Fast, Scalable, Warm-Start Semidefinite Programming with Spectral Bundling and Sketching

Rico Angell [1] **Andrew McCallum** [1]

Abstract

While semidefinite programming (SDP) has traditionally been limited to moderate-sized problems, recent algorithms augmented with matrix sketching techniques have enabled solving larger SDPs. However, these methods achieve scalability at the cost of an increase in the number of necessary iterations, resulting in slower convergence as the problem size grows. Furthermore, they require iteration-dependent parameter schedules that prohibit effective utilization of warm-start initializations important in practical applications with incrementally-arriving data or mixed-integer programming. We present Unified Spectral Bundling with Sketching (USBS), a provably correct, fast and scalable algorithm for solving massive SDPs that can leverage a warm-start initialization to further accelerate convergence. Our proposed algorithm is a spectral bundle method for solving general SDPs containing both equality and inequality constraints. Moreover, when augmented with an optional matrix sketching technique, our algorithm achieves the dramatically improved scalability of previous work while sustaining convergence speed. We empirically demonstrate the effectiveness of our method across multiple applications, with and without warm-starting. For example, USBS provides a 500x speed-up over the state-of-the-art scalable SDP solver on an instance with over 2 billion decision variables. We make our implementation in pure JAX publicly available[1].

[*]Equal contribution [1]Manning College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA. Correspondence to: Rico Angell <rangell@cs.umass.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]https://github.com/rangell/usbs

1. Introduction

Semidefinite programming (SDP) is a convex optimization paradigm capable of modeling or approximating many practical problems in combinatorial optimization (Alizadeh, 1995), neural network verification (Raghunathan et al., 2018; Dathathri et al., 2020), robotics (Rosen et al., 2019), optimal experiment design (Vandenberghe et al., 1998b), VLSI (Vandenberghe et al., 1998a), and systems and control theory (Bertsimas, 1995). Despite their widespread applicability, practitioners often dismiss the use of SDPs under the presumption that optimization is intractable at real-world scale. This assumption is grounded in the prohibitively high computational complexity of standard SDP solvers (Alizadeh et al., 1998; Vandenberghe & Boyd, 1999).

The main challenge when solving SDPs with standard constrained optimization approaches is the high cost of projection onto the feasible region. Projecting a symmetric matrix onto the semidefinite cone requires a full eigendecomposition, an operation that scales cubicly with the problem dimension. The high computational cost of this single operation severely limits the applicability of projected gradient and ADMM-based approaches (Boyd et al., 2011; O'Donoghue et al., 2021). Interior point methods, the de facto approach for solving small-to-moderate sized SDPs, require even more computational resources.

Projection-free and conditional gradient-based methods have been proposed to avoid the computational burden of projecting iterates onto the semidefinite cone (Yurtsever et al., 2019; 2015; Arora et al., 2005). These methods avoid projection onto the feasible region with various techniques including subgradient descent in the dual space and a conditional gradient method on an unconstrained augmented objective function. The bulk of the computational burden for these methods is typically an extreme eigenvector calculation. The scalablity of these methods is still limited, despite the improved per-iteration complexity, since they require storing the entire primal matrix which scales quadratically with the problem dimension.

The most well-known method for scaling semidefinte programming without storing the entire primal matrix in memory is the Burer–Monteiro (BM) factorization heuristic (Bu-

rer & Monteiro, 2003; Cifuentes & Moitra, 2022). The core idea of the BM method is to explicitly control the memory usage by restricting the primal matrix to a low-rank factorization. However, this method sacrifices the convexity of the problem for scalability. In some cases, the BM method either requires having a high rank factorization, leading to burdensome memory requirements, or risks optimization getting stuck in local minima (Waldspurger & Waters, 2020). Yurtsever et al. (2021) extend the conditional gradient augmented Lagrangian approach (Yurtsever et al., 2019) with a matrix sketching technique. This approach avoids storing the entire primal matrix while maintaining the convexity of the problem leading to provable convergence for general SDPs. In addition, this method uses iteration-dependent parameters prohibiting it from reliably leveraging a warm-start initialization.

Spectral bundle methods, first proposed by Helmberg & Rendl (2000), are an appealing framework for solving SDPs due to their low per-iteration computational complexity and fast empirical convergence. While several spectral bundle methods have been presented in the literature (Helmberg & Kiwiel, 2002; Apkarian et al., 2008; Helmberg et al., 2014; Ding & Grimmer, 2023), previous work considers SDPs with either only equality constraints or only inequality constraints. Furthermore, the lack of an efficient standalone implementation has prevented the evaluation of spectral bundle method on massive SDPs.

Contributions. We present USBS, a unified spectral bundle method designed for a broader class of SDPs and show it is a practical approach for solving large problem instances quickly. USBS flexibly allows the user to control the trade-off between per-iteration complexity and the empirical speed at which the algorithm converges. In addition, USBS can be augmented with a matrix sketching technique that can dramatically improve the scalability of the algorithm while maintaining its fast convergence. We demonstrate the empirical efficacy of our proposed spectral bundle method on three types of practical large SDPs seeing extraordinary performance improvements in comparison with the previous state-of-the-art scalable solver for general SDPs. For example, USBS is able to obtain a quality solution to an SDP with over 10^{13} decision variables while the previous state-of-the-art fails to reach an accurate solution within 72 hours. In addition, we see a 500x speedup over the previous state-of-the-art on an instance with 2 billion decision variables. Finally, we find that warm-starting can lead to more than a 100x speedup in the convergence of USBS as compared to cold-starting while the previous state-of-the-art often is not able to reliably take advantage of a warm-start initialization. We provide a standalone implementation of the algorithm in pure JAX (Bradbury et al., 2018; Frostig et al., 2018), enabling efficient execution on various hardware (CPU, GPU, TPU) and wide-spread use.

2. Preliminaries

2.1. Semidefinite Programming

We begin with the notation necessary to define the semidefinite programming problem. Let $\mathbb{S}^n$ be the set of all real symmetric $n \times n$ matrices and $\mathbb{S}^n_+ := \{X \in \mathbb{S}^n : X \succeq 0\}$ be the positive semidefinite cone. Let $\langle \cdot, \cdot \rangle$ denote the standard inner product for vectors and the Frobenius inner product for matrices. For any matrix $M \in \mathbb{S}^n$, let $\lambda_{\max}(M)$ denote the maximum eigenvalue of M and let $\operatorname{tr}(M)$ denote the trace of M. Let $\mathcal{A} : \mathbb{S}^n \to \mathbb{R}^m$ be a given linear operator and $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{S}^n$ be its adjoint, i.e. $\langle \mathcal{A}X, y \rangle = \langle X, \mathcal{A}^* y \rangle$ for any $X \in \mathbb{S}^n$ and $y \in \mathbb{R}^m$, which take the following form,

$$\mathcal{A}X = \begin{bmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{bmatrix} \quad \text{and} \quad \mathcal{A}^* y = \sum_{i=1}^m y_i A_i,$$

for given matrices $A_i \in \mathbb{S}^n$. For any given $\mathcal{I} \subset \{1, \dots, m\}$, let $\mathcal{A}_\mathcal{I}$ denote the linear operator restricted to matrices A_i for $i \in \mathcal{I}$ and $y_\mathcal{I}$ to be the corresponding subset of rows for any $y \in \mathbb{R}^m$. In general, we utilize $(\cdot)_\mathcal{I}$ to extract the corresponding indices from any vector in $\mathbb{R}^m$. Finally, let $\mathcal{I}' := \{1, \cdots, m\} \setminus \mathcal{I}$ be the compliment of $\mathcal{I}$.

Given fixed $C \in \mathbb{S}^n$, $\mathcal{A}$, $b \in \mathbb{R}^m$, and $\mathcal{I}$, the primal (P) and dual (D) semidefinite programs can be written as follows:

$$\begin{aligned} \max_{X \succeq 0} \ & \langle C, X \rangle \\ \text{s.t.} \ & \mathcal{A}_{\mathcal{I}'} X = b_{\mathcal{I}'} \\ & \mathcal{A}_\mathcal{I} X \leq b_\mathcal{I} \end{aligned} \quad \text{(P)} \qquad \begin{aligned} \min_{y \in \mathbb{R}^m} \ & \langle b, y \rangle \\ \text{s.t.} \ & C - \mathcal{A}^* y \preceq 0 \\ & y_\mathcal{I} \geq 0 \end{aligned} \quad \text{(D)}$$

Note that while all SDPs of this form can technically be written as equality constrained SDPs (i.e. *standard-form* SDPs), doing so in practice is ill-advised since the problem dimension, n, grows by one for every inequality constraint. In addition, we also represent the primal feasibility set as the convex set $\mathcal{K} := \{z \in \mathbb{R}^m : z_\mathcal{I} \leq b_\mathcal{I}, z_{\mathcal{I}'} = b_{\mathcal{I}'}\}$, and measure the primal infeasibility $\operatorname{dist}(\mathcal{A}X, \mathcal{K})$. We use $\operatorname{dist}(z, \mathcal{Z})$ to denote the Euclidean distance from a point z to a closed set $\mathcal{Z}$ and $\operatorname{proj}_\mathcal{Z}(z)$ to denote the Euclidean projection of z onto $\mathcal{Z}$. Equivalently,

$$\operatorname{dist}(z, \mathcal{Z}) = \inf_{z' \in \mathcal{Z}} \|z - z'\| = \|z - \operatorname{proj}_\mathcal{Z}(z)\|.$$

We denote the solution sets of (P) and (D) by $\mathcal{X}_\star$ and $\mathcal{Y}_\star$, respectively. We make the standard assumption that (P) and (D) satisfy *strong duality*, namely that the solution sets $\mathcal{X}_\star$ and $\mathcal{Y}_\star$ are nonempty, compact, and every pair of solutions $(X_\star, y_\star) \in \mathcal{X}_\star \times \mathcal{Y}_\star$ has zero duality gap: $p_\star := \langle C, X_\star \rangle = \langle b, y_\star \rangle =: d_\star$. Strong duality holds whenever Slater's condition holds and $\mathcal{A}$ is a surjective linear operator. Additionally, we say that the SDP satisfies *strict complementarity* if there exists $(X_\star, y_\star)$ such

that for induced dual slack matrix $Z_\star := C - \mathcal{A}^* y_\star$, $\operatorname{rank}(X_\star) + \operatorname{rank}(Z_\star) = n$. Strict complementarity is satisfied by generic SDPs (Alizadeh et al., 1997) and many well-structured SDPs (Ding & Udell, 2021). Lastly, we denote the maximum nuclear norm of the primal solution set as $\mathfrak{N}(\mathcal{X}_\star) := \sup_{X_\star \in \mathcal{X}_\star} \|X_\star\|_*$.

In many important applications, SDPs take a highly structured and sparse forms that admit low-rank solutions. The cost matrix C is often sparse, containing many fewer than n^2 non-zero entries. Additionally, the number of primal constraints m is often much less n^2, being proportional to n or the number of non-zero entries in C. We would like to exploit these features with a provably correct, fast and scalable optimization algorithm for solving weakly-constrained SDPs. Note that USBS will provably solve any SDP, but it might not be fast and scalable for all SDPs, especially those with dense C or $m \geq \mathcal{O}(n^2)$ or high rank solutions.

2.2. Proximal Bundle Method

Proximal bundle methods (Lemarechal et al., 1981; Mifflin, 1977; Feltenmark & Kiwiel, 2000; Kiwiel, 2000) are a class of optimization algorithms for solving unconstrained convex minimization problems of the form $\min_{y \in \mathbb{R}^m} f(y)$, where $f : \mathbb{R}^m \to (-\infty, +\infty]$ is a proper closed convex function that attains its minimum value, $\inf f$, on some nonempty set. At each iteration, the proximal bundle method proposes an update to the current iterate y_t by applying a proximal step to an approximation of the objective function, $\hat{f}_t$:

$$\tilde{y}_{t+1} \leftarrow \operatorname*{arg\,min}_{y \in \mathbb{R}^m} \hat{f}_t(y) + \frac{\rho}{2}\|y - y_t\|^2 \tag{1}$$

where $\rho > 0$. The next iterate y_{t+1} is set equal to $\tilde{y}_{t+1}$ only when the decrease in the objective value is at least a fixed fraction of the decrease predicted by the model $\hat{f}_t$, i.e.

$$\beta(f(y_t) - \hat{f}_t(\tilde{y}_{t+1})) \leq f(y_t) - f(\tilde{y}_{t+1}) \tag{2}$$

for some fixed $\beta \in (0, 1)$. The iterations where (2) is satisfied (and thus, $y_{t+1} \leftarrow \tilde{y}_{t+1}$) are referred to as *descent steps*. Otherwise, the algorithm takes a *null step* and sets $y_{t+1} \leftarrow y_t$. Regardless of whether (2) is satisfied or not, $\tilde{y}_{t+1}$ is used to construct the next model $\hat{f}_{t+1}$.

Model Requirements. The model $\hat{f}_t$ can take many forms, but is usually constructed using subgradients of f at past and current iterates. Let $\partial f(y) := \{g : f(y') \geq f(y) + \langle g, y' - y\rangle, \forall y' \in \mathbb{R}^m\}$ denote the subdifferential of f at y (i.e. the set of subgradients of f evaluated at a point y). Following prior work, we require the next model $\hat{f}_{t+1}$ to satisfy the following mild assumptions:

1. ***Minorant.***

$$\hat{f}_{t+1}(y) \leq f(y), \forall y \in \mathbb{R}^m \tag{3}$$

2. ***Subgradient lower bound.*** For any $g_{t+1} \in \partial f(\tilde{y}_{t+1})$,

$$\hat{f}_{t+1}(y) \geq f(\tilde{y}_{t+1}) + \langle g_{t+1}, y - \tilde{y}_{t+1}\rangle, \forall y \in \mathbb{R}^m \tag{4}$$

3. ***Model subgradient lower bound.*** The first order optimality conditions for (1) gives the subgradient $s_{t+1} := \rho(y_t - \tilde{y}_{t+1}) \in \partial \hat{f}_t(\tilde{y}_{t+1})$. After a null step t,

$$\hat{f}_{t+1}(y) \geq \hat{f}_t(\tilde{y}_{t+1}) + \langle s_{t+1}, y - \tilde{y}_{t+1}\rangle, \forall y \in \mathbb{R}^m \tag{5}$$

The first two conditions serve to guarantee that a new model integrates first-order information from the objective at $\tilde{y}_{t+1}$. The third condition demands the new model to preserve approximation accuracy exhibited by the preceding model.

Assuming these model requirements, Díaz & Grimmer (2023) proved non-asymptotic convergence rates for the proximal bundle method under various conditions. We summarize the results relevant to this work in Theorem D.1.

3. Unified Spectral Bundling

In this section, we will present USBS, our proposed algorithm for solving the SDP defined in (P) and (D), Our proposed spectral bundle method is a unified algorithm for solving SDPs with both equality and inequality constraints and can be integrated with a matrix sketching technique for dramatically improved scalability as demonstrated in Section 4. To apply the proximal bundle method, we consider minimizing the following unconstrained objective

$$f(y) := \alpha\,[\lambda_{\max}(C - \mathcal{A}^* y)]_+ + \langle b, y\rangle + \iota_{\mathcal{Y}}(y), \quad \text{(pen-D)}$$

where $[\,\cdot\,]_+ := \max\{\,\cdot\,, 0\}$, $\mathcal{Y} := \{y \in \mathbb{R}^m : y_{\mathcal{I}} \geq 0\}$, and where $\iota_{\mathcal{Y}}(\cdot)$ is the indicator function defined such that $\iota_{\mathcal{Y}}(y) = 0$ if $y \in \mathcal{Y}$ and $\iota_{\mathcal{Y}}(y) = +\infty$ otherwise. Minimizing (pen-D) is equivalent to optimizing (D) in the sense that the optimal solution set and objective are the same (Appendix A.1). In the following subsections, we will detail how the model is constructed and how to compute the candidate iterates.

3.1. Spectral Bundle Model

The form of (pen-D) is not quite conducive to defining the model and solving for the candidate iterate $\tilde{y}_{t+1}$. To address this, we will consider an equivalent formulation to (pen-D) that is more amenable to this effort. For any fixed $\alpha \geq 2\mathfrak{N}(\mathcal{X}_\star)$, let $\mathcal{X} := \{X \in \mathbb{S}^n_+ : \operatorname{tr}(X) \leq \alpha\}$ be the trace constrained subset of the primal domain. Let $\mathrm{N} := \{\nu \in \mathbb{R}^m : \nu_{\mathcal{I}} \leq 0, \nu_{\mathcal{I}'} = 0\}$ be the domain of the dual slack variable $\nu \in \mathbb{R}^m$ for the indicator function $\iota_{\mathcal{Y}}(\cdot)$. Then, we can rewrite the following equivalent formulation to (pen-D)

$$f(y) = \sup_{(X,\nu) \in \mathcal{X} \times \mathrm{N}} \langle C - \mathcal{A}^* y, X\rangle + \langle b - \nu, y\rangle. \tag{6}$$

We derive this equivalent formulation in Appendix A.2.

Defining the model. This equivalent formulation is clearly just as difficult to optimize as (pen-D), but is helpful in defining the model we will utilize in the spectral bundle method. The main idea is to consider (6) over a low-dimensional subspace of $\mathcal{X}$ such that (3), (4), and (5) are satisfied. The subspace we will consider at step t is parameterized by matrices $\bar{X}_t \in \mathbb{S}^n_+$ and $V_t \in \mathbb{R}^{n\times k}$ and is defined as follows

$$\widehat{\mathcal{X}}_t := \left\{ \eta \bar{X}_t + V_t S V_t^\top \;\middle|\; \begin{array}{l} \eta \operatorname{tr}(\bar{X}_t) + \operatorname{tr}(S) \le \alpha \\ \eta \ge 0, S \in \mathbb{S}^k_+ \end{array} \right\}. \quad (7)$$

The matrix V_t has k orthonormal column vectors, where k is a small user defined parameter. The columns of V_t are partitioned into $k_c \ge 1$ current eigenvectors and $k_p \ge 0$ orthonormal vectors representing past spectral information. Regardless of the setting of k_c and k_p, the columns of V_t will always include a maximum eigenvector v_1 of $C - \mathcal{A}^*\tilde{y}_t$, which also means $b - \alpha\mathcal{A}v_1v_1^\top \in \partial f(\tilde{y}_t)$. For scalability reasons, we expect $k = k_c + k_p$ to be relatively small. The matrix $\bar{X}_t$ is a carefully selected weighted sum of past spectral bounds such that $\operatorname{tr}(\bar{X}_t) \le \alpha$ and enables the last model condition (5) to be satisfied. Hence, the model is

$$\hat{f}_t(y) := \sup_{(X,\nu)\in\widehat{\mathcal{X}}_t\times \mathrm{N}} \langle C - \mathcal{A}^*y, X\rangle + \langle b - \nu, y\rangle. \quad (8)$$

We show that this model satisfies the conditions (3), (4), and (5) in Appendix D.1.

Updating the model. Independent of whether a descent step or null step is taken, the model needs to be updated. At every step, we solve the following minimax problem

$$\min_{y\in\mathbb{R}^m} \sup_{(X,\nu)\in\widehat{\mathcal{X}}_t\times \mathrm{N}} \langle C - \mathcal{A}^*y, X\rangle + \langle b-\nu, y\rangle + \frac{\rho}{2}\|y - y_t\|^2, \quad (9)$$

where $(\tilde{y}_{t+1}, X_{t+1}, \nu_{t+1})$ is the minimax solution. The structure of $\widehat{\mathcal{X}}_t$ allows us to rewrite X_{t+1} as

$$X_{t+1} = \eta_{t+1}\bar{X}_t + V_t S_{t+1} V_t^\top. \quad (10)$$

To compute $\bar{X}_{t+1}$ and V_{t+1} we first compute an eigendecomposition of S_{t+1} to separate current and past spectral information as follows

$$S_{t+1} = Q_{\overline{p}}\Lambda_{\overline{p}}Q_{\overline{p}}^\top + Q_{\underline{c}}\Lambda_{\underline{c}}Q_{\underline{c}}^\top, \quad (11)$$

where $\Lambda_{\overline{p}}$ is a diagonal matrix containing the largest k_p eigenvalues and $Q_{\overline{p}} \in \mathbb{R}^{k\times k_p}$ contains the corresponding eigenvectors. The diagonal matrix $\Lambda_{\underline{c}}$ contains the remaining k_c eigenvalues and the corresponding eigenvectors are contained in the columns of $Q_{\underline{c}} \in \mathbb{R}^{k\times k_c}$. Given this decomposition, we set the next model's $\bar{X}_{t+1}$ as follows

$$\bar{X}_{t+1} \leftarrow \eta_{t+1}\bar{X}_t + V_t Q_{\underline{c}}\Lambda_{\underline{c}}Q_{\underline{c}}^\top V_t^\top. \quad (12)$$

In the case that $k_p = 0$, observe that $\bar{X}_t = X_t$ for all t. To update V_{t+1}, we first compute the top k_c orthonormal eigenvectors $v_1, \ldots, v_{k_c}$ of $C - \mathcal{A}^*\tilde{y}_{t+1}$. Then, we set V_{t+1} to k orthonormal vectors that span the columns of $[V_tQ_{\overline{p}}; v_1, \ldots, v_{k_c}]$ as follows

$$V_{t+1} \leftarrow \texttt{orthonormalize}\Big([V_tQ_{\overline{p}}; v_1, \ldots, v_{k_c}]\Big), \quad (13)$$

where we can use QR decomposition to orthonormalize the vectors. If $k_p = 0$, orthonormalization is unnecessary and we just set $V_{t+1} \leftarrow [v_1, \ldots, v_{k_c}]$.

