.915e-03 | 3.72e+03 |   0.41 |  1.3e+02 |   P |
...
|   14 | -1.1997e+04 | -1.2000e+04 |  9.510e-11 | 4.66e-06 |   0.00 |  9.4e+00 |   P |
|   15 | -1.1999e+04 | -1.2000e+04 |  1.902e-12 | 9.32e-07 |   0.02 |  5.1e+01 |   F |
-------------------------------------------------------------------------------
```

When Phase B converges, the solver extracts the solution status, recovers primal feasible solution (if available) and briefly prints solution and time statistics.

```
-------------------------------------------------------------------------------
| DSDP Phase B ends with status: DSDP_INTERNAL_ERROR
| Elapsed Time: 5.960 seconds
-------------------------------------------------------------------------------
| DSDP Ends.
-------------------------------------------------------------------------------
| Primal solution is extracted.
| Final pObj: -1.19993e+04   dObj: -1.20000e+04
-------------------------------------------------------------------------------
| DSDP Time Summary:
-------------------------------------------------------------------------------
|           Event |    Time(s) |
-------------------------------------------------------------------------------
|         Presolve |      0.371 |
|  Phase A (iter) |      0.535 | (2)
|  Phase B (iter) |      5.960 | (15)
|           Get X |      0.809 |
|       Postsolve |      0.000 |
|             All |      7.675 | (17)
-------------------------------------------------------------------------------
```

Compared to other IPM solvers, HDSDP tends to display more information so that the users better understand structures and features of the SDP instance. We believe that knowledge of the problem structure would assist the users when customzing the solver for their applications.

6 COMPUTATIONAL RESULTS

The efficiency of robustness of DSDP5.8 has been proven through years of computational experience and HDSDP aims to achieve further improvement on a special class of SDPs where dual method has advantage over the primal-dual solvers. In this section, we introduce several classes of SDPs suitable for the dual method and we compare the performance of HDSDP, DSDP5.8 and COPT 5.0 (fastest solver on Mittelmann's benchmark implementing both primal-dual and dual method) on several benchmark datasets to verify the performance improvement of HDSDP. For each problem, we only describe the mathematical model and refer the readers to [Borchers, 1999, Mittelmann, 2003] for the detailed background and formulation. The time statistics in this section are obtained using Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz with 64GB memory.

6.1 Maximum-cut

The SDP relaxation of the max-cut problem is represented by

$$\begin{aligned} \min_{\mathbf{X}} \quad & \langle \mathbf{C}, \mathbf{X} \rangle \\ \text{subject to} \quad & \operatorname{diag}(\mathbf{X}) = \mathbf{1} \\ & \mathbf{X} \succeq \mathbf{0}. \end{aligned}$$

Let $\mathbf{e}_i$ be the i-th column of the identity matrix and the constraint $\operatorname{diag}(\mathbf{X}) = \mathbf{e}$ is decomposed into $\left\langle \mathbf{X}, \mathbf{e}_i \mathbf{e}_i^\top \right\rangle = 1, i = 1, \ldots, n$. Note that $\mathbf{e}_i \mathbf{e}_i^\top$ is rank-one and has only one non-zero entry, **M2** and **M5** can greatly reduce the computation of the Schur matrix.

Instance	HDSDP	DSDP5.8	COPT v5.0	Instance	HDSDP	DSDP5.8	COPT v5.0
mcp100	0.03	0.03	0.12	maxG51	1.46	4.65	5.97
mcp124-1	0.05	0.03	0.15	maxG55	146.92	606.66	520.05
mcp124-2	0.05	0.02	0.17	maxG60	323.62	1485.00	1269.83
mcp124-3	0.05	0.04	0.16	G40_mb	12.76	17.08	25.76
mcp124-4	0.06	0.05	0.15	G40_mc	8.09	38.54	49.07
mcp250-1	0.15	0.10	0.66	G48_mb	16.70	29.71	50.49
mcp250-2	0.11	0.15	0.66	G48mc	4.42	18.10	33.19
mcp250-3	0.17	0.21	0.62	G55mc	118.11	344.80	505.18
mcp250-4	0.19	0.29	0.69	G59mc	181.20	774.70	727.89
mcp500-1	0.60	0.32	0.48	G60_mb	261.50	650.00	964.20
mcp500-2	0.69	0.74	0.72	G60mc	257.10	600.08	962.79
mcp500-3	0.82	1.12	1.11	torusg3-8	0.85	1.42	1.04
mcp500-4	1.11	1.84	2.35	torusg3-15	23.77	178.8	137.60
maxG11	1.27	0.82	1.07	toruspm3-8-50	0.76	0.93	0.76
maxG32	5.13	8.57	10.77	toruspm3-15-50	19.27	91.67	117.92

Table 2. Max-cut problems

Computational experience suggests that on large-scale max-cut instances, HDSDP is $2 \sim 3$ times faster than DSDP5.8.

