FAST, SCALABLE, WARM-START SEMIDEFINITE PROGRAMMING WITH APPLICATION TO KNOWLEDGE EDITING AND MIXED INTEGER SEMIDEFINITE OPTIMIZATION

FAST, SCALABLE, WARM-START SEMIDEFINITE PROGRAMMING WITH APPLICATION TO KNOWLEDGE EDITING AND MIXED INTEGER SEMIDEFINITE OPTIMIZATION

A Dissertation Presented

by

RICO ANGELL

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2025

Manning College of Information and Computer Sciences

FAST, SCALABLE, WARM-START SEMIDEFINITE PROGRAMMING WITH APPLICATION TO KNOWLEDGE EDITING AND MIXED INTEGER SEMIDEFINITE OPTIMIZATION

A Dissertation Presented

by

RICO ANGELL

Approved as to style and content by:

Andrew K. McCallum, Chair

Daniel R. Sheldon, Member

Cameron Musco, Member

J. Zico Kolter, Member

Ramesh K. Sitaraman, Associate Dean for Educational Programs and Teaching
Manning College of Information and Computer Sciences

ABSTRACT

FAST, SCALABLE, WARM-START SEMIDEFINITE PROGRAMMING WITH APPLICATION TO KNOWLEDGE EDITING AND MIXED INTEGER SEMIDEFINITE OPTIMIZATION

FEBRUARY 2025

RICO ANGELL

B.S.E., UNIVERSITY OF MICHIGAN

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew K. McCallum

Semidefinite programming (SDP) has traditionally been limited to moderate-sized problems. Methods for scaling to larger problems have sacrificed the convexity of the original problem for scalalability. More recently, 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 constraints.

We present Unified Spectral Bundling with Sketching (USBS), a 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.

The speed and scalability of USBS enables the use of SDPs in novel applications. First, we present a new paradigm of interactive feedback, *existential cluster constraints*, for correcting entity resolution predictions and present a novel SDP relaxation at the core of the proposed inference algorithm. We demonstrate empirically that our proposed framework facilitates more accurate entity resolution with dramatically fewer user feedback inputs. We show USBS is not only faster than the previous state-of-the-art scalable SDP solver, but can also effectively leverage a warm-start initialization to improve empirical convergence.

Finally, we provide evidence that USBS could potentially be used as part of a mixed-integer semidefinite program solver. There are many applications where we want to optimize a semidefinite program where a subset of the decision variables are subject to additional integrality constraints. One of the barriers to applying standard branch-and-bound techniques to mixed-integer semidefinite programs is the lack of a fast semidefinite program solver that can warm-started. Given existing evidence that USBS can effectively utilize a warm-start initialization, we explore the possibility of using USBS as part of a branch-and-bound solver for mixed-integer semidefinite programs.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

INTRODUCTION

Semidefinite programming (SDP) is a convex optimization paradigm capable of modeling or approximating many practical problems in combinatorial optimization [3, 64], neural network verification [42, 126], robotics [129], optimal experiment design [148], global optimization of two-layer polynomial activation neural networks [19], VLSI [147], and systems and control theory [23]. 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 interior point methods, the standard method for solving SDPs [5, 146].

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 [28, 119]. 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 [11, 162, 163]. 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 [31]. 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 [153]. Yurtsever et al. [164] extend the conditional gradient augmented Lagrangian approach [163] with a matrix sketching technique [144]. 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 in [75], are an appealing framework for solving SDPs due to their low per-iteration computational complexity and fast empirical convergence. Furthermore, it has been shown under certain conditions that spectral bundle methods can attain linear convergence [46, 101]. Additionally, spectral bundle methods can be implemented without explicitly storing the primal matrix, making them amenable to the same matrix sketching techniques [144] as used by [164]. While several spectral bundle methods have been presented in the literature [8, 46, 74, 76], 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.

Practicality of Semidefinite Programming

Semidefinite programming has been useful for many important theoretical applications, including approximation algorithms for various NP-hard problems among others in machine learning, robotics, and control. The general strategy for combinatorial optimization problems is to define an optimization problem that is intractable and then use semidefinite lifting to relax the problem to a tractable convex optimization problem in the form of a semidefinite program. The solution of the SDP is then used as the input to a rounding procedure which converts the relaxed solution to a solution that is feasible in the domain of the original problem. The most famous is the Goemans-Williamson MaxCut approximation algorithm [64]. Analogous approaches have been taken for other NP-hard problems including MAX-2SAT [64], SPARSEST-CUT [10], and correlation clustering [140]. Despite these great theoretical results, semidefinite programming is often not applied to practical problems due to the lack of speed and scalability of SDP solvers.