3.2. Computing the Candidate Iterate

We will now discuss how to compute the candidate iterate $\tilde{y}_{t+1}$ by applying a proximal step to the current approximation of the objective function (i.e. solve (1)). We show in Appendix A.3 that the candidate iterate can be computed as

$$\tilde{y}_{t+1} = y_t - \frac{1}{\rho}(b - \nu_{t+1} - \mathcal{A}X_{t+1}), \quad (14)$$

where $(X_{t+1}, \nu_{t+1}) \in \widehat{\mathcal{X}}_t \times \mathrm{N}$ is the solution to the following optimization problem

$$(X_{t+1}, \nu_{t+1}) \in \operatorname*{arg\,max}_{(X,\nu)\in\widehat{\mathcal{X}}_t\times\mathrm{N}} \psi_t(X,\nu), \quad (15)$$

where we define $\psi_t(X,\nu) := \langle C, X\rangle + \langle b - \nu - \mathcal{A}X, y_t\rangle - \frac{1}{2\rho}\|b - \nu - \mathcal{A}X\|^2$. To solve for $(X_{t+1}, \nu_{t+1}) \in \widehat{\mathcal{X}}_t \times \mathrm{N}$, we propose using the alternating maximization algorithm. After initializing $\tilde{\nu} = 0$, the the following update steps are repeated until convergence:

$$\tilde{X} \leftarrow \operatorname*{arg\,max}_{X\in\widehat{\mathcal{X}}_t} \psi_t(X, \tilde{\nu}) \quad (16)$$

$$\tilde{\nu} \leftarrow \operatorname*{arg\,max}_{\nu\in\mathrm{N}} \psi_t(\tilde{X}, \nu) = \operatorname{proj}_{\mathrm{N}}(b - \mathcal{A}\tilde{X} - \rho y_t) \quad (17)$$

Clearly ψ_t is smooth, and in this case the alternating maximization algorithm is known to converge at a $\mathcal{O}(1/\varepsilon)$ rate (Beck, 2015). If ψ_t happens to also be strongly concave, the rate improves to $\mathcal{O}(\log(1/\varepsilon))$ (Luo & Tseng, 1993). We solve (16) using a primal-dual interior point method derived in Appendix B.2.

Feasibility of $\tilde{y}_{t+1}$. If we substitute $\nu_{t+1} = \operatorname{proj}_{\mathrm{N}}(b - \mathcal{A}X_{t+1} - \rho y_t)$ into (14), it can be seen that the candidate $\tilde{y}_{t+1}$ is always feasible

$$(\tilde{y}_{t+1})_{\mathcal{I}} = \max\left\{(y_t)_{\mathcal{I}} - \frac{1}{\rho}\Big(b_{\mathcal{I}} - \mathcal{A}_{\mathcal{I}}X_{t+1}\Big), 0\right\} \ge 0,$$

and is also complementary to the dual slack variable ν_{t+1}, i.e. $\langle \tilde{y}_{t+1}, \nu_{t+1}\rangle = 0$. This view allows us to see that we can equivalently express the candidate iterate as

$$\tilde{y}_{t+1} = \operatorname{proj}_{\mathcal{Y}}\left(y_t - \frac{1}{\rho}(b - \mathcal{A}X_{t+1})\right).$$

Algorithm 1 Unified Spectral Bundling

Input: Problem specification $(C, \mathcal{A}, b, \mathcal{I})$, parameters $(\alpha, \rho, \beta, k_c, k_p)$, and initialization of $X_0 = \bar{X}_0$, y_0, and orthonormal V_0 to fully parameterize $\hat{f}_0$ and $\widehat{\mathcal{X}}_0$.
Output: X_T, y_T
for $t = 0, 1, \ldots, T$ **do**
  $(X_{t+1}, \nu_{t+1}) \leftarrow \underset{(X,\nu)\in\widehat{\mathcal{X}}_t\times\mathrm{N}}{\arg\max}\ \psi_t(X, \nu)$
  $\tilde{y}_{t+1} \leftarrow y_t - \frac{1}{\rho}(b - \nu_{t+1} - \mathcal{A}X_{t+1})$
  if $\beta(f(y_t) - \hat{f}_t(\tilde{y}_{t+1})) \le f(y_t) - f(\tilde{y}_{t+1})$ **then**
    $y_{t+1} \leftarrow \tilde{y}_{t+1}$ `// descent step`
  else
    $y_{t+1} \leftarrow y_t$ `// null step`
  end if
  Update $\hat{f}_{t+1}$ and $\widehat{\mathcal{X}}_{t+1}$ using (7), (8), (12), (13).
end for
return X_T, y_T

The complete algorithm is detailed in Algorithm 1. Notice that USBS has no iteration-dependent step-sizes or parameters, making it more amenable to effectively utilize a warm-start initialization. It is important to note that the main cost of computing $f(y_t)$ and $f(\tilde{y}_{t+1})$ is a maximum eigenvalue computation of $C - \mathcal{A}^* y_t$ and $C - \mathcal{A}^*\tilde{y}_{t+1}$, respectively. Lastly, we compute $\hat{f}_t(\tilde{y}_{t+1})$ using a primal-dual interior point method derived in Appendix B.1.

Convergence Rate. Under any setting of the parameters, Theorem 3.1 guarantees sublinear convergence for both primal and dual problems, showing that the iterates X_t and y_t converge in terms of objective gap and feasilibity.

Theorem 3.1. *Suppose strong duality holds. For any fixed $\rho > 0, \beta \in (0, 1), k_c \ge 1$, and $k_p \ge 0$, USBS produces iterates $X_t \succeq 0$ and $y_t \in \mathcal{Y}$ such that for any $\varepsilon \in (0, 1]$,*

$$\begin{aligned} \textit{penalized dual optimality: } & f(y_t) - f(y_\star) \le \varepsilon, \\ \textit{primal feasiblity: } & \operatorname{dist}(\mathcal{A}X_t, \mathcal{K}) \le \sqrt{\varepsilon}, \\ \textit{dual feasiblity: } & \lambda_{\max}(C - \mathcal{A}^* y_t) \le \varepsilon, \\ \textit{primal-dual optimality: } & |\langle b, y_t\rangle - \langle C, X_t\rangle| \le \sqrt{\varepsilon}, \end{aligned}$$

by some iteration $\mathcal{O}(1/\varepsilon^3)$. And, if strict complementarity holds, then these conditions are achieved by some iteration $\mathcal{O}(1/\varepsilon)$. Additionally, if Slater's condition holds and $y_\star$ is unique, then the approximate primal feasibility and approximate primal-dual optimality respectively improve to

$$\operatorname{dist}(\mathcal{A}X_t, \mathcal{K}) \le \varepsilon \quad \textit{and} \quad |\langle b, y_t\rangle - \langle C, X_t\rangle| \le \varepsilon.$$

The proof is of this theorem is given in Appendix D.

4. Scaling with Matrix Sketching

The memory required to store the primal iterates $\bar{X}_t$ (similarly X_t) is prohibitive to scaling SDPs to large problem instances. We will utilize a *Nyström sketch* (Tropp et al., 2017; Gittens, 2013; Halko et al., 2011; Li et al., 2017) to track a compressed low-rank projection of the primal iterate as it evolves which at any iteration can be used to compute a provably accurate low-rank approximation of $\bar{X}_t$.

First, notice that the operations carried out by USBS do not require explicitly storing $\bar{X}_t$. In fact, if we are not interested in the primal iterates and we only want to solve the dual problem (e.g. in the case where we want to test the feasibility of the primal problem), we only need to store $\langle C, \bar{X}_t\rangle$, $\operatorname{tr}(\bar{X}_t)$, and $\mathcal{A}\bar{X}_t$, which can be efficiently maintained with low-rank updates to $\bar{X}_t$ (see Appendix C for more details). This means if we are interested in the primal iterates, our storage of $\bar{X}_t$ is independent from the operations of USBS.

Consider any iterate $\bar{X}_t \in \mathbb{S}^n_+$. Let $r \in \{1, \ldots, n\}$ be a parameter that trades off accuracy for scalability. To construct the Nyström sketch, we sample a random projection matrix $\Psi \in \mathbb{R}^{n\times r}$ such that $\Psi_{ij} \sim \mathcal{N}(0, 1)$ are sampled i.i.d. The projection of $\bar{X}_t$, or *sketch*, can be computed as $P_t = \bar{X}_t\Psi \in \mathbb{R}^{n\times r}$. We can efficiently maintain this sketch P_t under low-rank updates made to $\bar{X}_t$,

$$\begin{aligned} P_{t+1} &\leftarrow \Big(\eta_{t+1}\bar{X}_t + V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\Big)\Psi \\ &= \eta_{t+1} P_t + V_t Q_{\underline{c}} \Lambda_{\underline{c}}\Big(Q_{\underline{c}}^\top V_t^\top \Psi\Big). \end{aligned} \tag{18}$$

Given the sketch P_t and the projection matrix Ψ, we can compute a rank-r approximation to $\bar{X}_t$. It is important to note that by applying this Nyström sketching technique the only approximation error is in the approximation of $\bar{X}_t$, and all of the error is constant (i.e. the error does not compound over time) and comes only from the projection matrix Ψ (and the choice of r). See Appendix C for more details.

5. Experiments

We evaluate USBS on three different problem types with and without warm-starting strategies against the scalable semidefinite programming algorithm CGAL (Yurtsever et al., 2019; 2021) (with sketching). Yurtsever et al. (2021) perform an extensive evaluation against strong SDP solvers, including SeDuMi (Sturm, 1999), SDPT3 (Toh et al., 1999), Mosek (ApS, 2019), SDPNAL+(Yang et al., 2015), and the BM factorization heuristic (Burer & Monteiro, 2003). They show on several applications that CGAL with sketching is by far the standard against which to compare for scalable semidefinite programming.

We consider the primal iterate an ε-approximate solution if the relative primal objective suboptimality and relative

Table 1: MaxCut data instance statistics.

	fe_sphere	hi2010	fe_body	me2010	fe_tooth	598a	144	auto	netherlands_osm	333SP
n	16K	25K	45K	70K	78K	111K	145K	449K	2.2M	3.7M
$\text{nnz}(L)$	115K	149K	372K	405K	983K	1.6M	2.3M	7.1M	7.1M	25.9M

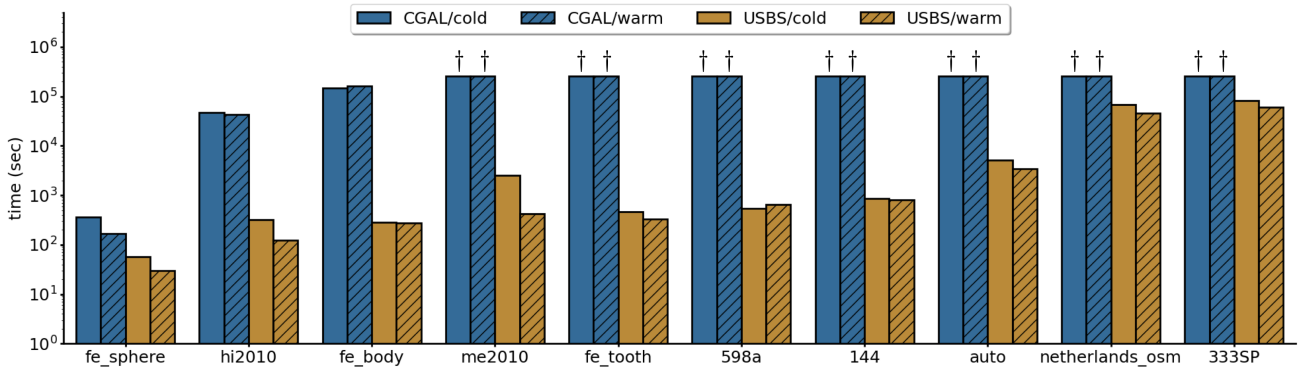


Figure 1: **Convergence time (sec) to moderate relative error tolerance (↓).** The time (in seconds) for CGAL and USBS to achieve an ε-approximate solution for $\varepsilon = 10^{-1}$ with and without warm-starting on 99% of initial data on ten `DIMACS10` MaxCut instances. The bars marked with † indicate an ε-approximate solution was not achieved in 72 hours. The datasets are sorted in ascending order by n, ranging from 16K to 3.7M (more than 10^{13} decision variables for 333SP). Note that warm-starting generally improves convergence, but does not always (e.g. 598a). We observe USBS achieves an extraordinary improvement in convergence over CGAL which fails to reach an accurate solution on 7 out of 10 instances. In contrast, USBS is able to reach a solution on all of the problem instances in 28 hours or less without a warm-start initialization.

infeasibility is less than ε, i.e.

$$\frac{\langle C, X\rangle - \langle C, X_\star\rangle}{1 + \langle C, X\rangle} \leq \varepsilon \quad \text{and} \quad \frac{\operatorname{dist}(\mathcal{A}X, \mathcal{K})}{1 + \|b\|} \leq \varepsilon. \quad (19)$$

Computing the relative infeasibility is simple. We do not usually have access to the optimal primal objective value $\langle C, X_\star\rangle$ to compute the relative primal objective suboptimality, but we can compute an upper bound. See (21) for how we compute this upper bound.

Following (Yurtsever et al., 2021), we scale the problem data for all problems such that the following condition holds,

$$\|C\|_{\mathrm{F}} = \operatorname{tr}(X_\star) = 1. \quad (20)$$

In all three problem types, the constraints exactly determine the trace of all optimal solutions. Since we know the trace of the optimal solution and we apply problem scaling (20), we can set $\alpha = 2$ for all problem types.

Computing the relative infeasibility is simple. As for the relative primal objective suboptimality, we do not usually have access to the optimal primal objective value $\langle C, X_\star\rangle$, but we can compute an upper bound. For any $y \in \mathcal{Y}$,

$$\begin{aligned} f(y) &= \alpha \max\{\lambda_{\max}(C - \mathcal{A}^* y), 0\} + \langle b, y\rangle \\ &\geq \langle C - \mathcal{A}^* y, X_\star\rangle - \langle b, y\rangle = \langle C, X_\star\rangle, \quad (21) \\ \implies \langle C, X\rangle - \langle C, X_\star\rangle &\leq \langle C, X\rangle - f(y). \end{aligned}$$

We use this upper bound to approximate the relative primal objective suboptimality in our experiments for USBS.

We use the upper bound given in (Yurtsever et al., 2021) to compute the relative primal objective suboptimality for CGAL.

For both CGAL and USBS, we compute eigenvectors and eigenvalues using an implementation of the thick-restart Lanczos algorithm (Wu et al., 1999; Hernández et al., 2007) with 32 inner iterations and a maximum of 10 restarts. Both CGAL and USBS are implemented using 64-bit floating point arithmetic. Unless otherwise specified, we use a CPU machine with 16 cores and up to 128 GB of RAM. Throughout the presentation of the experimental results we use ↑ to indicate that *higher is better* for that particular metric and ↓ to indicate *lower is better* for that particular metric.

We include further experimental details in Appendix E and results in Appendix F.

5.1. MaxCut

The MaxCut problem is a fundamental combinatorial optimization problem and its SDP relaxation is a common test bed for SDP solvers. Given an undirected graph, the MaxCut is a partitioning of the n vertices into two sets such that the number of edges between the two subsets is maximized. The MaxCut SDP relaxation (Goemans & Williamson, 1995) is as follows

$$\max \ \frac{1}{4}\operatorname{tr}(LX) \ \text{ s.t. } \ \operatorname{diag}(X) = \mathbf{1} \text{ and } X \succeq 0, \quad (22)$$

where L is the graph Laplacian, $\mathrm{diag}(\cdot)$ extracts the diagonal of a matrix into a vector, and $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. In many instances of MaxCut, the graph Laplacian L is sparse and the optimal solution is low-rank. This means that we can represent the MaxCut instance with far less than $\mathcal{O}(n^2)$ memory and leverage sketching the primal matrix X to improve scalability temendously.

We evaluate the CGAL and USBS with and without warm-starting on ten instances from the `DIMACS10` (Bader et al.) where n and the number of edges in each graph are shown in Table 1. We warm-start each method by dropping the last 1% of vertices from the graph, resulting in a graph with 99% of the vertices. We pad the solution from the 99%-sized problem with zeros and rescale as necessary to create the warm-start initialization. Figure 1 displays the time (in seconds) taken by CGAL and USBS, with and without warm-starting on each of the ten instances, where $r = 10$.

5.2. Quadratic Assignment Problem

The quadratic assignment problem (QAP) is a very difficult but fundamental class of combinatorial optimization problems containing the traveling salesman problem, max-clique, bandwidth problems, and facilities location problems among others (Loiola et al., 2007). SDP relaxations have been shown to facilitate finding good solutions to large QAPs (Zhao et al., 1998). For any QAP instance, the goal is to optimize an assignment matrix Π which aligns a weight matrix $W \in \mathbb{S}^{\mathtt{n}}$ and distance matrix $D \in \mathbb{S}^{\mathtt{n}}$. The number of $\mathtt{n} \times \mathtt{n}$ assignment matrices is $\mathtt{n}!$, so a brute-force search becomes quickly intractable as $\mathtt{n}$ grows. Generally, QAP instances with $\mathtt{n} > 30$ are intractable to solve exactly.

There are many SDP relaxtions for QAPs, but we consider the one presented by (Yurtsever et al., 2021) and inspired by (Huang et al., 2014; Bravo Ferreira et al., 2018) which is

$$\begin{aligned} \min \;& \mathrm{tr}\big((D \otimes W)\,\mathsf{Y}\big) \\ \text{s.t.} \;& \mathrm{tr}_1(\mathsf{Y}) = I,\; \mathrm{tr}_2(\mathsf{Y}) = I,\; \mathcal{G}(\mathsf{Y}) \geq 0, \\ & \mathrm{vec}(\mathsf{B}) = \mathrm{diag}(\mathsf{Y}),\; \mathsf{B}\mathbf{1} = \mathbf{1},\; \mathbf{1}^\top \mathsf{B} = \mathbf{1}^\top,\; \mathsf{B} \geq 0, \\ & X := \begin{bmatrix} 1 & \mathrm{vec}(\mathsf{B})^\top \\ \mathrm{vec}(\mathsf{B}) & \mathsf{Y} \end{bmatrix} \succeq 0,\; \mathrm{tr}(\mathsf{Y}) = \mathtt{n} \end{aligned} \tag{23}$$

where $\otimes$ denotes the Kronecker product, $\mathrm{tr}_1(\cdot)$ and $\mathrm{tr}_2(\cdot)$ denote the partial trace over the first and second systems of Kronecker product respectively, $\mathcal{G}(\mathsf{Y})$ extracts the entries of Y corresponding to the nonzero entries of $D \otimes W$, and $\mathrm{vec}(\cdot)$ stacks the columns of a matrix one on top of the other to form a vector. In many cases, one of D or W is sparse (i.e. $\mathcal{O}(n)$ nonzero entries) resulting in $\mathcal{O}(\mathtt{n}^3)$ total constraints for the SDP.

The primal variable X has dimension $(\mathtt{n}^2+1)\times(\mathtt{n}^2+1)$ and as a result the SDP relaxation has $\mathcal{O}(\mathtt{n}^4)$ decision variables.

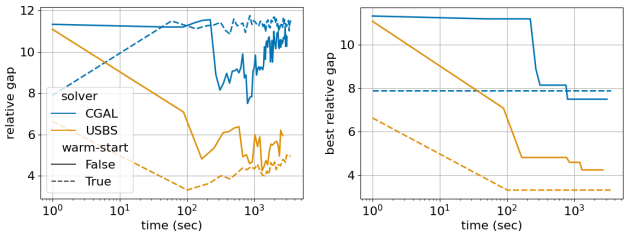


Figure 2: **`relative gap` (↓) vs. time.** We plot the `relative gap` (y-axis, left) and `best relative gap` (y-axis, right) against time in seconds (x-axis) for one QAP instance, pr144, from `TSPLIB` ($\mathtt{n} = 144$) over one hour of optimization. We observe that for both algorithms the best rounded solution is found early in optimization. We observe that USBS is able to leverage a warm-start initialization.