6.2 Graph Partitioning

The SDP relaxation of the graph partitioning problem is given by

$$\begin{aligned} \min_{\mathbf{X}} \quad & \langle \mathbf{C}, \mathbf{X} \rangle \\ \text{subject to} \quad & \operatorname{diag}(\mathbf{X}) = \mathbf{1} \\ & \left\langle \mathbf{1}\mathbf{1}^\top, \mathbf{X} \right\rangle = \beta \\ & k\mathbf{X} - \mathbf{1}\mathbf{1}^\top \succeq \mathbf{0} \\ & \mathbf{X} \succeq \mathbf{0}, \end{aligned}$$

where $\mathbf{1}$ denotes the all-one vector and k, β are the problem parameters. Although the dual $\mathbf{S}$ no longer enjoys sparsity, the low-rank structure is still available to accelerate convergence.

On graph partitioning, we see that HDSDP has comparable performance to DSDP but is more robust on some problems.

Instance	HDSDP	DSDP5.8	COPT v5.0	Instance	HDSDP	DSDP5.8	COPT v5.0
gpp100	0.03	0.04	0.19	gpp250-4	0.19	0.16	1.21
gpp124-1	0.04	0.09	0.28	gpp500-1	0.60	0.63	0.64
gpp124-2	0.05	0.05	0.28	gpp500-2	0.69	0.60	0.56
gpp124-3	0.05	0.05	0.34	gpp500-3	0.82	0.55	0.56
gpp124-4	0.06	0.05	0.36	gpp500-4	1.11	0.58	0.51
gpp250-1	0.15	0.16	1.58	bm1	2.28	2.36	1.74
gpp250-2	0.11	0.15	1.33	biomedP	221.2	Failed	Failed
gpp250-3	0.17	0.14	1.46	industry2	Failed	Failed	Failed

Table 3. Graph partitioning problems

6.3 Optimal Diagonal Pre-conditioning

The optimal diagonal pre-conditioning problem originates from [Qu et al., 2020], where given a matrix $\mathbf{B} \succ \mathbf{0}$, finding a diagonal matrix $\mathbf{D}$ to minimize the condition number $\kappa\left(\mathbf{D}^{-1/2}\mathbf{B}\mathbf{D}^{-1/2}\right)$ can be modeled as an SDP. The formulation for optimal diagonal pre-conditioning is given by

$$\begin{aligned} \max_{\tau,\mathbf{D}} \quad & \tau \\ \text{subject to} \quad & \mathbf{D} \preceq \mathbf{B} \\ & \tau\mathbf{B} - \mathbf{D} \preceq \mathbf{0}. \end{aligned}$$

Expressing $\mathbf{D} = \sum_i \mathbf{e}_i\mathbf{e}_i^\top d_i$, the problem is exactly in the SDP dual form. If $\mathbf{B}$ is also sparse, the problem can be efficiently solved using the dual method. When the matrix $\mathbf{B}$ is large and sparse, we see that HDSDP dominates the performance

Instance	HDSDP	DSDP5.8	COPT v5.0
diag-bench-500-0.1	6.70	7.50	6.80
diag-bench-1000-0.01	44.50	55.20	53.90
diag-bench-2000-0.05	34.30	340.70	307.10
diag-bench-west0989	6.72	113.23	108.20
diag-bench-DK01R	13.18	Failed	Failed
diag-bench-micromass_10NN	9.35	60.127	51.71

Table 4. Optimal diagonal pre-conditioning problems

of DSDP due to the Schur complement tricks.