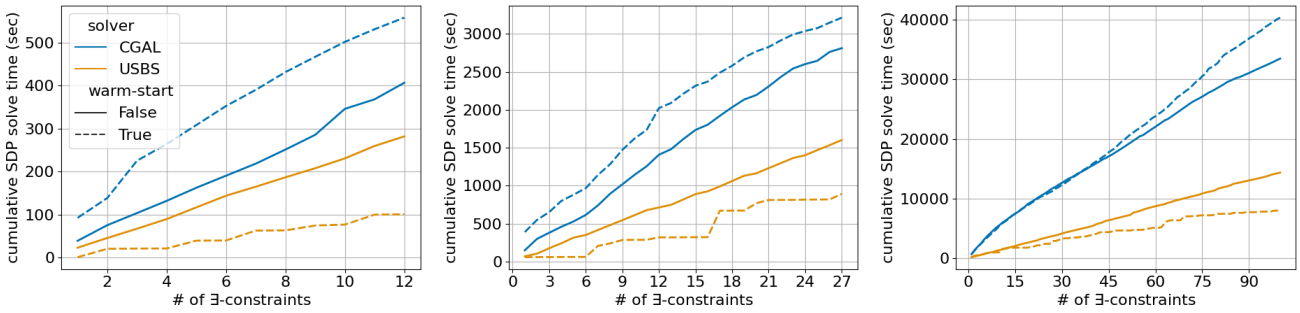
Given the aggressive growth in complexity, most SDP based algorithms have difficulty operating on instances where $\mathtt{n} > 50$. To reduce the number of decision variables, we sketch X with $r = \mathtt{n}$, resulting in $\mathcal{O}(\mathtt{n}^3)$ entries in the sketch and the same number of constraints in most natural instances. We show that this enables CGAL and USBS to scale to instances where $\mathtt{n} = 198$ (1.5 billion decision variables).

We evaluate CGAL and USBS on select large instances from `QAPLIB` (Burkard et al., 1997) and `TSPLIB` (Reinelt, 1995) ranging in size from 136 to 198. Provided with each of these QAP instances is the known optimum. For many instances, we find that the quality of the permutation matrix produced by the rounding procedure does not entirely correlate with the quality of the iterates with respect to the SDP (23). Thus, we apply the rounding procedure at every iteration of both CGAL and USBS. Our metric for evaluation is `relative gap` which is computed as follows

$$\texttt{relative gap} = \frac{\texttt{upper bound obtained} - \texttt{optimum}}{\texttt{optimum}}.$$

We report the lowest value so far of `relative gap` as `best relative gap`. CGAL (Yurtsever et al., 2021) is shown to obtain significantly smaller `best relative gap` than CSDP (Bravo Ferreira et al., 2018) and PATH (Zaslavskiy et al., 2008) on most instances, and thus, is a strong baseline for achieving high quality approximate solutions.

To create a warm-start initialization, we create a slightly smaller QAP by dropping the final row and column of both D and W (i.e. solve a size $\mathtt{n} - 1$ subproblem of the original instance). We use the solution to slightly smaller problem to set a warm-start initialization for the original problem, rescaling and padding with zeros where necessary. We stopped optimizing after one hour. When warm-starting, we optimize the slightly smaller problem for one hour and then optimize the original problem for one hour. Figure 2 plots the `relative gap` and `best relative`



(a) PubMed (b) QIAN (c) SCAD-zbMATH

Figure 3: **Cumulative SDP solve time (↓) vs. number of ∃-constraints**. In each plot (one for each author coreference dataset), the x-axis is the number of ∃-constraints generated one after the other over time and the y-axis is the the cumulative solve time (in seconds) for each SDP solver to reach a relative suboptimality, relative infeasibility, and max absolute infeasibility (i.e. $\|\mathcal{A}X - \text{proj}_{\mathcal{K}}(\mathcal{A}X)\|_\infty \leq \varepsilon$) of $\varepsilon = 10^{-1}$. When warm-starting, both solvers are initialized using the solution from the previous SDP (with one less ∃-constraint). ∃-constraints are generated until the perfect clustering is predicted. We observe that USBS is able to leverage a warm-start initialization. In addition, we observe that the performance gap between USBS and CGAL grows as the problem size grows. See Table 2 for dataset sizes and details.

`gap` against time for CGAL and USBS both with and without warm-starting for one `TSPLIB` instance.

5.3. Interactive Entity Resolution with ∃-constraints

Angell et al. (2022) introduced a novel SDP relaxation of a combinatorial optimization problem that arises in automatic knowledge base construction (Uhlen et al., 2010; Krishnamurthy, 2015; Iv et al., 2022). They consider the problem of interactive entity resolution with user feedback to correct predictions made by a machine learning algorithm. The human feedback is transformed into a logical constraint on the output entity resolution decisions. Specifically, they introduce ∃-*constraints*, a new paradigm of interactive feedback for correcting entity resolution predictions and presented a novel SDP relaxation as part of a heuristic algorithm for satisfying ∃-constraints in the predicted entity resolution decisions. This novel form of feedback allows users to specify constraints stating the existence of an entity with and without certain features. As each ∃-constraint is added to the optimization problem additional constraints are added to the SDP relaxation. We use the solution to the SDP without the newest ∃-constraint to warm-start the solver with the new ∃-constraint added. See Appendix E.3 for more details.

Table 2: Author coreference dataset statistics.

	PubMed	QIAN	SCAD-zbMATH
# mentions	315	410	1,196
# blocks	5	38	120
# clusters	34	77	166
nnz(W)	3,973	5,158	18,608
# features	14,093	10,366	8,203

We evaluate the performance of the CGAL and USBS with and without warm-starting on the same three author coreference datasets used in (Angell et al., 2022). In these datasets, each mention is an author-paper pair and the goal is to identify which author-paper pairs refer to the same author. We simulate ∃-constraint generation using the same oracle implemented in (Angell et al., 2022). Figure 3 shows the cumulative SDP solve time to reach an ε-approximate solution against the number of ∃-constraints for each of the three datasets until the perfect entity resolution decisions were predicted by the inference algorithm.

6. Limitations

The intended use for USBS is for solving SDPs with (approximately) low-rank solutions, particularly when matrix sketching is used. We should not necessarily expect USBS to be fast and scalable for *all* SDPs. For example, USBS should not be expected to efficiently solve SDPs instances where the cost matrix C is dense or the number of constraints m is on the order of $\mathcal{O}(n^2)$ or greater. It is important to note that convergence speed is dependent on n and m which are suppressed by $\mathcal{O}(\cdot)$ in the convergence rate. This is also related to the reason solving inequality-constrained SDPs written in standard-form is so inefficient. Moreover, USBS may struggle with important applications such as sum-of-squares or other polynomial optimization problems which have a large number of constraints, but we leave this investigation to future work. Finally, we emphasize USBS is generally intended to solve SDPs to relatively low accuracy. This is often perfectly reasonable for applications where we seek a rounded solution for combinatorial optimization problems, but may not be best for every application.

7. Related Work

Efficient SDP solving. Due to their wide-spread applicability and importance, methods for solving SDPs efficiently have been extensively studied in the literature (Todd, 2001; Nesterov, 1989; Nesterov & Nemirovskii, 1994; Alizadeh, 1995; Boyd et al., 2011; Friedlander & Macedo, 2016; Yurtsever et al., 2019; 2015; Ding & Udell, 2021). Interior point methods (Helmberg, 1994; Nesterov & Nemirovskii, 1994; Alizadeh, 1995) are by far the most widely-used and well-known method for solving SDPs. Interior point methods enjoy quadratic convergence, but have incredibly high per-iteration complexity, and thus, are impractical for instances having more than a few hundred variables. ADMM (Boyd et al., 2011) and splitting-based methods (O'Donoghue et al., 2021) have also been applied to solving SDPs. These methods are generally more applicable than interior point methods, but our still restricted do to the computationally expensive full eigenvalue decomposition. First-order methods such as conditional gradient augmented Lagrangian (Yurtsever et al., 2017), primal-dual subgradient algorithms (Yurtsever et al., 2015), the matrix multiplicative weight method (Arora et al., 2005; Tsuda et al., 2005; Lee & Padmanabhan, 2020), and the mirror-prox algorithm with the quantum entropy mirror map (Nemirovski, 2004) have by far the lowest per-iteration complexity among all of the methods for solving SDPs. This lower per-iteration complexity leads to slower convergence rate for these first-order methods.

Spectral bundle methods. Helmberg & Rendl (2000) were the first to introduce spectral bundle methods for equality-constrained SDPs. The biggest difference between the original spectral bundle method and our approach is the model where $k_c = 1$ is fixed and k_p is chosen by the user. We find that larger k_c provides better convergence in general and k_p is not very helpful (and sometimes harmful) to convergence. Several other algorithmic variants of spectral bundle methods exist including solving inequality-constrained SDPs (Helmberg & Kiwiel, 2002), trust region-based methods for nonconvex eigenvalue optimization (Apkarian et al., 2008), and incorporating second-order information to speed up empirical convergence (Helmberg et al., 2014). Ding & Grimmer (2023) present a spectral bundle method most similar to ours for solving equality-constrained SDPs. They prove an impressive set of convergence rates under various assumptions, including linear convergence if certain strong assumptions hold. The biggest difference between previous work of spectral bundle methods and USBS is the fact that USBS has more flexibility and scalability as compared with other methods both in terms of the constraints USBS supports in the SDP and the model the user can specify. In addition, this work addresses many of the practical implementation questions enabling scalability to instances with multiple orders of magnitude more variables.

Memory-efficient semidefinte programming. The most widely-known memory efficient approach to semidefinite programming is the Burer–Monteiro (BM) factorization heuristic (Burer & Monteiro, 2003). Several optimization techniques have been used to optimize the low-rank factorized problem (Wang et al., 2017; Boumal et al., 2014; Burer & Monteiro, 2005; Kulis et al., 2007; Sahin et al., 2019; Souto et al., 2022). Ding et al. (2021) present an optimal storage approach to solving SDPs by first approximately solving the dual problem and then using the dual slack matrix to solve the primal problem efficiently. Yurtsever et al. (2021) implement the conditional gradient augmented Lagrangian method (Yurtsever et al., 2019) with the matrix sketching technique in (Yurtsever et al., 2017) which forms a strong state-of-the-art method for scalably solving general SDPs. Ding & Grimmer (2023) implements a spectral bundle method for solving equality constrained SDPs with the same matrix sketching technique (Yurtsever et al., 2017).

8. Conclusion

In this work, we presented a practical spectral bundle method for solving large SDPs with both equality and inequality constraints. We proved non-asymptotic convergence rates under standard assumptions for USBS. We showed empirically that USBS is fast and scalable on practical SDP instances. Additionally, we showed that USBS can more reliably leverage a warm-start initialization to accelerate convergence. Lastly, we make our standalone implementation in pure JAX available for wide-spread use.

Impact Statement

This paper presents work whose goal is to advance the fields of Machine Learning and Optimization. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

This work is funded in part by the Center for Data Science and the Center for Intelligent Information Retrieval at the University of Massachusetts Amherst, and in part by the National Science Foundation under grants IIS-1763618 and IIS-1922090, and in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction, and in part by IBM Research AI through the AI Horizons Network. The work reported here was performed in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Rico Angell is partially supported by the National Science Foundation Graduate Research Fel-

lowship under grant 1938059. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor(s).

References

Agarwal, D., Angell, R., Monath, N., and McCallum, A. Entity linking and discovery via arborescence-based supervised clustering. *arXiv preprint arXiv:2109.01242*, 2021.

Agarwal, D., Angell, R., Monath, N., and McCallum, A. Entity linking via explicit mention-mention coreference modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.

Alizadeh, F. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM journal on Optimization*, 1995.

Alizadeh, F., Haeberly, J.-P. A., and Overton, M. L. Complementarity and nondegeneracy in semidefinite programming. *Mathematical programming*, 1997.

Alizadeh, F., Haeberly, J.-P. A., and Overton, M. L. Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results. *SIAM Journal on Optimization*, 1998.

Angell, R., Monath, N., Mohan, S., Yadav, N., and McCallum, A. Clustering-based inference for biomedical entity linking. *arXiv preprint arXiv:2010.11253*, 2020.

Angell, R., Monath, N., Yadav, N., and Mccallum, A. Interactive correlation clustering with existential cluster constraints. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Apkarian, P., Noll, D., and Prot, O. A trust region spectral bundle method for nonconvex eigenvalue optimization. *SIAM Journal on Optimization*, 2008.

ApS, M. Mosek optimization toolbox for matlab. *User's Guide and Reference Manual, Version*, 2019.

Arora, S., Hazan, E., and Kale, S. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, 2005.

Bader, D. A., Meyerhenke, H., Sanders, P., and Wagner, D. Graph partitioning and graph clustering. *10th DIMACS Implementation Challenge Workshop. February 13-14, 2012.* URL https://www.cise.ufl.edu/research/sparse/matrices/DIMACS10/index.html.

Bae, J., Helldin, T., Riveiro, M., Nowaczyk, S., Bouguelia, M.-R., and Falkman, G. Interactive clustering: A comprehensive review. *ACM Computing Surveys (CSUR)*, 2020.

Basu, S., Fisher, D., Drucker, S., and Lu, H. Assisting users with clustering tasks by combining metric learning and classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

Beck, A. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization*, 2015.

Bertsimas, D. The achievable region method in the optimal control of queueing systems; formulations, bounds and policies. *Queueing systems*, 1995.

Binette, O. and Steorts, R. C. (Almost) all of entity resolution. *Science Advances*, 2022.

Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 2014.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 2011.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., et al. Jax: composable transformations of python+ numpy programs. 2018.

Bravo Ferreira, J. F., Khoo, Y., and Singer, A. Semidefinite programming approach for the quadratic assignment problem with a sparse graph. *Computational Optimization and Applications*, 2018.

Burer, S. and Monteiro, R. D. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical programming*, 2003.

Burer, S. and Monteiro, R. D. Local minima and convergence in low-rank semidefinite programming. *Mathematical programming*, 2005.

Burkard, R. E., Karisch, S. E., and Rendl, F. Qaplib–a quadratic assignment problem library. *Journal of Global optimization*, 1997.

Chuang, J. and Hsu, D. J. Human-centered interactive clustering for data analysis. In *Conference on Neural Information Processing Systems (NIPS). Workshop on Human-Propelled Machine Learning*, 2014.

Cifuentes, D. and Moitra, A. Polynomial time guarantees for the Burer-Monteiro method. *Advances in Neural Information Processing Systems*, 35:23923–23935, 2022.

Coden, A., Danilevsky, M., Gruhl, D., Kato, L., and Nagarajan, M. A method to accelerate human in the loop clustering. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 237–245. SIAM, 2017.

Dathathri, S., Dvijotham, K., Kurakin, A., Raghunathan, A., Uesato, J., Bunel, R. R., Shankar, S., Steinhardt, J., Goodfellow, I., Liang, P. S., et al. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *Advances in Neural Information Processing Systems*, 2020.

Díaz, M. and Grimmer, B. Optimal convergence rates for the proximal bundle method. *SIAM Journal on Optimization*, 2023.

Ding, L. and Grimmer, B. Revisiting spectral bundle methods: Primal-dual (sub) linear convergence rates. *SIAM Journal on Optimization*, 2023.

Ding, L. and Udell, M. On the simplicity and conditioning of low rank semidefinite programs. *SIAM Journal on Optimization*, 2021.

Ding, L., Yurtsever, A., Cevher, V., Tropp, J. A., and Udell, M. An optimal-storage approach to semidefinite programming using approximate complementarity. *SIAM Journal on Optimization*, 2021.

Drusvyatskiy, D. and Wolkowicz, H. The many faces of degeneracy in conic optimization. *Foundations and Trends® in Optimization*, 2017.

Dubey, A., Bhattacharya, I., and Godbole, S. A cluster-level semi-supervision model for interactive clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010.

Feltenmark, S. and Kiwiel, K. C. Dual applications of proximal bundle methods, including lagrangian relaxation of nonconvex problems. *SIAM Journal on Optimization*, 2000.

Friedlander, M. P. and Macedo, I. Low-rank spectral optimization via gauge duality. *SIAM Journal on Scientific Computing*, 2016.

Frome, A., Singer, Y., Sha, F., and Malik, J. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *2007 IEEE 11th International Conference on Computer Vision*, 2007.

Frostig, R., Johnson, M. J., and Leary, C. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 2018.

Gittens, A. *Topics in randomized numerical linear algebra*. California Institute of Technology, 2013.

Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 1995.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 2011.

Helmberg, C. An interior point method for semidefinite programming and max-cut bounds. 1994.

Helmberg, C. and Kiwiel, K. C. A spectral bundle method with bounds. *Mathematical Programming*, 2002.

Helmberg, C. and Rendl, F. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 2000.

Helmberg, C., Overton, M. L., and Rendl, F. The spectral bundle method with second-order information. *Optimization Methods and Software*, 2014.

Hernández, V., Román, J. E., Tomás, A., and Vidal, V. Krylov-schur methods in slepc. *Universitat Politecnica de Valencia, Tech. Rep. STR-7*, 2007.

Huang, Q., Chen, Y., and Guibas, L. Scalable semidefinite relaxation for maximum a posterior estimation. In *International Conference on Machine Learning*. PMLR, 2014.

Iv, R., Passos, A., Singh, S., and Chang, M.-W. Fruit: Faithfully reflecting updated information in text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.

Jonker, R. and Volgenant, T. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*. Springer, 1988.

Karp, R. M. *Reducibility among Combinatorial Problems*, pp. 85–103. Springer US, 1972.

Kiwiel, K. C. Efficiency of proximal bundle methods. *Journal of Optimization Theory and Applications*, 2000.

Klein, D., Kamvar, S. D., and Manning, C. D. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, 2002.

Krishnamurthy, J. Learning to understand natural language with less human effort. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2015.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

Kulis, B., Surendran, A. C., and Platt, J. C. Fast low-rank semidefinite programming for embedding and clustering. In *Artificial Intelligence and Statistics*. PMLR, 2007.

Kulis, B., Basu, S., Dhillon, I., and Mooney, R. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 2009.

Lee, Y. T. and Padmanabhan, S. An $\widetilde{\gamma}(m/\varepsilon^{3.5})$-cost algorithm for semidefinite programs with diagonal constraints. In Abernethy, J. and Agarwal, S. (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, Proceedings of Machine Learning Research. PMLR, 2020.

Lemarechal, C., Strodiot, J.-J., and Bihain, A. On a bundle algorithm for nonsmooth optimization. In *Nonlinear programming 4*. Elsevier, 1981.

Li, H., Linderman, G. C., Szlam, A., Stanton, K. P., Kluger, Y., and Tygert, M. Algorithm 971: An implementation of a randomized algorithm for principal component analysis. *ACM Transactions on Mathematical Software (TOMS)*, 2017.

Li, Z., Liu, J., and Tang, X. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th international conference on Machine learning*, 2008.

Li, Z., Liu, J., and Tang, X. Constrained clustering via spectral regularization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Loiola, E. M., De Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P., and Querido, T. A survey for the quadratic assignment problem. *European journal of operational research*, 2007.

Lu, Z. and Carreira-Perpinan, M. A. Constrained spectral clustering through affinity propagation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2008.

Luo, Z.-Q. and Tseng, P. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 1993.

Mazumdar, A. and Saha, B. Clustering with noisy queries, 2017.

Mifflin, R. An algorithm for constrained optimization with semismooth functions. *Mathematics of Operations Research*, 1977.

Mirsky, L. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 1960.

Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 1957.

Nemirovski, A. Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 2004.

Nesterov, J. E. Self-concordant functions and polynomial-time methods in convex programming. *Report, Central Economic and Mathematic Institute, USSR Acad. Sci*, 1989.

Nesterov, Y. and Nemirovskii, A. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. SCS: Splitting Conic Solver, version 3.1.1. https://github.com/cvxgrp/scs, November 2021.

Overton, M. L. Large-scale optimization of eigenvalues. *SIAM Journal on Optimization*, 1992.

Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. *Advances in neural information processing systems*, 2018.

Reinelt, G. Tsplib95. *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Heidelberg*, 1995. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Rosen, D. M., Carlone, L., Bandeira, A. S., and Leonard, J. J. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 2019.

Sahin, M. F., Alacaoglu, A., Latorre, F., Cevher, V., et al. An inexact augmented lagrangian framework for nonconvex optimization with nonlinear constraints. *Advances in Neural Information Processing Systems*, 2019.

Sato, Y. and Iwayama, M. Interactive constrained clustering for patent document set. In *Proceedings of the 2nd international workshop on Patent information retrieval*, pp. 17–20, 2009.

Schacke, K. On the kronecker product. *Master's thesis, University of Waterloo*, 2004.

Shental, N., Bar-Hillel, A., Hertz, T., and Weinshall, D. Computing gaussian mixture models with em using equivalence constraints. *Advances in neural information processing systems*, 2004.

Souto, M., Garcia, J. D., and Veiga, Á. Exploiting low-rank structure in semidefinite programming by approximate operator splitting. *Optimization*, 2022.

Spielman, D. A. and Srivastava, N. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008.

Sremac, S., Woerdeman, H. J., and Wolkowicz, H. Error bounds and singularity degree in semidefinite programming. *SIAM Journal on Optimization*, 2021.

Sturm, J. F. Using SeDuMi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 1999.

Sturm, J. F. Error bounds for linear matrix inequalities. *SIAM Journal on Optimization*, 2000.