Remark 2. *For optimal pre-conditioning, we start* HDSDP *from a non-default trivial dual feasible solution* $\tau = -10^6, \mathbf{D} = \mathbf{0}$.

6.4 Other Problems

So far HDSDP is tested and tuned over a large set of benchmarks including SDPLIB [Borchers, 1999] and Hans Mittelmann's sparse SDP benchmark [Mittelmann, 2003]. By the time this manuscript is written, HDSDP is the fourth fastest among all the benchmarked solvers.

```
Scaled shifted geometric means of runtimes ("1" is fastest solver)
              1     2.30     4.61     2.28     12.3     3.72
-------------------------------------------------------------------
count of "a"      8        5        17       13       2        11
solved of 75      74       70       61       69       64       70
===================================================================
problem         COPT     CSDP     SDPA     SDPT3    SeDuMi    HDSDP
===================================================================
```

We report the solution accuracy and local CPU time of HDSDP on Mittlelmann's benchmark in the appendix. Readers can refer to [Mittelmann, 2003] for a detailed explanation of the error measures and the criterion of a successful solve. Among all of 75 problems, 70 are successfully solved; 3 problems fail due to insufficient memory, 1 fails due to failure to find a feasible dual solution and 1 fails due to error in primal solution recovery. The benchmark test proves the efficiency and robustness of HDSDP as a general purpose SDP solver. Below we present some benchmark datasets with nice structure for HDSDP. They enjoy at least one of sparsity and low-rank structure.

Instance	Background	Feature	HDSDP	DSDP5.8	COPT v5.0
checker_1.5	unknown	sparse, low-rank	55.38	146.80	137.15
foot	unknown	sparse, low-rank	25.76	23.83	262.54
hand	unknown	low-rank	5.60	5.57	50.70
ice_2.0	unknown	low-rank	706.16	1106.00	1542.96
p_auss2_3.0	unknown	sparse, low-rank	739.40	1066.00	1111.72
rendl1_2000_1e-6	unknown	low-rank	15.74	22.31	231.80
trto3	topology design	sparse, low-rank	1.67	1.73	3.04
trto4	topology design	sparse, low-rank	12.28	12.40	30.09
trto5	topology design	sparse, low-rank	111.59	151.00	233.16
sensor_500b	sensor network localization	sparse, low-rank	86.73	37.87	7.01
sensor_1000b	sensor network localization	sparse, low-rank	232.05	143.32	32.27

Table 5. Feature of several benchmark problems

6.5 Summary

HDSDP implements data structures and computational techniques specially optimized for *sparse* SDPs with *low-rank* constraint structure. Vast computational experiments suggests that on these problems HDSDP outperforms both primal-dual and conventional dual solvers in terms of efficiency and robustness on these instances.

7 WHEN (NOT) TO USE DSDP/HDSDP

While HDSDP is designed for general SDPs, it targets the problems more tractable in the dual form than by the primal-dual methods. This is the principle for the techniques implemented by HDSDP. Here are some rules in mind when deciding whether to use the dual method (or HDSDP).

(1) Does the problem enjoys nice dual structure?
Many combinatorial problems have formulations friendly to the dual methods. Some typical features include (aggregated) sparsity and low-rank structure. Dual methods effectively exploit these features by iterating in the dual space and using efficient computational tricks. If the problem is dense and most constraints are full-rank, dual method has no advantage over the primal-dual solvers due to **1)** comparable iteration cost to primal-dual methods. **2)** more iterations for convergence.

(2) Do we need the primal optimal solution or just the optimal value?
For some applications dual method fails to recover a correct primal solution due to numerical difficulties. If the optimal value is sufficient, there is no problem. But if an accurate primal optimal solution is always necessary, it is better to be more careful and to test the recovery procedure in case of failure at the last step.

(3) Do we need to certificate infeasibility strictly?

One weakness of the dual method is the difficulty in infeasibility certificate. Although on the dual side this issue is addressed by HDSDP using the embedding, dual methods still suffer from failure to identify primal infeasibility.

(4) Is dual-feasibility hard to attain?

The first phase of HDSDP adopts the infeasible Newton's method and focuses on eliminating the dual infeasibility. This principle works well if the dual constraints are relatively easy to satisfy, but if this condition fails to hold (e.g., empty dual interior), experiments suggest the embedding spend a long time deciding feasibility. In this case it is suggested using DSDP5.8 or supply an initial dual solution.

To conclude, HDSDP solves SDPs but it solves *certain* SDPs *efficiently*.

8 CONCLUSIONS

In this manuscript we propose an extension of the dual-scaling algorithm based on the HSD embedding. The resultant solver, HDSDP, is presented to demonstrate how dual method can be effectively integrated with the embedding trick. HDSDP is developed in parallel to DSDP5.8 and is entailed with several newly added computational techniques. 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.

9 ACKNOWLEDGEMENT

We thank Dr. Qi Huangfu from COPT development team for his constructive ideas 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. 1999. SDPLIB 1.2, a library of semidefinite programming test problems. *Optimization Methods and Software* 11, 1-4 (1999), 683–690.

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.

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.

Hans D Mittelmann. 2003. An independent benchmarking of SDP and SOCP solvers. *Mathematical Programming* 95, 2 (2003), 407–430.

Cardinal Operations. 2022. Cardinal Optimizer. https://www.shanshu.ai/solver

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.

Zhaonan Qu, Yinyu Ye, and Zhengyuan Zhou. 2020. Diagonal Preconditioning: Theory and Algorithms. *arXiv preprint arXiv:2003.07545* (2020).

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).

Xiaojie Xu, Pi-Fang Hung, and Yinyu Ye. 1996. A simplified homogeneous and self-dual linear programming algorithm and its implementation. *Annals of Operations Research* 62, 1 (1996), 151–171.

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.

A MITTLELMANN'S BENCHMARK TEST

The following test of the benchmark dataset is run on an `intel i11700K` with 128GB memory.

Instance	Error 1	Error 2	Error 3	Error 4	Error 5	Error 6	Time
1dc.1024	3.440000e-08	0.00000e+00	0.00000e+00	0.00000e+00	5.400000e-06	5.600000e-06	2500.186
1et.1024	7.660000e-09	0.00000e+00	0.00000e+00	0.00000e+00	3.000000e-06	2.900000e-06	175.211
1tc.1024	6.790000e-09	0.00000e+00	0.00000e+00	0.00000e+00	2.600000e-06	2.600000e-06	142.422
1zc.1024	1.230000e-08	0.00000e+00	0.00000e+00	0.00000e+00	1.400000e-06	1.300000e-06	469.066
AlH	1.510000e-09	0.00000e+00	0.00000e+00	0.00000e+00	1.100000e-04	1.100000e-04	10162.681
BH2	2.850000e-11	0.00000e+00	3.800000e-09	0.00000e+00	1.300000e-07	3.900000e-07	315.241
Bex2_1_5	1.130000e-08	0.00000e+00	3.300000e-12	0.00000e+00	5.000000e-07	1.200000e-07	273.331
Bst_jcbpaf2	1.410000e-12	0.00000e+00	5.000000e-11	0.00000e+00	2.800000e-07	1.700000e-07	419.132
CH2	1.370000e-11	0.00000e+00	2.300000e-09	0.00000e+00	3.300000e-07	4.000000e-07	315.060