Subramanian, S., King, D., Downey, D., and Feldman, S. S2AND: A Benchmark and Evaluation System for Author Name Disambiguation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2021*, 2021.

Todd, M. J. Semidefinite optimization. *Acta Numerica*, 2001.

Todd, M. J., Toh, K.-C., and Tütüncü, R. H. On the nesterov–todd direction in semidefinite programming. *SIAM Journal on Optimization*, 1998.

Toh, K.-C., Todd, M. J., and Tütüncü, R. H. Sdpt3—a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 1999.

Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. Fixed-rank approximation of a positive-semidefinite matrix from streaming data. *Advances in Neural Information Processing Systems*, 2017.

Tsuda, K., Rätsch, G., and Warmuth, M. K. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 2005.

Uhlen, M., Oksvold, P., Fagerberg, L., Lundberg, E., Jonasson, K., Forsberg, M., Zwahlen, M., Kampf, C., Wester, K., Hober, S., et al. Towards a knowledge-based human protein atlas. *Nature biotechnology*, 2010.

Vandenberghe, L. and Boyd, S. Applications of semidefinite programming. *Applied Numerical Mathematics*, 1999.

Vandenberghe, L., Boyd, S., and El Gamal, A. Optimizing dominant time constant in RC circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1998a.

Vandenberghe, L., Boyd, S., and Wu, S.-P. Determinant maximization with linear matrix inequality constraints. *SIAM journal on matrix analysis and applications*, 1998b.

Vikram, S. and Dasgupta, S. Interactive bayesian hierarchical clustering. In *International Conference on Machine Learning*, 2016.

Viswanathan, V., Gashteovski, K., Lawrence, C., Wu, T., and Neubig, G. Large language models enable few-shot clustering. *arXiv preprint arXiv:2307.00524*, 2023.

Wagstaff, K. and Cardie, C. Clustering with instance-level constraints. *AAAI/IAAI*, 2000.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. Constrained k-means clustering with background knowledge. In *Icml*, 2001.

Waldspurger, I. and Waters, A. Rank optimality for the burer–monteiro factorization. *SIAM journal on Optimization*, 2020.

Wang, P.-W., Chang, W.-C., and Kolter, J. Z. The mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints. *arXiv preprint arXiv:1706.00476*, 2017.

Wu, K., Canning, A., Simon, H., and Wang, L.-W. Thick-restart lanczos method for electronic structure calculations. *Journal of Computational Physics*, 1999.

Xing, E., Jordan, M., Russell, S. J., and Ng, A. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 2002.

Xiong, C., Johnson, D. M., and Corso, J. J. Active clustering with model-based uncertainty reduction. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

Yadav, N., Monath, N., Angell, R., and McCallum, A. Event and entity coreference using trees to encode uncertainty in joint decisions. In *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, 2021.

Yang, L., Sun, D., and Toh, K.-C. Sdpnal+: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 2015.

Yurtsever, A., Tran Dinh, Q., and Cevher, V. A universal primal-dual convex optimization framework. *Advances in Neural Information Processing Systems*, 2015.

Yurtsever, A., Udell, M., Tropp, J., and Cevher, V. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Artificial intelligence and statistics*. PMLR, 2017.

Yurtsever, A., Fercoq, O., and Cevher, V. A conditional-gradient-based augmented lagrangian framework. In *International Conference on Machine Learning*, 2019.

Yurtsever, A., Tropp, J. A., Fercoq, O., Udell, M., and Cevher, V. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 2021.

Zaslavskiy, M., Bach, F., and Vert, J.-P. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

Zhao, Q., Karisch, S. E., Rendl, F., and Wolkowicz, H. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 1998.

A. Derivations

A.1. Penalized Dual Objective

Begin by considering the following Lagrangian formulation of the model problem

$$p_\star = \max_{X \in \mathbb{S}^n_+} \min_{y \in \mathbb{R}^m : y_{\mathcal{I}} \geq 0} \mathcal{L}(X, y) = \min_{y \in \mathbb{R}^m : y_{\mathcal{I}} \geq 0} \max_{X \in \mathbb{S}^n_+} \mathcal{L}(X, y) = d_\star, \tag{A.1}$$

where the Lagrangian is defined as

$$\mathcal{L}(X, y) := \langle C - \mathcal{A}^* y, X \rangle + \langle b, y \rangle. \tag{A.2}$$

Since $\mathcal{X}_\star \subset \mathcal{X}$, we know

$$\max_{X \in \mathbb{S}^n_+} \min_{y \in \mathcal{Y}} \mathcal{L}(X, y) = \max_{X \in \mathcal{X}} \min_{y \in \mathcal{Y}} \mathcal{L}(X, y) = \min_{y \in \mathcal{Y}} \max_{X \in \mathcal{X}} \mathcal{L}(X, y). \tag{A.3}$$

Furthermore, it is easy to show the following fact (Overton, 1992),

$$\max_{X \succeq 0\,,\, \operatorname{tr}(X) \leq 1} \langle Z, X \rangle = \max\{\lambda_{\max}(Z), 0\}. \tag{A.4}$$

Incorporating (A.4) with (A.3) allows us to write an equivalent formulation of the original dual problem by defining the penalized dual objective as shown in (pen-D)

$$f(y) := \alpha \max\{\lambda_{\max}(C - \mathcal{A}^* y), 0\} + \langle b, y \rangle + \iota_{\mathcal{Y}}(y), \tag{pen-D}$$

where $\iota_{\mathcal{Y}}(\cdot)$ is the indicator function defined such that $\iota_{\mathcal{Y}}(y) = 0$ if $y \in \mathcal{Y}$ and $\iota_{\mathcal{Y}}(y) = +\infty$ otherwise.

A.2. Equivalent Penalized Dual Objective

We begin by considering the dual form of the indicator function $\iota_{\mathcal{Y}}(\cdot)$

$$\iota_{\mathcal{Y}}(y) = \sup\{-\langle \nu, y \rangle : \nu_{\mathcal{I}} \leq 0, \nu_{\mathcal{I}'} = 0\}. \tag{A.5}$$

Define $\mathrm{N} := \{\nu \in \mathbb{R}^m : \nu_{\mathcal{I}} \leq 0, \nu_{\mathcal{I}'} = 0\}$. Substituting (A.5) into (pen-D) and utilizing (A.4) gives

$$f(y) = \sup_{(X, \nu) \in \mathcal{X} \times \mathrm{N}} \langle C - \mathcal{A}^* y, X \rangle + \langle b - \nu, y \rangle.$$

A.3. Candidate Iterate

Notice that the candidate iterate can be written as follows

$$\tilde{y}_{t+1} = \operatorname*{arg\,min}_{y \in \mathbb{R}^m} \sup_{(X, \nu) \in \widehat{\mathcal{X}}_t \times \mathrm{N}} F(X, \nu, y), \tag{A.6}$$

where

$$F(X, \nu, y) := \langle C - \mathcal{A}^* y, X \rangle + \langle b - \nu, y \rangle + \frac{\rho}{2} \|y - y_t\|^2.$$

To solve for $\tilde{y}_{t+1}$, start by fixing $(X, \nu) \in \widehat{\mathcal{X}}_t \times \mathrm{N}$ arbitrarily. Then, completing the square gives

$$\begin{aligned}
\operatorname*{arg\,min}_{y \in \mathbb{R}^m} F(X, \nu, y) &= \operatorname*{arg\,min}_{y \in \mathbb{R}^m} \langle b - \nu - \mathcal{A}X, y \rangle + \frac{\rho}{2} \|y - y_t\|^2 \\
&= \operatorname*{arg\,min}_{y \in \mathbb{R}^m} \frac{\rho}{2} \left\| y - y_t + \frac{1}{\rho}(b - \nu - \mathcal{A}X) \right\|^2 \\
&= y_t - \frac{1}{\rho}(b - \nu - \mathcal{A}X).
\end{aligned}$$

This implies that the candidate iterate can be computed as follows

$$\tilde{y}_{t+1} = \operatorname*{arg\,min}_{y \in \mathbb{R}^m} \hat{f}_t(y) + \|y - y_t\|^2 = y_t - \frac{1}{\rho}(b - \nu_{t+1} - \mathcal{A}X_{t+1}), \tag{A.7}$$

where $(X_{t+1}, \nu_{t+1}) \in \widehat{\mathcal{X}}_t \times \mathrm{N}$ is the solution to the following optimization problem

$$(X_{t+1}, \nu_{t+1}) \in \underset{(X,\nu)\in\widehat{\mathcal{X}}_t\times\mathrm{N}}{\arg\max}\ \psi_t(X,\nu), \tag{A.8}$$

where

$$\psi_t(X,\nu) := \langle C, X\rangle + \langle b-\nu-\mathcal{A}X, y_t\rangle - \frac{1}{2\rho}\|b-\nu-\mathcal{A}X\|^2. \tag{A.9}$$

B. Solving Iteration Subproblems

In this section, we detail the primal-dual path-following interior point methods used to solve the small semidefinite program to compute the value $\hat{f}_t(\tilde{y}_{t+1})$ and the small quadratic semidefinite program (16). Before we can derive the algorithms, it is necessary to define the svec operator and symmetric Kronecker product (Alizadeh et al., 1998; Schacke, 2004; Todd et al., 1998).

For any matrix $A \in \mathbb{S}^n$, the vector $\mathrm{svec}(A) \in \mathbb{R}^{\binom{n+1}{2}}$ is defined as

$$\mathrm{svec}(A) = \left[a_{11}, \sqrt{2}a_{21}, \ldots, \sqrt{2}a_{n1}, a_{22}, \sqrt{2}a_{32}, \ldots, \sqrt{2}a_{n2}, \ldots, a_{nn}\right]^\top.$$

The svec-operator is a structure preserving map between $\mathbb{S}^n$ and $\mathbb{R}^{\binom{n+1}{2}}$ where the constant $\sqrt{2}$ multiplied by some of the entries ensures that

$$\langle A, B\rangle = \mathrm{tr}(AB) = \mathrm{svec}(A)^\top \mathrm{svec}(B), \quad \forall\, A, B \in \mathbb{S}^n.$$

For any $M \in \mathbb{R}^{n\times n}$, let $\mathrm{vec}(M)$ be the map from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{n^2}$ defined by stacking the columns of M into a single n^2-dimensional vector. It is also useful to define the matrix $U \in \mathbb{R}^{\binom{n+1}{2}\times n^2}$ which maps $\mathrm{vec}(A) \mapsto \mathrm{svec}(A)$ for any $A \in \mathbb{S}^n$. Let $u_{ij,kl}$ be the entry in the row which defines element a_{ij} in $\mathrm{svec}(A)$ and the column that is multiplied with the element a_{kl} in $\mathrm{vec}(A)$. Then

$$u_{ij,kl} = \begin{cases} 1 & i=j=k=l \\ \frac{1}{\sqrt{2}} & i=k\neq j=l, \text{ or } i=l\neq j=k \\ 0 & \text{o.w.} \end{cases}$$

As an example, the case when $n = 2$:

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that U is a unique matrix with orthonormal rows and has the following property

$$U^\top U\,\mathrm{vec}(A) = U^\top \mathrm{svec}(A) = \mathrm{vec}(A), \quad \forall\, A \in \mathbb{S}^n.$$

The symmetric Kronecker product $\otimes_s$ can be defined for any two square matrices $G, H \in \mathbb{R}^{n\times n}$ by its action on a vector $\mathrm{svec}(A)$ for $A \in \mathbb{S}^n$ as follows

$$(G \otimes_s H)\,\mathrm{svec}(A) = \frac{1}{2}\mathrm{svec}(HAG^\top + GAH^\top).$$

Alternatively, but equivalently (Schacke, 2004), the symmetric Kronecker product can be defined more explicitly using the matrix U defined above as follows

$$G \otimes_s H = \frac{1}{2}U(G\otimes H + H \otimes G)U^\top,$$

where $\otimes$ is the standard Kronecker product. We use this latter definition in our implementation.

B.1. Computing $\hat{f}_t(\tilde{y}_{t+1})$

The value $\hat{f}_t(\tilde{y}_{t+1})$ is the optimum value of the following optimization problem (remember that ν can be dropped since $\tilde{y}_{t+1}$ is always feasible)

$$\begin{aligned} \max \quad & \langle C - \mathcal{A}^* \tilde{y}_{t+1}, \eta \bar{X}_t + V_t S V_t^\top \rangle + \langle b, \tilde{y}_{t+1} \rangle \\ \text{s.t.} \quad & \eta \geq 0 \\ & S \succeq 0 \\ & \eta + \operatorname{tr}(S) \leq \alpha \end{aligned}$$

This amounts to computing $\langle C - \mathcal{A}^* \tilde{y}_{t+1}, \eta_\star \bar{X}_t + V_t S_\star V_t^\top \rangle + \langle b, \tilde{y}_{t+1} \rangle$ where $(\eta_\star, S_\star)$ is a solution to the following (small) semidefinite program

$$\begin{aligned} \min \quad & g_1^\top \operatorname{svec}(S) + \eta\, g_2 \\ \text{s.t.} \quad & \eta \geq 0 \\ & S \succeq 0 \\ & 1 - v_I^\top \operatorname{svec}(S) - \eta \geq 0 \end{aligned} \tag{B.1}$$

where

$$\begin{aligned} g_1 &= \alpha \operatorname{svec}(V_t^\top (\mathcal{A}^* \tilde{y}_{t+1} - C) V_t), \\ g_2 &= \frac{\alpha}{\operatorname{tr}(\bar{X})} \langle \bar{X}_t, \mathcal{A}^* \tilde{y}_{t+1} - C) \rangle, \\ v_I &= \operatorname{svec}(I). \qquad (I \text{ is the } k \times k \text{ identity matrix}) \end{aligned}$$

We will use a primal-dual interior point method to solve (B.1). We follow the well known technique for deriving primal-dual interior point methods (Helmberg, 1994). We start by defining the Lagrangian of the dual barrier problem of (B.1),

$$\begin{aligned} L_\mu(S, \eta, T, \zeta, \omega) = g_1^\top \operatorname{svec}(S) + \eta g_2 - \operatorname{svec}(S)^\top \operatorname{svec}(T) - \eta\zeta \\ - \omega(1 - v_I^\top \operatorname{svec}(S) - \eta) + \mu(\log\det(T) + \log\zeta + \log\omega), \end{aligned} \tag{B.2}$$

where we introduce a dual slack matrix $T \succeq 0$ as complementary to S, a dual slack scalar $\zeta \geq 0$ as complementary to η, a Lagrange multiplier $\omega \geq 0$ for the trace constraint inequality, and a barrier parameter $\mu > 0$. Notice that we have moved from needing to optimize a constrained optimization problem to an unconstrained optimization problem. The saddle point solution of (B.2) is given by the solution of the KKT-conditions, which reduces to just the first-order optimality conditions since the problem is unconstrained. The first-order optimality conditions of (B.2) are the following system of equations

$$\nabla_S L_\mu = g_1 - \operatorname{svec}(T) + \omega v_I = 0 \tag{B.3}$$
$$\nabla_\eta L_\mu = g_2 - \zeta + \omega = 0 \tag{B.4}$$
$$\nabla_T L_\mu = S - \mu T^{-1} = 0 \tag{B.5}$$
$$\nabla_\zeta L_\mu = \eta - \mu\zeta^{-1} = 0 \tag{B.6}$$
$$\nabla_\omega L_\mu = 1 - v_I^\top \operatorname{svec}(S) - \eta - \mu\omega^{-1} = 0 \tag{B.7}$$

By the strict concavity of $\log\det T$, $\log\zeta$, and $\log\omega$, there exists a unique solution $(S_\mu, \eta_\mu, T_\mu, \zeta_\mu, \omega_\mu)$ to this system of equations for any value of the barrier parameter $\mu > 0$. The sequence of these solutions as $\mu \to 0$ forms the *central trajectory* (also known as the central *path*). For a point $(S, \eta, T, \zeta, \omega)$ on the central trajectory, we can use any combination of (B.5), (B.6), and/or (B.7) to solve for μ,

$$\mu = \frac{\langle S, T \rangle}{k} = \eta\zeta = \omega(1 - v_I^\top \operatorname{svec}(S) - \eta) = \frac{\langle S, T \rangle + \eta\zeta + \omega(1 - v_I^\top \operatorname{svec}(S) - \eta)}{k + 2}. \tag{B.8}$$

The fundamental idea of primal-dual interior point methods is to use Newton's method to follow the central path to a solution of (B.1). Before we can apply Newton's method, we must linearize the non-linear equations (B.5), (B.6), and (B.7) into an equivalent linear formulation. There are several ways one could linearize these equations and the choice of linearization

significantly impacts the algorithm's behavior (see (Alizadeh et al., 1998) or (Helmberg, 1994) for more on the choice of linearization). We choose one standard method linearization, detailed in the following system of equations

$$\mathrm{F}_\mu(\theta) = \mathrm{F}_\mu(S, \eta, T, \zeta, \omega) := \begin{pmatrix} g_1 - \mathrm{svec}(T) + \omega v_I \\ g_2 - \zeta + \omega \\ ST - \mu I \\ \eta\zeta - \mu \\ \omega(1 - v_I^\top \mathrm{svec}(S) - \eta) - \mu \end{pmatrix} = 0. \tag{B.9}$$

The solution $\theta_\star$ to this system of equations $\mathrm{F}_\mu(\theta) = 0$ satisfies the first-order optimality conditions (B.3)-(B.7) and is the optimal solution to the barrier problem. We utilize Newton's method to take steps in the update direction $\Delta\theta = (\Delta S, \Delta\eta, \Delta T, \Delta\zeta, \Delta\omega)$ towards $\theta_\star$. The update direction $\Delta\theta$ determined by Newton's method must satisfy the following equation

$$\mathrm{F}_\mu(\theta) + \nabla \mathrm{F}_\mu(\Delta\theta) = 0.$$

Hence, the update direction $\Delta\theta$ is the solution to the following system of equations (after the same standard linearization has been applied to the following system as in (B.9))

$$-\mathrm{svec}(\Delta T) + \Delta\omega\, v_I = \mathrm{svec}(T) - g_1 - \omega\, v_I \tag{B.10}$$

$$-\Delta\zeta + \Delta\omega = \zeta - g_2 - \omega \tag{B.11}$$

$$\omega^{-1}(1 - v_I^\top \mathrm{svec}(S) - \eta)\Delta\omega - v_I^\top \mathrm{svec}(\Delta S) - \Delta\eta = \mu\omega^{-1} + v_I^\top \mathrm{svec}(S) + \eta - 1 \tag{B.12}$$

$$(T \otimes_s S^{-1})\,\mathrm{svec}(\Delta S) + \mathrm{svec}(\Delta T) = \mu\,\mathrm{svec}(S^{-1}) - \mathrm{svec}(T) \tag{B.13}$$

$$\zeta\eta^{-1}\Delta\eta + \Delta\zeta = \mu\eta^{-1} - \zeta \tag{B.14}$$

We can solve this system efficiently by analytically eliminating all variables except $\mathrm{svec}(\Delta S)$. We can compute $\mathrm{svec}(\Delta S)$ by solving the following linear matrix equation using an off-the-shelf linear system solver

$$\begin{aligned} \left(T \otimes_s S^{-1} + \frac{\zeta\eta^{-1}}{\kappa_1\zeta\eta^{-1} + 1} v_I v_I^\top\right) \mathrm{svec}(\Delta S) &= v_I g_2 - v_I\mu\eta^{-1} - g_1 + \mu\,\mathrm{svec}(S^{-1}) \\ &\quad + v_I\zeta\eta^{-1}(-\kappa_1\zeta\eta^{-1} - 1)^{-1}\left(-\kappa_1(\mu\eta^{-1} - g_2 - \omega) + \mu\omega^{-1} + v_I^\top \mathrm{svec}(S) + \eta - 1\right) \end{aligned} \tag{B.15}$$

where $\kappa_1 := \omega^{-1}(1 - v_I^\top \mathrm{svec}(S) - \eta)$. Then, computing the rest of the update directions $\Delta\eta$, $\mathrm{svec}(\Delta T)$, $\Delta\zeta$, and $\Delta\omega$ amounts to back-substituting the solution to (B.15) for $\mathrm{svec}(\Delta S)$ through the analytical variable elimination equations.

Given how to compute the update directions, the primal-dual interior point method proceeds as follows. Initialize the variables $\theta = (S, \eta, T, \zeta, \omega)$ to an arbitrary strictly feasible point (i.e. $S \succ 0$, $\eta > 0$, $T \succ 0$, $\zeta > 0$, and $\omega > 0$). Starting from this primal-dual pair we compute an estimate of the barrier parameter as follows

$$\mu \leftarrow \frac{\langle S, T\rangle + \eta\zeta + \omega(1 - v_I^\top \mathrm{svec}(S) - \eta)}{2(k+2)},$$

where, as done in (Helmberg, 1994), we use (B.8) and divide by two. Then, we compute the update direction $\Delta\theta$ as described above and perform a backtracking line search to find a step size $\delta \in (0, 1]$ such that $\theta + \delta\,\Delta\theta$ is again strictly feasible. Lastly, following (Helmberg & Rendl, 2000), we compute a non-increasing estimate of the barrier parameter

$$\mu \leftarrow \min\left\{\mu_{\text{prev}}, \gamma\frac{\langle S, T\rangle + \eta\zeta + \omega(1 - v_I^\top \mathrm{svec}(S) - \eta)}{2(k+2)}\right\} \quad \text{where} \quad \gamma = \begin{cases} 1 & \text{if } \delta \le \frac{1}{5} \\ \frac{5}{10} - \frac{4}{10}\delta^2 & \text{if } \delta > \frac{1}{5} \end{cases}.$$

We iterate over these steps until the barrier parameter μ is small enough (e.g. $\mu < 10^{-7}$).