G40_mb	1.170000e-06	1.200000e-17	0.00000e+00	0.00000e+00	1.800000e-05	6.000000e-07	7.025
G48_mb	1.310000e-04	0.00000e+00	0.00000e+00	1.100000e-12	1.900000e-03	6.800000e-07	8.489
G48mc	7.080000e-11	0.00000e+00	0.00000e+00	0.00000e+00	2.400000e-06	2.400000e-06	2.681
G55mc	1.390000e-10	0.00000e+00	0.00000e+00	0.00000e+00	7.300000e-07	7.300000e-07	179.720
G59mc	1.220000e-10	0.00000e+00	0.00000e+00	0.00000e+00	4.900000e-07	4.900000e-07	264.597
G60_mb	1.030000e-09	0.00000e+00	0.00000e+00	2.000000e-12	2.400000e-03	2.400000e-03	213.472
G60mc	1.030000e-09	0.00000e+00	0.00000e+00	2.000000e-12	2.400000e-03	2.400000e-03	212.088
H3O	1.090000e-09	0.00000e+00	3.700000e-09	0.00000e+00	3.100000e-07	3.600000e-07	1344.764
NH2	1.820000e-09	0.00000e+00	2.300000e-09	0.00000e+00	3.800000e-07	4.300000e-07	290.156
NH3	1.360000e-09	0.00000e+00	2.700000e-09	0.00000e+00	3.200000e-07	3.500000e-07	1367.722
NH4	4.670000e-10	0.00000e+00	3.400000e-09	0.00000e+00	2.300000e-07	3.900000e-07	5202.509
biggs	3.620000e-09	0.00000e+00	1.900000e-12	9.100000e-13	3.500000e-08	4.200000e-08	14.316
broyden25	2.320000e-09	0.00000e+00	0.00000e+00	0.00000e+00	7.000000e-09	7.300000e-09	1774.592
buck4	1.390000e-11	0.00000e+00	0.00000e+00	0.00000e+00	4.400000e-07	2.400000e-07	21.185
buck5	4.920000e-09	0.00000e+00	0.00000e+00	0.00000e+00	5.400000e-04	2.800000e-04	194.738
cancer_100	5.250000e-13	0.00000e+00	0.00000e+00	3.400000e-15	1.700000e-08	3.400000e-08	396.231
checker_1.5	2.830000e-09	0.00000e+00	0.00000e+00	0.00000e+00	1.300000e-06	1.200000e-06	41.693
chs_5000	1.210000e-17	0.00000e+00	0.00000e+00	0.00000e+00	1.500000e-07	1.500000e-07	36.825
cnhil10	3.980000e-08	0.00000e+00	0.00000e+00	0.00000e+00	8.500000e-09	2.000000e-08	63.071
cphil12	8.550000e-10	0.00000e+00	0.00000e+00	0.00000e+00	9.300000e-09	9.300000e-09	259.832
diamond_patch	5.000000e-01	-0.00000e+00	0.00000e+00	0.00000e+00	-9.400000e-01	0.00000e+00	Failed
e_moment_quad	2.370000e-10	0.00000e+00	5.400000e-09	0.00000e+00	-1.900000e-06	1.100000e-08	334.162
e_moment_stable	2.180000e-09	7.700000e-19	3.500000e-07	8.500000e-10	-1.100000e-05	4.900000e-08	256.368
foot	1.130000e-04	0.00000e+00	0.00000e+00	6.700000e-11	-2.300000e-03	1.900000e-04	12.878
hamming_8_3_4	2.740000e-09	0.00000e+00	0.00000e+00	0.00000e+00	1.600000e-08	1.600000e-08	38.635
hamming_9_5_6	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	Failed

Table 6. Mittelmann's Benchmark Test Part 1

Instance	Error 1	Error 2	Error 3	Error 4	Error 5	Error 6	Time
hand	4.500000e-08	0.00000e+00	0.00000e+00	5.200000e-14	1.100000e-06	5.200000e-07	2.431
ice_2.0	5.330000e-09	0.00000e+00	0.00000e+00	0.00000e+00	1.400000e-06	1.200000e-06	372.376
inc_1200	9.920000e-06	0.00000e+00	0.00000e+00	0.00000e+00	-2.300000e-04	2.500000e-07	128.205
mater-5	4.830000e-11	0.00000e+00	0.00000e+00	0.00000e+00	8.400000e-07	8.400000e-07	23.988
mater-6	1.670000e-10	0.00000e+00	0.00000e+00	0.00000e+00	1.100000e-06	1.100000e-06	61.504
neosfbr25	3.640000e-09	0.00000e+00	0.00000e+00	0.00000e+00	2.100000e-07	2.100000e-07	1084.074
neosfbr30e8	2.650000e-08	0.00000e+00	0.00000e+00	0.00000e+00	4.000000e-07	4.200000e-07	5924.588
neu1	3.650000e-06	9.200000e-18	4.800000e-07	5.700000e-10	-1.300000e-05	5.000000e-06	109.041
neu1g	7.850000e-10	0.00000e+00	0.00000e+00	0.00000e+00	4.000000e-09	6.400000e-08	89.921
neu2	2.480000e-08	2.700000e-17	3.900000e-07	8.300000e-10	-1.100000e-07	4.600000e-08	108.879
neu2c	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	Failed
neu2g	4.280000e-10	0.00000e+00	0.00000e+00	0.00000e+00	9.900000e-09	4.100000e-08	90.972
neu3	8.110000e-09	0.00000e+00	2.600000e-07	0.00000e+00	-8.000000e-07	3.000000e-08	1731.641
neu3g	9.090000e-09	1.600000e-17	0.00000e+00	0.00000e+00	6.500000e-09	2.500000e-08	1807.446
p_auss2_3.0	6.030000e-08	0.00000e+00	0.00000e+00	0.00000e+00	-2.200000e-06	3.600000e-07	451.407
prob_2_4_0	2.380000e-15	0.00000e+00	4.400000e-10	0.00000e+00	-7.100000e-07	6.200000e-07	221.371
prob_2_4_1	9.810000e-15	0.00000e+00	0.00000e+00	0.00000e+00	4.100000e-07	4.100000e-07	98.066
rabmo	9.160000e-10	0.00000e+00	1.600000e-07	0.00000e+00	-4.300000e-05	1.500000e-08	182.872
reimer5	1.740000e-09	0.00000e+00	1.100000e-07	0.00000e+00	-3.600000e-05	4.600000e-09	1862.190
rendl	1.530000e-10	0.00000e+00	0.00000e+00	0.00000e+00	4.100000e-07	4.100000e-07	7.351
ros_2000	5.510000e-16	0.00000e+00	0.00000e+00	0.00000e+00	1.400000e-07	1.400000e-07	3.760
rose15	8.550000e-08	7.900000e-18	1.300000e-07	3.000000e-10	-7.200000e-05	1.500000e-08	104.258
sensor_1000b	6.940000e-10	0.00000e+00	0.00000e+00	0.00000e+00	6.400000e-07	1.400000e-07	203.969
shmup4	1.770000e-08	0.00000e+00	1.400000e-13	0.00000e+00	9.700000e-07	5.000000e-07	63.314
shmup5	1.070000e-05	0.00000e+00	2.400000e-10	0.00000e+00	6.300000e-05	5.100000e-07	770.997
spar060-020-1_LS	7.770000e-08	0.00000e+00	0.00000e+00	0.00000e+00	9.100000e-07	9.900000e-09	779.882
swissroll	9.480000e-06	0.00000e+00	0.00000e+00	0.00000e+00	-7.300000e-03	2.100000e-07	37.456
taha1a	1.800000e-10	0.00000e+00	2.400000e-07	0.00000e+00	-1.900000e-07	2.400000e-07	196.912
taha1b	8.660000e-09	0.00000e+00	2.100000e-10	0.00000e+00	7.200000e-08	9.100000e-08	670.662
taha1c	9.900000e-09	0.00000e+00	4.100000e-07	0.00000e+00	1.800000e-05	9.900000e-04	2199.454
theta12	4.920000e-08	0.00000e+00	0.00000e+00	0.00000e+00	2.500000e-07	2.900000e-07	1095.150
theta102	7.370000e-11	0.00000e+00	0.00000e+00	0.00000e+00	6.600000e-07	6.600000e-07	5378.786
theta123	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	Failed
tiger_texture	9.230000e-04	0.00000e+00	1.200000e-12	8.700000e-11	1.600000e-03	2.000000e-03	53.752
torusg3-15	1.860000e-11	0.00000e+00	0.00000e+00	0.00000e+00	3.300000e-07	3.300000e-07	24.455
trto4	8.580000e-08	0.00000e+00	0.00000e+00	0.00000e+00	-1.400000e-06	1.800000e-07	7.633
trto5	1.430000e-06	0.00000e+00	0.00000e+00	0.00000e+00	-4.700000e-05	2.400000e-07	82.491
vibra4	6.030000e-09	0.00000e+00	0.00000e+00	0.00000e+00	2.000000e-03	1.300000e-03	33.061
vibra5	3.690000e-06	0.00000e+00	1.600000e-06	0.00000e+00	5.600000e-05	1.100000e-04	326.107
yalsdp	5.290000e-09	0.00000e+00	0.00000e+00	0.00000e+00	2.400000e-06	2.400000e-06	125.136

Table 7. Mittelmann's Benchmark Test Part 2