B.2. Solving (16)

The subproblem (16) can be rewritten as the following (small) quadratic semidefinite program

$$\begin{aligned} \max\ & \langle C - \mathcal{A}^* y_t, \eta\bar{X}_t + V_t S V_t^\top\rangle + \langle b - \tilde{\nu}, y_t\rangle - \frac{1}{2\rho}\left\|b - \tilde{\nu} - \mathcal{A}(\eta\bar{X}_t + V_t S V_t^\top)\right\|_2^2 \\ \text{s.t.}\ & \eta \ge 0 \\ & S \succeq 0 \\ & \eta + \mathrm{tr}(S) \le \alpha \end{aligned}$$

For this subproblem, unlike the subproblem solved in subsection B.1, we are solving for η and S to compute the candidate iterate $\tilde{y}_{t+1}$, update the primal variable, and update the model. This (small) quadratic semidefinite program is equivalent to the following optimization problem

$$\begin{aligned} \min \quad & \frac{1}{2}\operatorname{svec}(S)^\top Q_{11}\operatorname{svec}(S) + \eta\, q_{12}^\top \operatorname{svec}(S) + \frac{1}{2}\eta^2 q_{22} + h_1^\top \operatorname{svec}(S) + \eta\, h_2 \\ \text{s.t.} \quad & \eta \geq 0 \\ & S \succeq 0 \\ & 1 - v_I^\top \operatorname{svec}(S) - \eta \geq 0 \end{aligned} \tag{B.16}$$

where

$$\begin{aligned} Q_{11} &= \frac{\alpha^2}{\rho}\sum_{i=1}^{m} \operatorname{svec}(V_t^\top A_i V_t)\operatorname{svec}(V_t^\top A_i V_t)^\top \\ q_{12} &= \frac{\alpha^2}{\rho \operatorname{tr}(\bar{X}_t)}\operatorname{svec}\left(V_t^\top \mathcal{A}^*\mathcal{A}\,\bar{X}_t V_t\right) \\ q_{22} &= \frac{\alpha^2}{\rho \operatorname{tr}(\bar{X}_t)^2}\left\langle \mathcal{A}\bar{X}_t, \mathcal{A}\bar{X}_t\right\rangle \\ h_1 &= \alpha\ \operatorname{svec}\left(V_t^\top\left(\mathcal{A}^* y_t - C - \frac{1}{\rho}\mathcal{A}^*(b-\tilde{\nu})\right)V_t\right) \\ h_2 &= \frac{\alpha}{\operatorname{tr}(\bar{X}_t)}\left\langle \bar{X},\, \mathcal{A}^* y_t - C - \frac{1}{\rho}\mathcal{A}^*(b-\tilde{\nu})\right\rangle \\ v_I &= \operatorname{svec}(I) \end{aligned}$$

We follow the same derivation procedure as in subsection B.1, so we proceed by including the details which differ from the previous section. The Lagrangian of the dual barrier problem of (B.16) is as follows

$$\begin{aligned} L_\mu(S,\eta,T,\zeta,\omega) = {} & \frac{1}{2}\operatorname{svec}(S)^\top Q_{11}\operatorname{svec}(S) + \eta\, q_{12}^\top \operatorname{svec}(S) \\ & + \frac{1}{2}\eta^2 q_{22} + h_1^\top \operatorname{svec}(S) + \eta\, h_2 - \operatorname{svec}(S)^\top \operatorname{svec}(T) - \eta\zeta \\ & - \omega(1 - v_I^\top \operatorname{svec}(S) - \eta) + \mu(\log\det(T) + \log\zeta + \log\omega). \end{aligned} \tag{B.17}$$

The first-order optimality conditions of (B.17) yields the following system of equations (after the same standard linearization)

$$\mathrm{F}_\mu(S,\eta,T,\zeta,\omega) := \begin{pmatrix} Q_{11}\operatorname{svec}(S) + \eta\, q_{12} + h_1 - \operatorname{svec}(T) + \omega v_I \\ q_{12}^\top \operatorname{svec}(S) + \eta\, q_{22} + h_2 - \zeta + \omega \\ ST - \mu I \\ \eta\zeta - \mu \\ \omega(1 - v_I^\top \operatorname{svec}(S) - \eta) - \mu \end{pmatrix} =: \begin{pmatrix} \mathsf{F}_1 \\ \mathsf{F}_2 \\ \mathsf{F}_3 \\ \mathsf{F}_4 \\ \mathsf{F}_5 \end{pmatrix} = 0. \tag{B.18}$$

The Newton's method step direction $(\Delta S, \Delta\eta, \Delta T, \Delta\zeta, \Delta\omega)$ is determined via the following linearized system

$$Q_{11}\operatorname{svec}(\Delta S) + \Delta\eta\, q_{12} - \operatorname{svec}(\Delta T) + \Delta\omega\, v_I = -\mathsf{F}_1 \tag{B.19}$$

$$q_{12}^\top \operatorname{svec}(\Delta S) + \Delta\,\eta q_{22} - \Delta\zeta + \Delta\omega = -\mathsf{F}_2 \tag{B.20}$$

$$\omega^{-1}(1 - v_I^\top \operatorname{svec}(S) - \eta)\Delta\omega - v_I^\top \operatorname{svec}(\Delta S) - \Delta\eta = \mu\omega^{-1} + v_I^\top \operatorname{svec}(S) + \eta - 1 \tag{B.21}$$

$$(T \otimes_s S^{-1})\operatorname{svec}(\Delta S) + \operatorname{svec}(\Delta T) = \mu\operatorname{svec}(S^{-1}) - \operatorname{svec}(T) \tag{B.22}$$

$$\zeta\eta^{-1}\Delta\eta + \Delta\zeta = \mu\eta^{-1} - \zeta \tag{B.23}$$

We can solve this system efficiently by analytically eliminating all variables except $\operatorname{svec}(\Delta S)$. We can compute $\operatorname{svec}(\Delta S)$

by solving the following linear matrix equation using an off-the-shelf linear system solver

$$\begin{aligned}\big(Q_{11} + T \otimes_s S^{-1} - (\kappa_1\kappa_2 + 1)^{-1}\big(q_{12}(\kappa_1 q_{12} + v_I)^\top + v_I(q_{12} - \kappa_2 v_I)^\top\big)\big)\,\mathrm{svec}(\Delta S)\\ = q_{12}\left((\kappa_1\kappa_2+1)^{-1}\left(\mu\omega^{-1} + v_I^\top \mathrm{svec}(S) + \eta - 1 + \kappa_1(\mathsf{F}_2 - \mu\eta^{-1} + \zeta)\right)\right)\\ + v_I\left((\kappa_1\kappa_2+1)^{-1}\left(\mathsf{F}_2 - \mu\eta^{-1} + \zeta - \kappa_2(\mu\omega^{-1} + v_I^\top\mathrm{svec}(S) + \eta - 1)\right)\right)\\ - \mathsf{F}_1 + \mu\,\mathrm{svec}(S^{-1}) - \mathrm{svec}(T),\end{aligned} \tag{B.24}$$

where $\kappa_1 := \omega^{-1}(1 - v_I^\top \mathrm{svec}(S) - \eta)$ and $\kappa_2 := \zeta\eta^{-1} + q_{22}$. Then, computing the rest of the update directions $\Delta\eta$, $\mathrm{svec}(\Delta T)$, $\Delta\zeta$, and $\Delta\omega$ amounts to back-substituting the solution to (B.24) for $\mathrm{svec}(\Delta S)$ through the analytical variable elimination equations.

We make a single change to the initialization as compared with the interior point method presented in subsection B.1. Since this interior point method can be executed multiple times in a row with only the value of $\tilde{\nu}$ changing as the first step in the alternating maximization algorithm, we warm-start initialize θ with the previous execution's $\theta_\star$. We observe non-negligible convergence improvements over arbitrary initialization as this interior point method procedure is called in sequence.

B.3. Remark on Solving Subproblems

Ding & Grimmer (2023) claim that (16) could be solved using (accelerated) projected gradient descent and describe the method necessary for projection and proper scaling, but they use Mosek (ApS, 2019), an off-the-shelf solver, in their experiments. We find that while theoretically possible, projected gradient descent does not work well in practice since choosing the correct step size to obtain a high quality solution to (16) is difficult and varies between time steps t. We instead use (and advocate for) the primal-dual interior point method derived in Appendix B.2.

C. Additional Details on Scaling with Matrix Sketching

The values $\langle C, \bar{X}_t\rangle$, $\mathrm{tr}(\bar{X}_t)$, and $\mathcal{A}\bar{X}_t$ can be efficiently maintained given low-rank updates to $\bar{X}_t$ due to the linearity of the operations,

$$\langle C, \bar{X}_{t+1}\rangle \leftarrow \langle C, \eta_{t+1}\bar{X}_t + V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\rangle = \eta_{t+1}\langle C, \bar{X}_t\rangle + \mathrm{tr}\Big(V_t^\top Q_{\underline{c}}^\top \big(C V_t Q_{\underline{c}} \Lambda_{\underline{c}}\big)\Big), \tag{C.1}$$

$$\mathrm{tr}(\bar{X}_{t+1}) \leftarrow \mathrm{tr}\Big(\eta_{t+1}\bar{X}_t + V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\Big) = \eta_{t+1}\mathrm{tr}\big(\bar{X}_t\big) + \mathrm{tr}\big(\Lambda_{\underline{c}}\big)\,, \tag{C.2}$$

$$\mathcal{A}\bar{X}_{t+1} \leftarrow \mathcal{A}\Big(\eta_{t+1}\bar{X}_t + V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\Big) = \eta_{t+1}\mathcal{A}\bar{X}_t + \mathcal{A}\big(V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\big), \tag{C.3}$$

where $\mathcal{A}\big(V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\big)$ can be computed efficiently since

$$\Big(\mathcal{A}\big(V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\big)\Big)_i = \mathrm{tr}\Big(V_t^\top Q_{\underline{c}}^\top \big(A_i V_t Q_{\underline{c}} \Lambda_{\underline{c}}\big)\Big). \tag{C.4}$$

C.1. Nyström Sketch Reconstruction

Given the sketch P_t and the projection matrix Ψ, we can compute a rank-r approximation to $\bar{X}_t$. The approximation is as follows

$$\widehat{\bar{X}_t} := P_t(\Psi^\top P_t)^+ P_t^\top = (\bar{X}_t\Psi)(\Psi^\top \bar{X}_t \Psi)^+ (\bar{X}_t\Psi)^\top, \tag{C.5}$$

where $^+$ is the Moore-Penrose inverse. The reconstruction is called a *Nyström approximation*. We will almost always compute the best rank-r approximation of $\widehat{\bar{X}_t}$ for memory efficiency. In practice, we use the numerically stable implementation of the Nyström approximation provided by Yurtsever et al. (2021). The Nyström approximation yields a provably good approximation for any sketched matrix. See the following fact from (Tropp et al., 2017),

Fact C.1. *Fix $X \in \mathbb{S}_+^n$ arbitrarily. Let $P := X\Psi$ where $\Psi \in \mathbb{R}^{n\times r}$ has independently sampled standard normal entries. For each $r' < r - 1$, the Nyström approximation $\widehat{X}$ as defined in (C.5) satisfies*

$$\mathbb{E}_\Psi\big\|X - \widehat{X}\big\|_* \le \left(1 + \frac{r'}{r - r' - 1}\right)\big\|X - [\![X]\!]_r\big\|_*, \tag{C.6}$$

where $\mathbb{E}_\Psi$ is the expectation with respect to Ψ and $[\![X]\!]_r$ is returns an r-truncated singular-value decomposition of the matrix X, which is a best rank-r approximation with respect to every unitarily invariant norm (Mirsky, 1960)*. If we replace $\widehat{X}$ with $[\![\widehat{X}]\!]_r$, this error bound still remains valid.*

C.2. USBS Memory Requirements.

When we implement USBS using this matrix sketching procedure we see significant decrease in time and memory required by USBS. Storing the projection matrix Ψ and P_t requires $\mathcal{O}(nr)$ memory. Storing $\langle C, \bar{X}_t\rangle$ and $\operatorname{tr}(\bar{X}_t)$ requires $\mathcal{O}(1)$ memory, respectively, and storing $\mathcal{A}\bar{X}_t$, ν_t, y_t, and $\tilde{y}_{t+1}$ requires $\mathcal{O}(m)$ memory. The primal-dual interior point methods require $\mathcal{O}(k^2)$ and $\mathcal{O}(k^4)$ memory, respectively, and storing V_t requires $\mathcal{O}(nk)$ memory. This means that the entire memory required by USBS is $\mathcal{O}(nr + nk + m + k^4)$ working memory (not including the memory to store the problem data) which for many SDPs is much less than explicitly storing the iterate $\bar{X}_t$ which requires $\mathcal{O}(n^2)$ memory.

D. Proof of Theorem 3.1

In this section, we present the non-asymptotic convergence results for USBS. First, we summarize the relevant results presented by Díaz & Grimmer (2023) in the following theorem.

Theorem D.1. *Let $\beta > 0$ and $\rho > 0$ be constant, $f : \mathbb{R}^m \to (-\infty, +\infty]$ be a proper closed convex function with nonempty set of minimizers $\mathcal{Y}_\star$, and the sequence of models produced by the proximal bundle method $\left\{\hat{f}_{t+1} : \mathbb{R}^m \to (-\infty, +\infty]\right\}$ satisfy the conditions (3), (4), and (5). If the iterates y_t and subgradients g_{t+1} are bounded, then the iterates y_t have $f(y_t) - f(y_\star) \leq \varepsilon$ for all $t \geq \mathcal{O}(1/\varepsilon^3)$. Additionally, if $f(y) - f(y_\star) \geq \mathcal{O}(\operatorname{dist}^2(y, \mathcal{Y}_\star))$ for all y, then the bound improves to $f(y_t) - f(y_\star) \leq \varepsilon$ for all $t \geq \mathcal{O}(1/\varepsilon)$.*

We will use this result to prove the non-asymptotic convergence rates for the spectral bundle method.

Note the following fact necessary to use Theorem D.1.

Fact D.2. *The set $\{y \in \mathcal{Y} : f(y_\star) \leq f(y) \leq f(y_0)\}$ is compact and contains all iterates y_t. Moreover, the iterates y_t and subgradients g_{t+1} are bounded, i.e. $\sup_{t\geq 0}\{\operatorname{dist}(y_t, \mathcal{Y}_*)\} < \infty$ and $\sup_{t\geq 0}\{\|g_{t+1}\| : g_{t+1} \in \partial f(\tilde{y}_{t+1})\} < \infty$.*

Proof. The function f is continuous and proper over the domain $\mathcal{Y}$. It is a well-known fact that the level sets of continuous proper functions are compact. Hence, the union of level sets $\{y \in \mathcal{Y} : f(y_\star) \leq f(y) \leq f(y_0)\}$ is compact, and therefore, $\sup_{t\geq 0}\{\operatorname{dist}(y_t, \mathcal{Y}_*)\} < \infty$. The subgradients take the form $g_{t+1} = b + \alpha\mathcal{A}(vv^\top) \in \partial f(\tilde{y}_{t+1})$ where $v \in \mathbb{R}^n$ is a unit-normed vector, and thus, $\|g_{t+1}\| < \infty$ for all t. □

The following lemma guarantees quadratic growth whenever strict complementarity holds.

Lemma D.3. *Let $y \in \mathcal{Y}$ be arbitrary. Then,*

$$\operatorname{dist}^{2^d}(y, \mathcal{Y}_\star) \leq \mathcal{O}(f(y) - f(y_\star)), \tag{D.1}$$

where $d \in \{0, 1, \ldots, m\}$ is the singularity degree (Sturm, 2000; Drusvyatskiy & Wolkowicz, 2017; Sremac et al., 2021)*. If strict complementarity holds, then $d \leq 1$, and if Slater's condition holds, then $d = 0$.*

Proof. Let $\mathcal{F} := \{y \in \mathbb{R}^m : \mathcal{A}^* y - C \in \mathbb{S}^n_+\}$. Observe that $\mathcal{Y}_\star = \mathcal{Y} \cap \mathcal{F}$. Since $\mathcal{Y}_\star$ is compact, the Hölderian error bound (Sturm, 2000; Drusvyatskiy & Wolkowicz, 2017; Sremac et al., 2021) ensures that

$$\operatorname{dist}(y, \mathcal{Y}_\star) \leq \mathcal{O}\left(\operatorname{dist}^{2^{-d}}(y, \mathcal{Y}) + \operatorname{dist}^{2^{-d}}(y, \mathcal{F})\right),$$

where $d \in \{0, 1, \ldots, m\}$ is the singularity degree of the SDP. This implies that for $y \in Y$,

$$\begin{aligned}\operatorname{dist}^{2^d}(y, \mathcal{Y}_\star) &\leq \mathcal{O}\left(\operatorname{dist}(y, \mathcal{F})\right) \\ &\leq \mathcal{O}\left(\alpha \max\{\lambda_{\max}(C - \mathcal{A}^* y), 0\}\right) \\ &\leq \mathcal{O}\left(f(y) - f(y_\star)\right).\end{aligned}$$

□

The following three lemmas guarantee primal feasiblity, dual feasibility, and primal-dual objective optimality given convergence of the penalized dual objective, i.e. $f(y_t) - f(y_\star) \leq \varepsilon$.

Lemma D.4. *At every descent step t,*

$$\operatorname{dist}(\mathcal{A}X_{t+1}, \mathcal{K}) \leq \mathcal{O}\Big(\sqrt{f(y_t) - f(y_\star)}\Big). \tag{D.2}$$

Additionally, if Slater's condition holds and $y_\star$ is unique, then

$$\operatorname{dist}(\mathcal{A}X_{t+1}, \mathcal{K}) \leq \mathcal{O}(f(y_t) - f(y_\star)). \tag{D.3}$$

Proof. For any descent step t, it is easy to verify from (14) and (17) that

$$\operatorname{proj}_{\mathcal{K}}(\mathcal{A}X_{t+1}) - \mathcal{A}X_{t+1} \leq \rho(y_t - y_{t+1}),$$

and therefore,

$$\|\mathcal{A}X_{t+1} - \operatorname{proj}_{\mathcal{K}}(\mathcal{A}X_{t+1})\|^2 \leq \rho^2 \|y_t - y_{t+1}\|^2.$$

To complete the proof we utilize the fact that $\hat{f}_t(y_{t+1}) = \min_{y\in\mathcal{Y}} \hat{f}_t(y) + \frac{\rho}{2}\|y - y_t\|^2$, the fact that $\hat{f}_t(y) \leq f(y)$ for all $y \in \mathcal{Y}$, and the definition of a descent step, which gives

$$\frac{\rho}{2}\|y_{t+1} - y_t\|^2 \leq \hat{f}_t(y_t) - \hat{f}_t(y_{t+1}) \leq f(y_t) - \hat{f}_t(y_{t+1}) \leq \frac{f(y_t) - f(y_{t+1})}{\beta} \leq \frac{f(y_t) - f(y_\star)}{\beta}.$$

Assume Slater's condition holds and $y_\star$ is unique. Then, using Lemma D.3 gives

$$\begin{aligned}
\|\mathcal{A}X_{t+1} - \operatorname{proj}_{\mathcal{K}}(\mathcal{A}X_{t+1})\|^2 &\leq \rho^2 \|y_t - y_{t+1}\|^2 \\
&\leq \rho^2 \left(\|y_t - y_\star\|^2 + \|y_{t+1} - y_\star\|^2\right) \\
&\leq \mathcal{O}\big((f(y_t) - f(y_\star))^2\big) + \mathcal{O}\big((f(y_{t+1}) - f(y_\star))^2\big) \\
&\leq \mathcal{O}\big((f(y_t) - f(y_\star))^2\big).
\end{aligned}$$

□

Lemma D.5. *Suppose strong duality holds. Then, at every descent step t,*

$$\lambda_{\max}(C - \mathcal{A}^* y_t) \leq \mathcal{O}(f(y_t) - f(y_\star)). \tag{D.4}$$

Proof. By definition of strong duality, we know that for any $X_\star \in \mathcal{X}_\star$ that $\langle C, X_\star\rangle = \langle b, y_\star\rangle$, and equivalently, $\langle X_\star, \mathcal{A}^* y_\star - C\rangle = 0$. Then, the dual objective gap is bounded as follows

$$\begin{aligned}
\langle b, y_t - y_\star\rangle &= \langle \mathcal{A}X_\star, y_t - y_\star\rangle \\
&= \langle X_\star, \mathcal{A}^*(y_t - y_\star)\rangle \\
&= \langle X_\star, (\mathcal{A}^* y_t - C) - (\mathcal{A}^* y_\star - C)\rangle \\
&= \langle X_\star, \mathcal{A}^* y_t - C\rangle \\
&\geq -\|X_\star\|_* \max\{\lambda_{\max}(C - \mathcal{A}^* y_t), 0\}.
\end{aligned}$$

We now utilize this bound to obtain the desired result

$$f(y_t) - f(y_\star) = \langle b, y_t - y_\star\rangle + \alpha \max\{\lambda_{\max}(C - \mathcal{A}^* y_t), 0\} \geq \|X_\star\|_* \max\{\lambda_{\max}(C - \mathcal{A}^* y_t), 0\}.$$

□

Lemma D.6. *At every descent step t,*

$$|\langle b, y_{t+1}\rangle - \langle C, X_{t+1}\rangle| \leq \mathcal{O}(f(y_t) - f(y_\star)) + \mathcal{O}\Big(\sqrt{f(y_t) - f(y_\star)}\Big). \tag{D.5}$$

Additionally, if Slater's condition holds and $y_\star$ is unique, then

$$|\langle b, y_{t+1}\rangle - \langle C, X_{t+1}\rangle| \leq \mathcal{O}(f(y_t) - f(y_\star)). \tag{D.6}$$

Proof. We start by rewriting the primal-dual gap as follows

$$\begin{aligned}\langle C, X_{t+1}\rangle - \langle b, y_{t+1}\rangle &= \langle C, X_{t+1}\rangle - \langle \mathcal{A}X_{t+1}, y_{t+1}\rangle + \langle \mathcal{A}X_{t+1} - b, y_{t+1}\rangle \\ &= \langle X_{t+1}, C - \mathcal{A}^* y_{t+1}\rangle + \langle \mathcal{A}X_{t+1} - b, y_{t+1}\rangle.\end{aligned}$$

We will now bound the absolute values of the two resulting terms to bound the desired quantity. Using the Cauchy-Schwarz inequality and Lemma D.4 it can be seen

$$\begin{aligned}|\langle \mathcal{A}X_{t+1} - b, y_{t+1}\rangle| &\leq \|\mathcal{A}X_{t+1} - b\|\|y_{t+1}\| \\ &\leq \mathcal{O}\Big(\|y_{t+1}\|\sqrt{f(y_t) - f(y_\star)}\Big) \\ &\leq \mathcal{O}\Big(\sqrt{f(y_t) - f(y_\star)}\Big),\end{aligned}$$

where the last inequality comes from Fact D.2. Assume Slater's condition holds and $y_\star$ is unique. Then, using Lemma D.3 gives

$$|\langle \mathcal{A}X_{t+1} - b, y_{t+1}\rangle| \leq \mathcal{O}\left(f(y_t) - f(y_\star)\right).$$

Since $X_{t+1} \succeq 0$ and $\operatorname{tr}(X_{t+1}) \leq \alpha$ by construction, we can use Lemma D.5 to yield

$$\begin{aligned}|\langle X_{t+1}, C - \mathcal{A}^* y_{t+1}\rangle| &\leq \mathcal{O}(\max\{C - \mathcal{A}^* y_{t+1}, 0\}) \\ &\leq \mathcal{O}(f(y_{t+1}) - f(y_\star)) \\ &\leq \mathcal{O}(f(y_t) - f(y_\star)).\end{aligned}$$

The immediately yields the desired result. □

D.1. Proof of Theorem 3.1

The overall proof strategy is to use Theorem D.1 to show that the penalized dual gap, $f(y_t) - f(y_*)$, converges at a worst case rate of $\mathcal{O}(1/\varepsilon^3)$, which improves to $\mathcal{O}(1/\varepsilon)$ if Slater's condition or strict complementarity hold. Then, we can use Lemmas D.4, D.5, and D.6 to obtain the stated convergence of primal feasibility, dual feasibility, and primal-dual optimality.

To apply the result of Theorem D.1, we showed that the norms of the iterates and subgradients of f are bounded (Fact D.2), and so all we need to do is show that the model satisfies the conditions (3), (4), and (5).

Let v be a maximum eigenvector $C - \mathcal{A}^*\tilde{y}_{t+1}$ if $\lambda_{\max}(C - \mathcal{A}^*\tilde{y}_{t+1}) > 0$ and $v = 0$, otherwise. Then denote $g_{t+1} = b + \alpha\mathcal{A}(vv^\top) \in \partial f(\tilde{y}_{t+1})$ as the subgradient of f corresponding v at the candidate iterate $\tilde{y}_{t+1}$. Let $s_{t+1} = \rho(y_t - \tilde{y}_{t+1}) \in \partial\hat{f}_t(\tilde{y}_{t+1})$ be the aggregate subgradient. Recall that at iteration $t+1$ the model approximates $\mathcal{X}$ by the low-dimensional spectral set

$$\widehat{\mathcal{X}}_{t+1} := \left\{\eta\bar{X}_{t+1} + V_{t+1}SV_{t+1}^\top : \eta\operatorname{tr}(\bar{X}_{t+1}) + \operatorname{tr}(S) \leq \alpha, \eta \geq 0, S \in \mathbb{S}_+^k\right\}.$$

and therefore the penalized dual objective and model respectively take the following forms for any $y \in \mathcal{Y}$,

$$f(y) = \sup_{X\in\mathcal{X}} \langle C - \mathcal{A}^* y, X\rangle + \langle b, y\rangle \quad \text{and} \quad \hat{f}_{t+1}(y) = \sup_{X\in\widehat{\mathcal{X}}_{t+1}} \langle C - \mathcal{A}^* y, X\rangle + \langle b, y\rangle. \tag{D.7}$$

Verifying (3). The condition (3) follows immediately from (D.7), since $\widehat{\mathcal{X}}_{t+1} \subseteq \mathcal{X}$.

Verifying (4). Since V_{t+1} spans v, there exists a vector $s \in \mathbb{R}^k$ such that $V_{t+1}s = v$. Taking $\eta = 0$ and $S = \alpha ss^\top$ implies $\alpha vv^\top \in \widehat{\mathcal{X}}_{t+1}$. Hence,

$$\begin{aligned}\hat{f}_{t+1}(y) &\geq \langle C - \mathcal{A}^* y, \alpha vv^\top\rangle + \langle b, y\rangle \\ &= \langle C - \mathcal{A}^*, \alpha vv^\top\rangle + \langle b, \tilde{y}_{t+1}\rangle + \langle b + \alpha\mathcal{A}(vv^\top), y - \tilde{y}_{t+1}\rangle \\ &= f(\tilde{y}_{t+1}) + \langle g_{t+1}, y - \tilde{y}_{t+1}\rangle.\end{aligned}$$

Verifying (5). From the first-order optimality conditions and the update step (14) we know that

$$s_{t+1} = \rho(y_t - \tilde{y}_{t+1}) = b - \nu_{t+1} - \mathcal{A}X_{t+1},$$
$$\hat{f}_t(\tilde{y}_{t+1}) = \langle C - \mathcal{A}^*\tilde{y}_{t+1}, X_{t+1}\rangle + \langle b - \nu_{t+1}, \tilde{y}_{t+1}\rangle.$$

To show the desired result, we first want to show that $X_{t+1} \in \widehat{X}_{t+1}$. If $k_p = 0$, we are done since $X_{t+1} = \bar{X}_{t+1}$. Otherwise, $k_p \geq 1$. First note that $\operatorname{tr}(X_{t+1}) \leq \alpha$ by construction. Then,

$$\begin{aligned} X_{t+1} &= \eta_{t+1}\bar{X}_t + V_t S_{t+1} V_t^\top \\ &= \eta_{t+1}\bar{X}_t + V_t(Q_{\overline{p}}\Lambda_{\overline{p}}Q_{\overline{p}}^\top + Q_{\underline{c}}\Lambda_{\underline{c}}Q_{\underline{c}}^\top)V_t^\top \\ &= \eta_{t+1}\bar{X}_t + V_t Q_{\overline{p}}\Lambda_{\overline{p}}Q_{\overline{p}}^\top V_t^\top + V_t Q_{\underline{c}}\Lambda_{\underline{c}}Q_{\underline{c}}^\top V_t^\top \\ &= \bar{X}_{t+1} + V_t Q_{\overline{p}}\Lambda_{\overline{p}}Q_{\overline{p}}^\top V_t^\top. \end{aligned}$$

Since V_{t+1} spans each of the columns of $V_t Q_{\overline{p}}$, we can conclude that $X_{t+1} \in \widehat{X}_{t+1}$. Thus, for any $y \in \mathcal{Y}$,

$$\hat{f}_{t+1}(y) \geq \langle C - \mathcal{A}^* y, X_{t+1}\rangle + \langle b - \nu_{t+1}, y\rangle = \hat{f}_t(\tilde{y}_{t+1}) + \langle s_{t+1}, y - \tilde{y}_{t+1}\rangle.$$

E. Experimental Setup and Details

E.1. MaxCut

The MaxCut problem is a fundamental combinatorial optimization problem and its SDP relaxation is a common test bed for SDP solvers. Given an undirected graph, the MaxCut is a partitioning of the n vertices into two sets such that the number of edges between the two subsets is maximized. Formally, the MaxCut problem can be written as follows

$$\max \quad \frac{1}{4} x^\top L\, x \quad \text{s.t.} \quad x \in \{\pm 1\}^n, \tag{E.1}$$

where L is the graph Laplacian. This optimization problem is known to be NP-hard (Karp, 1972), but rounding the solution to the SDP relaxation can yield very strong approximate solutions (Goemans & Williamson, 1995).

E.1.1. SDP Relaxation

For any vector $x \in \{\pm 1\}^n$, the matrix $X := xx^\top$ is positive semidefinite and all of its diagonal entries are exactly one. These implicit constraints allow us to arrive at the MaxCut SDP as follows

$$\max \quad \frac{1}{4}\operatorname{tr}(LX) \quad \text{s.t.} \quad \operatorname{diag}(X) = \mathbf{1} \text{ and } X \succeq 0, \tag{E.2}$$

where $\operatorname{diag}(\cdot)$ extracts the diagonal of a matrix into a vector and $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. A matrix solution $X_\star$ to (22) does not readily elicit a graph cut. One way to round $X_\star$ is to compute a best rank-one approximation $x_\star x_\star^\top$ and use $\operatorname{sgn}(x_\star)$ as an approximate cut. There are more sophisticated rounding procedures (e.g. (Goemans & Williamson, 1995)), but this one works quite well in practice.

Notice that (22) contains exactly n equality constraints. Additionally, in many instances of MaxCut, the graph Laplacian L is sparse and the optimal solution is low-rank. This means that we can represent the MaxCut instance with far less than $\mathcal{O}(n^2)$ memory. These attributes make sketching the matrix variable X with relatively small r an attractive and effective approach when considering the MaxCut problem.

E.1.2. Rounding

Both CGAL and USBS maintain a sketch of the primal variable that, along with the test matrix, can be used to construct a low-rank approximation of the primal iterate $\widehat{X} = U\Lambda U^\top$ where $U \in \mathbb{R}^{n\times r}$ has orthonormal columns. Following (Yurtsever et al., 2021), we evaluate the size of r cuts given by the column vectors of $\operatorname{sgn}(U)$ and choose the largest.

E.1.3. EXPERIMENTAL SETUP

We evaluate the CGAL and USBS with and without warm-starting on ten instances from the DIMACS10 (Bader et al.) where n and the number of edges in each graph are shown in Table 1. We warm-start each method by dropping the last 1% of vertices from the graph resulting in a graph with 99% of the vertices. We pad the solution from the 99%-sized problem with zeros and rescale as necessary to create the warm-start initialization. Unless otherwise specified, we use the following hyperparameters for USBS in our experiments: $r = 10$, $\rho = 0.01$, $\beta = 0.25$, $k_c = 10$, $k_p = 1$. In our experiments, we find that $k_p > 0$ does improve performance slightly. See the implementation for more details.

E.2. Quadratic Assignment Problem

The quadratic assignment problem (QAP) is a very difficult but fundamental class of combinatorial optimization problems containing the traveling salesman problem, max-clique, bandwidth problems, and facilities location problems among others (Loiola et al., 2007). SDP relaxations have been shown to facilitate finding good solutions to large QAPs (Zhao et al., 1998). The QAP can be formally defined as follows

$$\min \quad \operatorname{tr}\left(W\Pi D\Pi^\top\right) \quad \text{s.t.} \quad \Pi \text{ is an } \mathtt{n} \times \mathtt{n} \text{ permutation matrix}, \tag{E.3}$$

where $W \in \mathbb{S}^{\mathtt{n}}$ is the weight matrix, $D \in \mathbb{S}^{\mathtt{n}}$ is the distance matrix, and the goal is to optimize for an assignment Π which aligns W and D. The number of $\mathtt{n} \times \mathtt{n}$ permutation matrices is $\mathtt{n}!$, so a brute-force search becomes quickly intractable as $\mathtt{n}$ grows. Generally, QAP instances with $\mathtt{n} > 30$ are intractable to solve exactly. In our experiments, using an SDP relaxation and rounding procedure, we can obtain good solutions to QAPs where $\mathtt{n}$ is between 136 and 198.

E.2.1. SDP RELAXATION

There are many SDP relaxtions for QAPs, but we consider the one presented by (Yurtsever et al., 2021) and inspired by (Huang et al., 2014; Bravo Ferreira et al., 2018) which is formulated as follows

$$\begin{aligned} \min \quad & \operatorname{tr}\big((D \otimes W)\,\mathsf{Y}\big) \\ \text{s.t.} \quad & \operatorname{tr}_1(\mathsf{Y}) = I,\ \operatorname{tr}_2(\mathsf{Y}) = I,\ \mathcal{G}(\mathsf{Y}) \geq 0, \\ & \operatorname{vec}(\mathsf{B}) = \operatorname{diag}(\mathsf{Y}),\ \mathsf{B}\mathbf{1} = \mathbf{1},\ \mathbf{1}^\top \mathsf{B} = \mathbf{1}^\top,\ \mathsf{B} \geq 0, \\ & X := \begin{bmatrix} 1 & \operatorname{vec}(\mathsf{B})^\top \\ \operatorname{vec}(\mathsf{B}) & \mathsf{Y} \end{bmatrix} \succeq 0,\ \operatorname{tr}(\mathsf{Y}) = \mathtt{n} \end{aligned} \tag{E.4}$$

where $\otimes$ denotes the Kronecker product, $\operatorname{tr}_1(\cdot)$ and $\operatorname{tr}_2(\cdot)$ denote the partial trace over the first and second systems of Kronecker product respectively, $\mathcal{G}(\mathsf{Y})$ extracts the entries of Y corresponding to the nonzero entries of $D \otimes W$, and $\operatorname{vec}(\cdot)$ stacks the columns of a matrix one on top of the other to form a vector. In many cases, one of D or W is sparse (i.e. $\mathcal{O}(n)$ nonzero entries) resulting in $\mathcal{O}(\mathtt{n}^3)$ total constraints for the SDP.

The primal variable X has dimension $(\mathtt{n}^2+1) \times (\mathtt{n}^2+1)$ and as a result the SDP relaxation has $\mathcal{O}(\mathtt{n}^4)$ decision variables. Given the aggressive growth in complexity, most SDP based algorithms have difficulty operating on instances where $\mathtt{n} > 50$. To reduce the number of decision variables, we sketch X with $r = \mathtt{n}$, resulting in $\mathcal{O}(\mathtt{n}^3)$ entries in the sketch and the same number of constraints in most natural instances. We show that this enables CGAL and USBS to scale to instances where $\mathtt{n} = 198$ (more than 1.5 billion decision variables and constraints).

E.2.2. ROUNDING

We adopt the rounding method presented in (Yurtsever et al., 2021) to convert an approximate solution of (23) into a permutation matrix. Reconstructing an approximation to the primal variable from the sketch yields $\widehat{X} = U\Lambda U^\top$ where U has shape $(\mathtt{n}^2+1) \times \mathtt{n}$. For each column of U, we discard the first entry of the column vector, reshape the remaining $\mathtt{n}^2$ entries into an $\mathtt{n} \times \mathtt{n}$ matrix, and project the reshaped matrix onto the set of $\mathtt{n} \times \mathtt{n}$ permutation matrices using the Hungarian method (also known as Munkres' assignment algorithm) (Kuhn, 1955; Munkres, 1957; Jonker & Volgenant, 1988). This procedure yields a feasible point to the original QAP (E.3). To get an upper bound for (E.3), we perform the rounding procedure on each column of U and choose the permutation matrix Π which minimizes the objective $\operatorname{tr}\left(W\Pi D\Pi^\top\right)$.

E.2.3. EXPERIMENTAL SETUP

We evaluate CGAL and USBS on select large instances from `QAPLIB` (Burkard et al., 1997) and `TSPLIB` (Reinelt, 1995) ranging in size from 136 to 198. Provided with each of these QAP instances is the known optimum. For many instances, we find that the quality of the permutation matrix produced by the rounding procedure does not entirely correlate with the quality of the iterates with respect to the SDP (23). Thus, we apply the rounding procedure at every iteration of both CGAL and USBS. Our metric for evaluation is `relative gap` which is computed as follows

$$\texttt{relative gap} = \frac{\texttt{upper bound obtained} - \texttt{optimum}}{\texttt{optimum}}.$$

We report the lowest value so far of `relative gap` as `best relative gap`. CGAL (Yurtsever et al., 2021) is shown to obtain significantly smaller `best relative gap` than CSDP (Bravo Ferreira et al., 2018) and PATH (Zaslavskiy et al., 2008) on most instances, and thus, is a strong baseline.

To create a warm-start initialization, we create a slightly smaller QAP by dropping the final row and column of both D and W (i.e. solve a size $\texttt{n} - 1$ subproblem of the original instance). We use the solution to slightly smaller problem to set a warm-start initialization for the original problem, rescaling and padding with zeros where necessary. We stopped optimizing after one hour. When warm-starting, we optimize the slightly smaller problem for one hour and then optimize the original problem for one hour.

Following (Yurtsever et al., 2021), we apply the following scaling of the problem variables in (23)

$$\|C\|_{\mathrm{F}} = \operatorname{tr}(X_\star) = \|\mathcal{A}\|_{\mathrm{op}} = 1 \quad \text{and} \quad \|A_1\|_{\mathrm{F}} = \cdots = \|A_m\|_{\mathrm{F}}. \tag{E.5}$$

We use the following hyperparameters for USBS in our experiments: $\rho = 0.005$, $\beta = 0.25$, $k_c = 2$, $k_p = 0$. In our experiments, we find that $k_p > 0$ does not improve performance. See the implementation for more details.

E.3. Interactive Entity Resolution with ∃-constraints

Angell et al. (2022) introduced a novel SDP relaxation of a combinatorial optimization problem that arises in an automatic knowledge base construction problem (Uhlen et al., 2010; Krishnamurthy, 2015; Iv et al., 2022). They consider the problem of interactive entity resolution with user feedback to correct predictions made by a machine learning algorithm. Entity resolution (also known as coreference resolution or record linkage) is the process of identifying which mentions (sometimes referred to as records) of entities refer to the same entity (Binette & Steorts, 2022; Angell et al., 2020; Agarwal et al., 2021; 2022; Yadav et al., 2021). Entity resolution is an important problem central to automated knowledge base construction where the correctness of the resulting knowledge base is vital to its usefulness. The entity resolution problem is fundamentally a clustering problem where the prefect clusters contain all mentions of exactly one entity. Due to the large volumes of raw data frequently involved, machine learning approaches are often used to perform entity resolution. To correct inevitable errors in the predictions made by these automated methods, human feedback can be used to correct errors in entity resolution.

Utilizing human feedback to guide and correct clustering decisions can be broadly categorized into two groups (Bae et al., 2020): active clustering (also know as semi-supervised clustering) (Viswanathan et al., 2023; Vikram & Dasgupta, 2016; Mazumdar & Saha, 2017; Sato & Iwayama, 2009; Xiong et al., 2016) and interactive clustering (Chuang & Hsu, 2014; Basu et al., 2010; Coden et al., 2017; Dubey et al., 2010)[2]. There are several paradigms of human feedback used to steer a clustering algorithm (Frome et al., 2007; Wagstaff et al., 2001; Shental et al., 2004; Klein et al., 2002; Kulis et al., 2009; Lu & Carreira-Perpinan, 2008; Li et al., 2008; 2009; Xing et al., 2002), the most common of which is pairwise constraints (Wagstaff & Cardie, 2000) (i.e. statements about whether two points or mentions must or cannot be clustered together). Recently, Angell et al. (2022) introduced ∃-*constraints*, a new paradigm of interactive feedback for correcting entity resolution predictions and presented a novel SDP relaxation as part of a heuristic algorithm for satisfying ∃-constraints in the predicted clustering. This novel form of feedback allows users to specify constraints stating the existence of an entity with and without certain features.

[2]Commonly, *active* clustering is used to describe techniques where the algorithm queries the human for feedback about a specific pair or group of points or clusters (akin to active learning) and *interactive* clustering is used to describe techniques where humans observe the clustering output and provide unelicited feedback, typically in the form of some type of constraint, to the algorithm.

E.3.1. ∃-CONSTRAINTS

To define ∃-constraints precisely, we must introduce some terminology and notation. Let the set of mentions be represented by the vertex set V of a fully connected graph. We assume that each of the mentions $v \in V$ has an associated set of discrete features (attributes, relations, other features, etc.) that is readily available or can be extracted easily using some automated method. Let $\Phi = \{\phi_1, \phi_2, \ldots\}$ be the set of possible features and $\Phi(v) \subseteq \Phi$ be the subset of features associated with vertex $v \in V$. A clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of vertices V is a set of nonempty sets such that $C_i \subseteq V$, $C_i \cap C_j = \emptyset$ for all $i \neq j$, and $\bigcup_{C \in \mathcal{C}} C = V$. We denote the ground truth clustering of the vertices (partitioning mentions into entities) as $\mathcal{C}^\star$. Furthermore, for any subset of vertices $S \subseteq V$, we define the set of features associated with S as $\Phi(S) = \bigcup_{v \in S} \Phi(v)$. For any cluster C_i, we use $\Phi(C_i)$ as the canonicalization function for a predicted entity, but this framework allows for more complex (and even learned) canonicalization functions. When providing feedback, users are able to view the features of predicted clusters and then provide one or more ∃-constraints.

Formally, a ∃-constraint $\xi \subseteq \{+, -\} \times \Phi$ uniquely characterizes a constraint asserting there exists a cluster $C \in \mathcal{C}^\star$ such that all of the positive features $\xi^+ := \{\phi \in \Phi : (+, \phi) \in \xi\}$ are contained in the features of C (i.e. $\xi^+ \subseteq \Phi(C)$) and none of the negative features $\xi^- := \{\phi \in \Phi : (-, \phi) \in \xi\}$ are contained in the features of C (i.e. $\xi^- \cap \Phi(C) = \emptyset$). At any given time, let the set of ∃-constraints be represented by Ξ. We say that a subset of nodes $S \subseteq V$ *satisfies* a ∃-constraint ξ if $\xi^+ \subseteq \Phi(S)$ and $\xi^- \cap \Phi(S) = \emptyset$. Note that more than one ground truth cluster can satisfy a ∃-constraint. We say that a subset of vertices $S \subseteq V$ is *incompatible* with a ∃-constraint ξ if $\Phi(S) \cap \xi^- \neq \emptyset$ and denote this as $S \perp \xi$. Similarly, two ∃-constraints ξ_a, ξ_b are incompatible if $\xi_a^+ \cap \xi_b^- \neq \emptyset$ or $\xi_a^- \cap \xi_b^+ \neq \emptyset$, and is also denoted $\xi_a \perp \xi_b$. Furthermore, we say a subset of vertices $S \subseteq V$ and a ∃-constraint ξ or two ∃-constraints ξ_a, ξ_b are *compatible* if they are not incompatible and denote this as $S \not\perp \xi$ and $\xi_a \not\perp \xi_b$, respectively. Moreover, we use *compatible* to describe when a subset of vertices $S \subseteq V$ could be part of a cluster that *satisfies* a certain ∃-constraint ξ. This definition allows for $\Phi(S) \cap \xi^+$ to be empty, but still enables S to be compatible with ξ just as long as $\Phi(S) \cap \xi^-$ is empty. Observe that if every mention has a unique feature, the ∃-constraint framework fully encompasses must-link and cannot-link constraints.

E.3.2. SDP RELAXATION

Following (Angell et al., 2022), the general approach to clustering with ∃-constraints is to jointly cluster the vertices and the ∃-constraints, so we include ∃-constraints as additional vertices of the graph V (i.e. $\Xi \subset V$) and create corresponding decision variables for each of the ∃-constraints. Additionally, we create constraints to ensure that all of the ∃-constraints are satisfied and none of the ∃-constraints are incompatible with any of the other members of the predicted clustering. Given a ∃-constraint ξ and positive feature $\phi \in \xi^+$, let $\Gamma(\xi, \phi) := \{v \in V \setminus \Xi : \phi \in \Phi(v) \wedge \Phi(v) \cap \xi^- = \emptyset\}$ be the set of candidate vertices that could satisfy $\phi \in \xi^+$. The resulting integer linear programming problem is as follows

$$\max_{x_u, x_v \in \{e_1, e_2, \ldots, e_n\}} \sum_{(u,v) \in V \times V} w_{uv}\, x_u^\top x_v \quad \text{s.t.} \quad \begin{cases} \sum_{v \in \Gamma(\xi,\phi)} x_v^\top x_\xi \geq 1, & \text{for all } \phi \in \xi^+ \\ x_v^\top x_\xi = 0, & \text{for all } v \perp \xi \end{cases} \tag{E.6}$$

where $w_{uv} \in \mathbb{R}$ is the pairwise similarity between mentions u and v (usually computed using a trained machine learning model trained on pairs of mentions) and $e_i \in \mathbb{R}^n$ is the i^{th} standard basis vector. Note that we assume that the $w_{v\xi} = 0$ as in (Angell et al., 2022). The first constraint ensures that all of the positive features in ξ^+ are contained in the same cluster as ξ for all $\xi \in \Xi$. The last constraint ensures that ξ is not in the same cluster as anything incompatible with it for all $\xi \in \Xi$. This problem is intractable to optimize exactly, so we relax (E.6) to an SDP in the standard way as follows

$$\max_{X \succeq 0} \langle W, X \rangle \quad \text{s.t.} \quad \begin{cases} X_{vv} = 1, & \text{for all } v \in V \\ X_{uv} \geq 0, & \text{for all } u, v \in V \\ \sum_{v \in \Gamma(\xi,\phi)} X_{v\xi} \geq 1, & \text{for all } \phi \in \xi^+ \\ X_{v\xi} = 0, & \text{for all } v \perp \xi \end{cases} \tag{E.7}$$

The decision variables are changed from standard basis vectors, whose dot products determine whether or not two elements (vertices or ∃-constraints) are in the same cluster, to a positive semidefinite matrix where each entry represents the global affinity two elements have for one another (corresponding to the row and column of the entry). We constrain the diagonal of this positive semidefinite matrix to be all ones (representing each element's affinity with itself) and we enforce that all entries in the matrix are non-negative. The constraints ensuring ∃-constraint satisfaction are similar to (E.6). Let W be the (weighted) adjacency matrix of the fully connected graph over the n vertices.

E.3.3. ROUNDING

The solution $X_\star$ to (E.7) is a fractional approximation of the problem we wish to solve. In order to infer a predicted clustering, we need some procedure to round the fractional solution to an integer solution. Angell et al. (2022) propose first building a hierarchical clustering over V using $X_\star$ as the similarity function. Then, a dynamic programming algorithm is used to extract a (predicted) flat clustering from the hierarchical clustering which maximizes the number $\exists$-constraints satisfied and the (correlation) clustering objective. For a more detailed description of the rounding procedure, see (Angell et al., 2022). The predicted clustering is then observed by the human users in terms of the features of each predicted cluster enabling the users to provide any additional $\exists$-constraints to correct the entity resolution decisions. We simulate this human feedback loop with an oracle $\exists$-constraint generator.

E.3.4. EXPERIMENTAL SETUP

We evaluate the performance of the CGAL and USBS with and without warm-starting on the same three author coreference datasets used in (Angell et al., 2022). In these datasets, each mention is an author-paper pair and the goal is to identify which author-paper pairs were written by the same author. As standard in author coreference, the datasets are preprocessed into *blocks* (also known as *canopies*) based on the author's first name initial and last name (e.g. authors "Rajarshi Das" and "Ravi Das" would both be contained in the same block "`r das`", but in a different block than "Jane Smith", which would be in "`j smith`"). Within each block, similarity scores are computed between all pairs of mentions using a trained pairwise model (Angell et al., 2022; Subramanian et al., 2021). We then perform two additional preprocessing steps to convert the many small dense problems into one large sparse problem. We aggregate all of these pairwise similarity scores into a block diagonal weight matrix and fill the remaining entries with negative one. We then sparsify this highly structured similarity matrix using a spectral sparsifier (Spielman & Srivastava, 2008). Table 2 details the dataset statistics including the number of non-zeros in the pairwise similarity matrix after spectral sparsification, denoted nnz(W).

We simulate $\exists$-constraint generation using the same oracle implemented in (Angell et al., 2022). The oracle has access to the features of the ground truth clusters and the features of the predicted clusters output by the rounding algorithm and generates a small $\exists$-constraint which is satisfied by the ground truth clustering, but is not satisfied by the predicted clustering. The oracle generates $\exists$-constraints one at a time and added to the optimization problem until the ground truth clustering is predicted. For both solvers, we iterate until the relative error tolerance (19) and the following absolute infeasibility condition are satisfied

$$\|\mathcal{A}X - \text{proj}_{\mathcal{K}}(\mathcal{A}X)\|_\infty \leq \varepsilon. \tag{E.8}$$

We use $\varepsilon = 10^{-1}$ to determine the stopping condition and find that this is sufficient for the rounding algorithm to satisfy nearly all of the $\exists$-constraints generated by the oracle. We also note that the ℓ^∞-norm condition is especially important in this application in order to make sure the relaxed $\exists$-constraints are satisfied. Due to the small problem size, we simplify the implementation by using variants of CGAL and USBS without sketching.

When warm-starting, we predict the values of the expanded primal variable corresponding to the newly added $\exists$-constraint by computing representations of each vertex using a low-rank factorization, performing a weighted average of the representations of the positive vertices in the newly added $\exists$-constraint, and expanding the factorization into an initialization of primal variable for the new problem. We pad the dual variable with zeros. Since there is no pairwise similarity between $\exists$-constraints and mentions, we find that scaling up the norms of A_i's corresponding to the constraints $X_{vv} = 1$ for all v such that $\Phi(v) \cap \xi_{\text{new}} \neq \emptyset$, where ξ_{new} is the newly added $\exists$-constraint. We use the following USBS hyperparameters in our experiments: $\rho = 0.01$, $\beta = 0.25$, $k_c = 3$, $k_p = 0$. See the implementation for more details.

F. Additional Experimental Results

F.1. MaxCut

Figure 4 plots three different convergence measures over time for a large data instance, `144`, for a total time of 12 hours. For this instance, warm-starting helps achieve a faster convergence rate of both relative primal objective suboptimality and relative infeasibility. Additionally, the weight of the cut produced from the primal iterates generated by USBS drops in value less when warm-starting as compared with CGAL. This is an additional indication that USBS is able to utilize the warm-start initialization better than CGAL. Finally, observe that USBS, with and without warm-starting, produces a marginally better graph cut than CGAL.

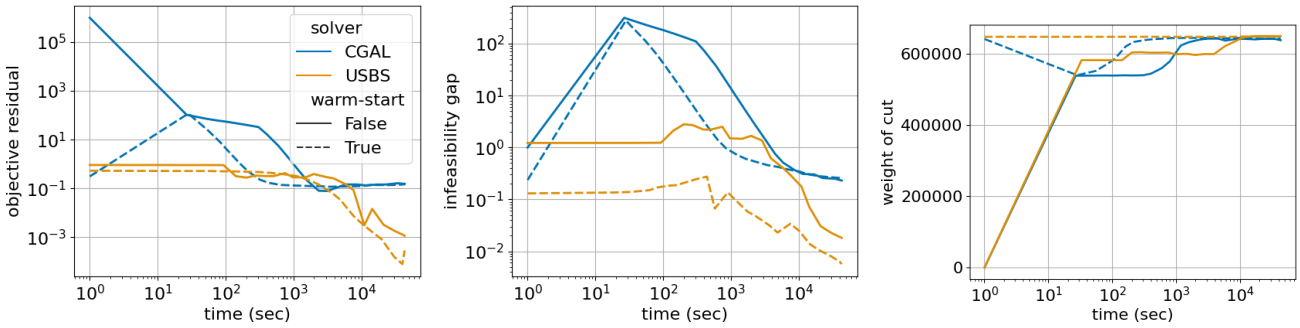
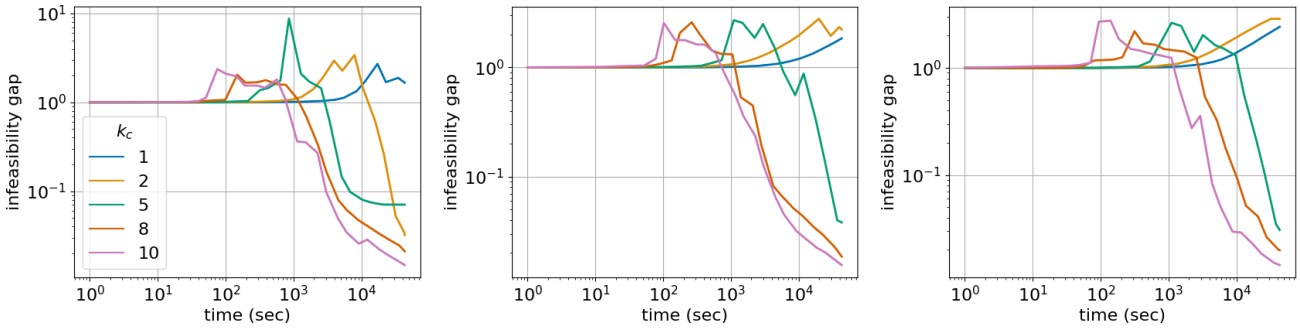


Figure 4: **Convergence measures on instance 144.** We solve instance 144 from DIMACS10 and plot the primal objective sub-optimality (objective residual, ↓), relative infeasibility (infeasibility gap, ↓), and weight of the cut (↑) produced by the rounding procedure. The warm-started runs use 99% of the original data to obtain a warm-start initialization. We observe that USBS is able to more reliably leverage a warm-start initialization. In these plots, USBS is executed with $k_c = 8, k_p = 8$. All runs were executed on a compute node with 16 cores and 128GB of RAM.

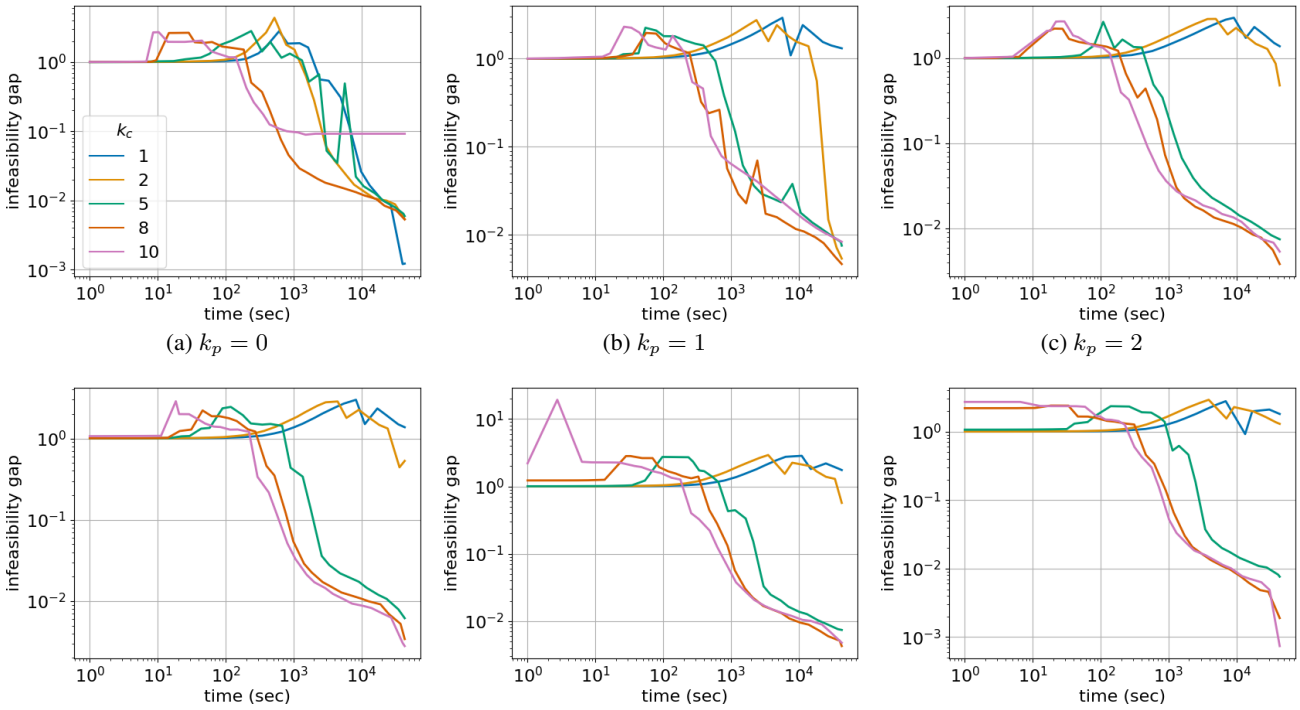


(a) $k_p = 0$ (b) $k_p = 1$ (c) $k_p = 2$

(d) $k_p = 5$ (e) $k_p = 8$ (f) $k_p = 10$

Figure 5: **Infeasibility gap vs. time for different settings of k_c and k_p.** In each plot, the x-axis is time (up to 12 hours) and the y-axis is the relative infeasibility gap. In every case, USBS is cold-started on the 144 instance from the `DIMACS10` dataset. Each plot considers one value of k_p and several values of k_c. All runs were executed on a compute node with 16 cores and 128GB of RAM. We observe that USBS performs best when $k_c \geq k_p$.

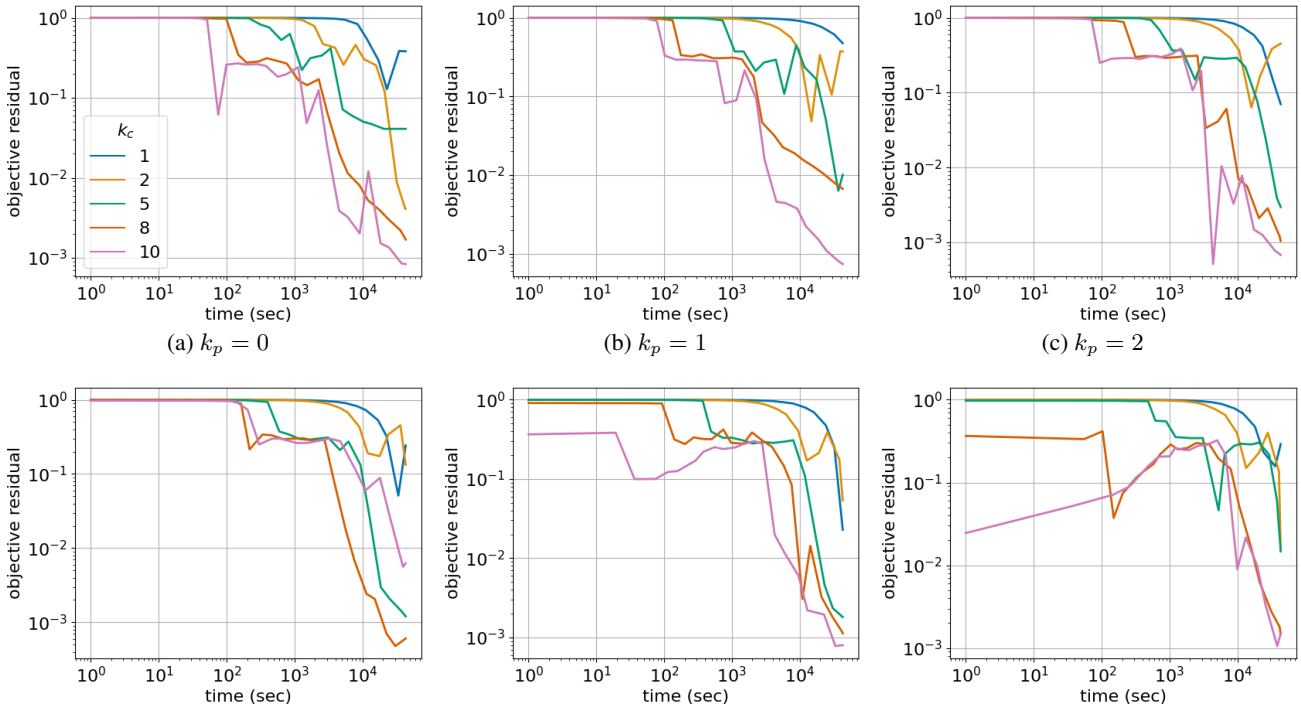
Figures 5 and 6 compare USBS's infeasibility gap against time for different settings of k_c and k_p on the 144 instance from `DIMACS10` on both CPU and GPU. Figures 7 and 8 compare USBS's infeasibility gap against time for different settings of k_c and k_p on the 144 instance from `DIMACS10` on both CPU and GPU. Figures 9 and 10 compare CGAL and USBS on CPU and GPU for different settings of k_c and k_p on the 144 instance from `DIMACS10`. We observe anywhere from 10-25x speedup on GPU as compared to CPU. We also observe that USBS performs best when $k_c \geq k_p$.



(a) $k_p = 0$ (b) $k_p = 1$ (c) $k_p = 2$

(d) $k_p = 5$ (e) $k_p = 8$ (f) $k_p = 10$

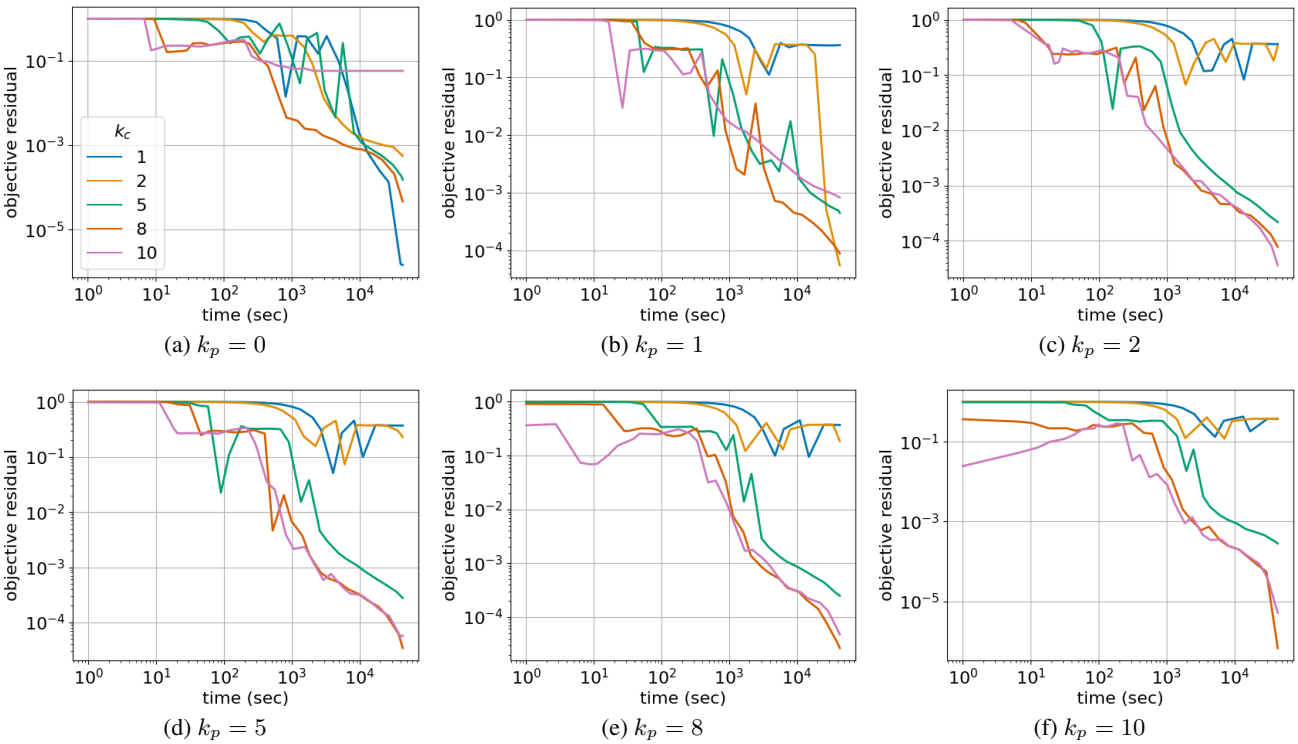
Figure 6: **Infeasibility gap vs. time for different settings of k_c and k_p.** In each plot, the x-axis is time (up to 12 hours) and the y-axis is the relative infeasibility gap. In every case, USBS is cold-started on the 144 instance from the `DIMACS10` dataset. Each plot considers one value of k_p and several values of k_c. All runs were executed on a single NVIDIA GeForce 1080 Ti GPU. We observe that USBS performs best when $k_c \geq k_p$.



(a) $k_p = 0$ (b) $k_p = 1$ (c) $k_p = 2$

(d) $k_p = 5$ (e) $k_p = 8$ (f) $k_p = 10$

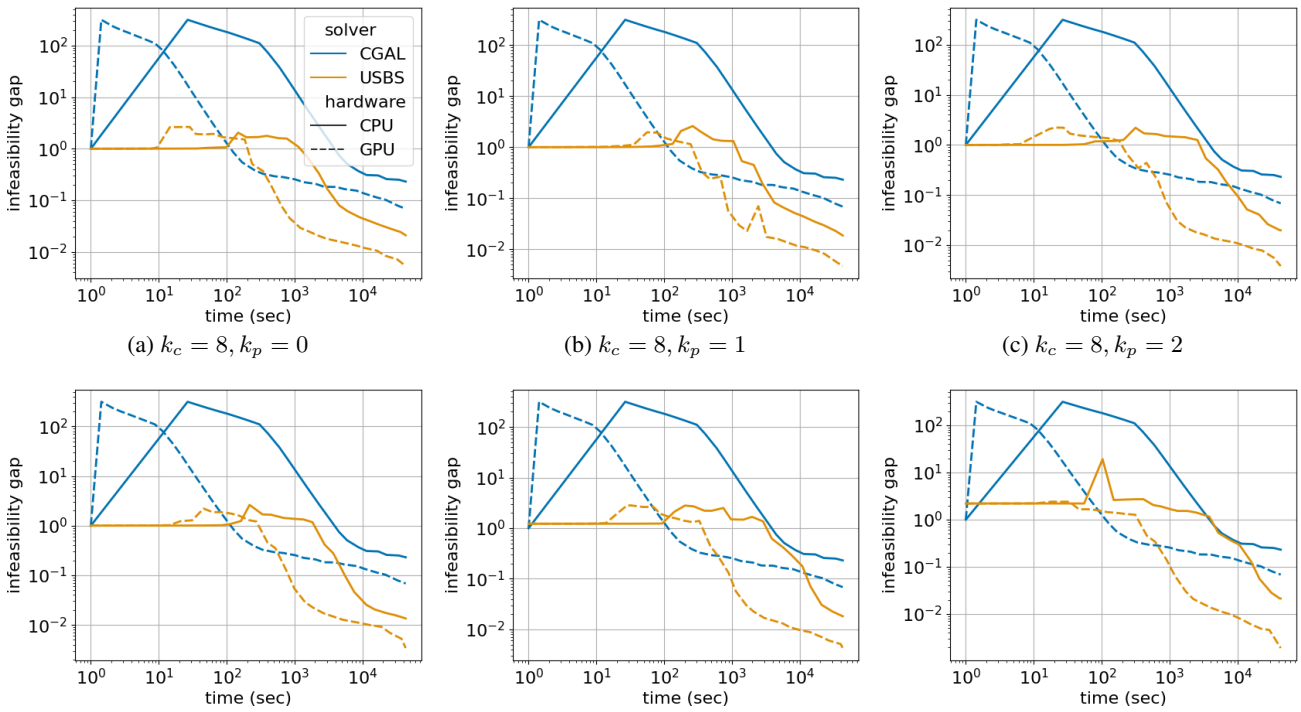
Figure 7: **Objective gap vs. time for different settings of k_c and k_p.** In each plot, the x-axis is time (up to 12 hours) and the y-axis is the primal objective suboptimality. In every case, USBS is cold-started on the 144 instance from the DIMACS10 dataset. Each plot considers one value of k_p and several values of k_c. All runs were executed on a compute node with 16 cores and 128GB of RAM. We observe that USBS performs best when $k_c \geq k_p$.



(a) $k_p = 0$ (b) $k_p = 1$ (c) $k_p = 2$

(d) $k_p = 5$ (e) $k_p = 8$ (f) $k_p = 10$

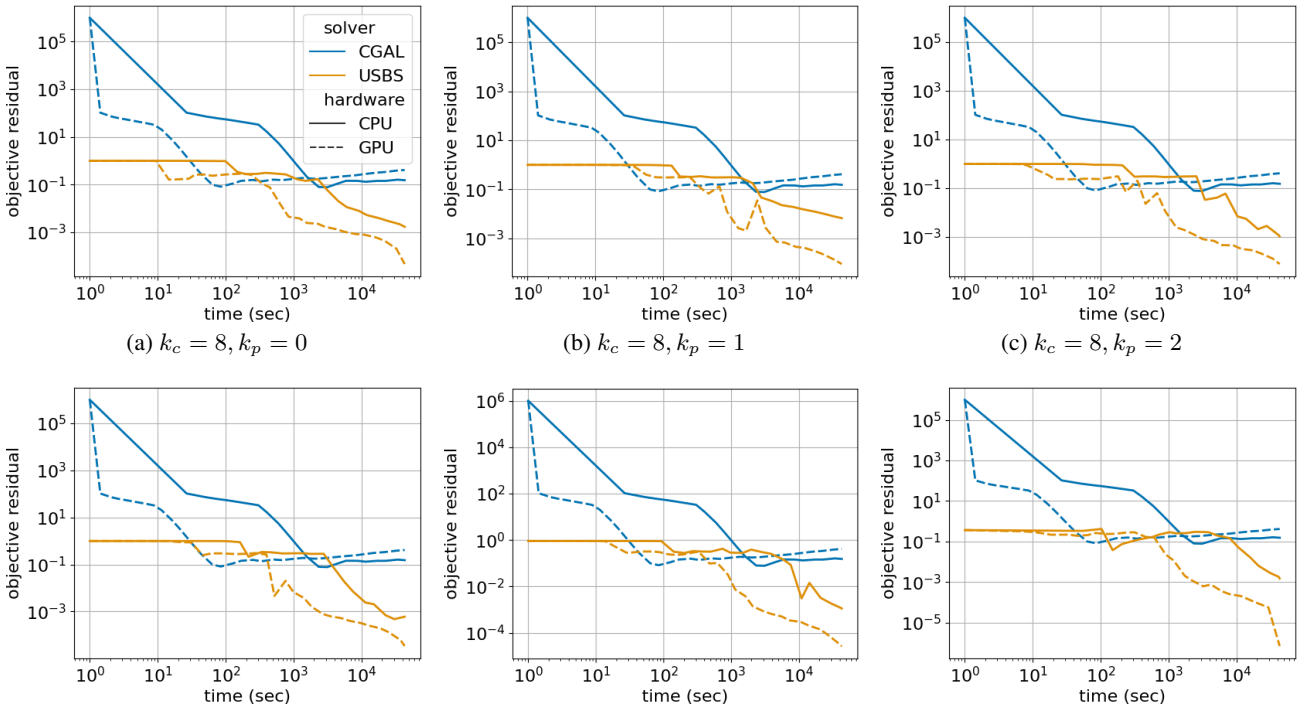
Figure 8: **Objective gap vs. time for different settings of k_c and k_p.** In each plot, the x-axis is time (up to 12 hours) and the y-axis is the relative primal objective suboptimality. In every case, USBS is cold-started on the 144 instance from the `DIMACS10` dataset. Each plot considers one value of k_p and several values of k_c. All runs were executed on a single NVIDIA GeForce 1080 Ti GPU. We observe that USBS performs best when $k_c \geq k_p$.



(a) $k_c = 8, k_p = 0$ (b) $k_c = 8, k_p = 1$ (c) $k_c = 8, k_p = 2$

(d) $k_c = 8, k_p = 5$ (e) $k_c = 8, k_p = 8$ (f) $k_c = 8, k_p = 10$

Figure 9: **Infeasibility gap vs. time on CPU and GPU.** In each plot, the x-axis is time (up to 12 hours) and the y-axis is the relative infeasibility gap. In every setting of k_c and k_p, we compare CGAL and USBS cold-started on the 144 instance from the `DIMACS10` dataset on both a compute node with 16 cores and 128GB of RAM (`CPU`) and a single NVIDIA GeForce 1080 Ti GPU (`GPU`). We observe that USBS performs best when $k_c \geq k_p$.



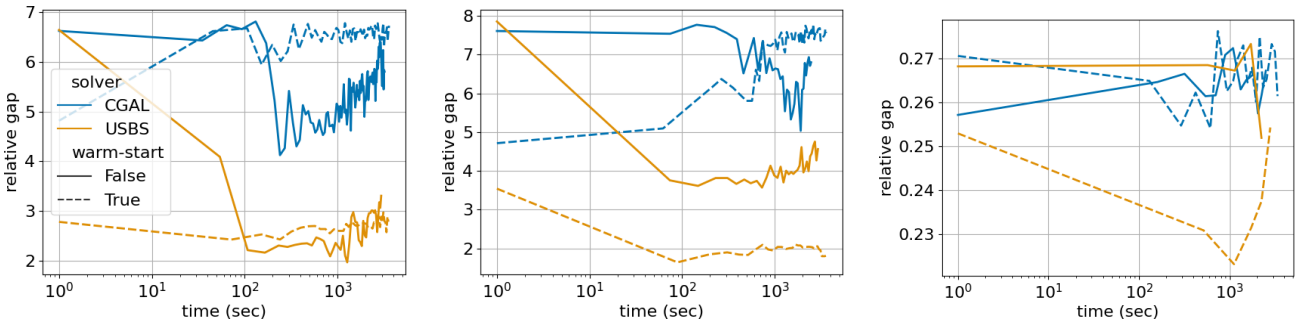
(a) $k_c = 8, k_p = 0$ (b) $k_c = 8, k_p = 1$ (c) $k_c = 8, k_p = 2$

(d) $k_c = 8, k_p = 5$ (e) $k_c = 8, k_p = 8$ (f) $k_c = 8, k_p = 10$

Figure 10: **Objective gap vs. time on CPU and GPU.** In each plot, the x-axis is time (up to 12 hours) and the y-axis is the relative primal objective suboptimality. In every setting of k_c and k_p, we compare CGAL and USBS cold-started on the 144 instance from the `DIMACS10` dataset on both a compute node with 16 cores and 128GB of RAM (`CPU`) and a single NVIDIA GeForce 1080 Ti GPU (`GPU`). We observe that USBS performs best when $k_c \geq k_p$.

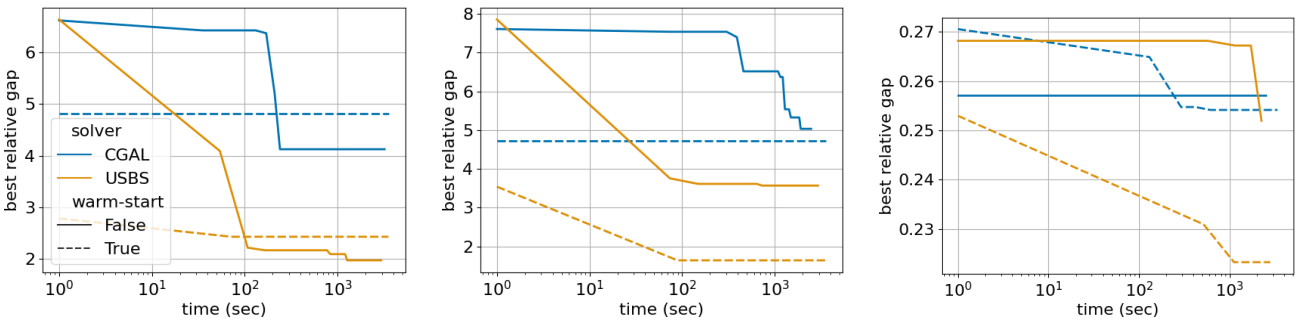
F.2. QAP

Figure 11 and Figure 12 plot `relative gap` and `best relative gap` against time (in seconds) for three instances, respectively. Figure 11 shows that even for large instances the `best relative gap` is obtained within the first few minutes of the hour of optimization. In these instances, we can also observe that USBS not only obtains a better `best relative gap`, but also is able to leverage a warm-start solution to improve performance. Figure 13 shows the `best relative gap` obtained for ten data instances over one hour of optimization. It can be seen that warm-starting does not always yield a better `best relative gap`, but USBS is better able to leverage a warm-start solution than CGAL. These results show that warm-starting is a beneficial heuristic for finding a quality approximate solution to QAPs. The warm-starting technique used in these experiments is not the only possible warm-starting initialization strategy. For examples, one could generate several warm-start initializations for (23) by dropping other rows and columns of D and W besides the last row and column. Then, after creating `n` warm-start initializations, we could take the best upper bound obtained by optimizing (23) from each of those warm-start initializations.



(a) pr136 (b) kroA150 (c) tai150b

Figure 11: **`relative gap` (↓) vs. time.** We plot the `relative gap` (y-axis) against time in seconds (x-axis) for three instances from `QAPLIB` and `TSPLIB`. We observe that for both algorithms the best rounded solution is found early in optimization. In addition, we observe that USBS is able to more reliably leverage a warm-start initialization.



(a) pr136 (b) kroA150 (c) tai150b

Figure 12: **`best relative gap` (↓) vs. time.** We plot the `best relative gap` (y-axis) against time in seconds (x-axis) for three instances from `QAPLIB` and `TSPLIB`. We observe that USBS is able to more reliably leverage a warm-start initialization.

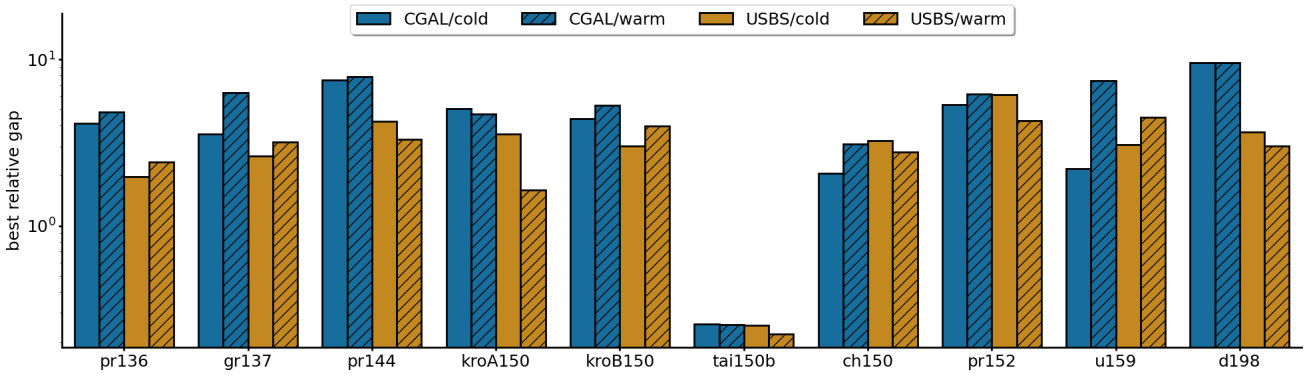


Figure 13: **`best relative gap` (↓).** The `best relative gap` obtained in one hour of optimization is shown for ten instances from `QAPLIB` and `TSPLIB`. We observe that on most problem instance that USBS produces a better relative gap than CGAL and that a warm-start initialization helps USBS obtain a better relative gap. CGAL is much less reliable in leveraging a warm-start initialization.

F.3. Interactive Entity Resolution with ∃-constraints

Figure 14 shows the average warm-start SDP solve time fold change. Warm-start fold change greater than one indicates a speedup in solve time while warm-start fold change less than one indicates a slowdown in solve time. We see that the warm-start fold change for CGAL is less than or equal to one, indicating a slowdown consistent with Figure 3. We see that on average warm-starting affords USBS a 20-100x speedup on average per ∃-constraint. Note that this is much faster than the 2-3x we see in Figure 3. This is due to the fact that sometimes warm-starting does not help USBS, which makes intuitive sense if the warm-start initialization is far away from the solution set for the new SDP. We believe that with some further experimentation including additional pairwise learned similarities between mentions and ∃-constraints or different warm-starting strategies we might be able to mitigate these situations. Regardless, in most cases, warm-starting provides a significant improvement in convergence time when using USBS. We also note that Figure 14 shows that warm-starting provides more of a benefit the larger the SDP we are trying to solve.

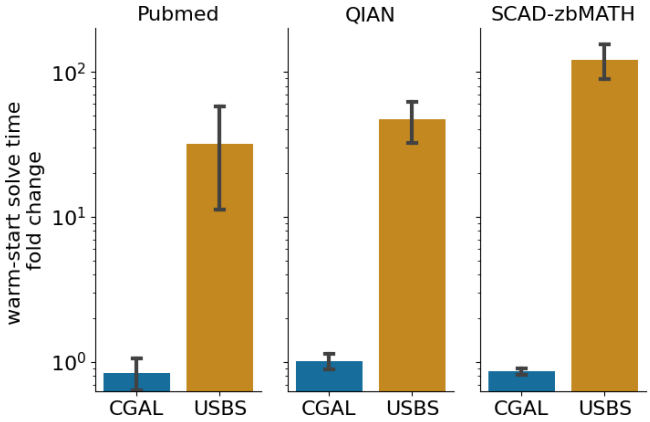


Figure 14: **Average warm-start fold change (↑).** Warm-start fold change represents the ratio of SDP solve time without warm-starting divided by the SDP solve time with warm-starting per ∃-constraint. The error bars indicate one standard deviation from the mean. We observe that USBS is able to much more reliably leverage a warm-start initialization. In addition, we observe that the performance gap between CGAL and USBS grows as the problem size grows.