A major contributor to the scalability issues of SDP solvers is due to the fact that SDPs are extremely expressive convex optimization paradigm, encompassing subclasses including linear programming, second-order cone programming, and quadratic programming. The goal is to find a general purpose solution to solving SDPs due to the number of problems that can be expressed as SDPs. While this is unlikely due to the generality and difficulty optimizing with respect to the semidefinite cone, a suite of solutions designed for subclasses of SDPs based on problem structure and compromises users are willing to make. This thesis contributes to a certain subclass of SDPs for which we care about the positive semidefinite constraint and not as much about fully satisfying the constraints.

In this thesis, we seek to demonstrate that a faster and more scalable approach to solve massive weakly-constrained SDPs, and thus, improve the practicality of using SDPs at real-world scale. Our approach compromises finding a strictly feasible solu-

tion for scalability. We postulate that the improved practicality of solvers shown in this thesis will enable the use of SDPs to solve problems beyond the already established problems in this context. Finally, we seek to show that this scalable approach can be warm-started, an uncommon attribute of many SDP solvers.

Summary of 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. We provide a standalone implementation of the algorithm in pure JAX [29, 56], enabling efficient execution on various hardware (CPU, GPU, TPU) and wide-spread use.

We introduce *existential cluster constraints* (i.e. $\exists$-*constraints*), a new paradigm of interactive feedback for correcting entity resolution predictions and present a novel SDP relaxation as part of a heuristic algorithm for satisfying $\exists$-constraints in the predicted clustering. We demonstrate empirically that $\exists$-constraints facilitates more accurate clustering with dramatically fewer user feedback inputs as compared to traditional must-link and cannot-link constraints. We show that USBS performs much

better than the previous state-of-the-art scalable SDP solver on this problem, especially as the problem size grows. In addition, 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.

Finally, we seek to show the potential efficacy of using USBS as part of a mixed-integer semidefinite programming solver. Many practical problems can be formulated as mixed-integer programming problems, but one of the limiting factors for mixed-integer SDPs is the lack of a reliable solution that can be efficiently warm-started. Another well-known downside of interior point methods is that they cannot be effectively warm-started. In contrast, we further support the ability the warm-start USBS by showing that it can be efficiently warm-started in the context of a branch and bound algorithm. Specifically, we implement a branch and bound algorithm for solving the mixed-integer programming version of interactive clustering with existential cluster constraints. As a proof of concept, we show that USBS can be reliably and consistently warm-started in this context to provably satisfy the existential cluster constraints. We see roughly 2x improvement in convergence on average warm-starting USBS in the context of a branch and bound on all clustering with existential cluster constraints datasets. Additionally, we observe that we need fewer existential cluster constraints to achieve the perfect ground truth clustering since the branch and bound algorithm will always find a high quality clustering that satisfies all of the constraints.

List of Completed Works

This thesis covers just a fraction of work on machine learning and optimization completed during the doctoral studies. For completeness, we include all of the publications with major contribution from this author, some of which are included in this thesis. We list the publications in reverse chronological order:

Fast, Scalable, Warm-Start Semidefinite Programming with Spectral Bundling and Sketching
Rico Angell and Andrew McCallum
ICML 2024

Interactive Correlation Clustering with Existential Cluster Constraints
Rico Angell, Nicholas Monath, Nishant Yadav, and Andrew McCallum
ICML 2022

Clustering-based Inference for Biomedical Entity Linking
Rico Angell, Nicholas Monath, Sunil Mohan, Nishant Yadav,
and Andrew McCallum
NAACL 2021

Inferring Latent Velocities from Weather Radar Data using Gaussian Processes
Rico Angell and Daniel Sheldon
NeurIPS 2018

Themis: Automatically Testing Software for Discrimination
Rico Angell, Brittany Johnson, Yuriy Brun and Alexandra Meliou
ESEC/FSE 2018

CHAPTER 1

SEMIDEFINITE PROGRAMMING

Semidefinite programming is a convex optimization paradigm concerned with optimizing a linear objective over the intersection of an affine space with the cone of positive semidefinite matrices. This description is technically sound, but does not provide much intuition about where semidefinite programs could be useful or how we should approach optimizing them. We instead will approach semidefinite programming in a more accessible and intuitive way for those readers with some basic machine learning knowledge. All semidefinite programs can be expressed in the following form

$$\begin{aligned} &\max_{x_1,\ldots,x_n \in \mathbb{R}^n} c_{ij}(x_i^\top x_j) \\ &\text{s.t.} \sum_{i,j\in[n]} a_{ijk}(x_i^\top x_j) \leq b_k, \quad k = 1,\ldots,m \end{aligned} \tag{1.1}$$

where $c_{ij}, a_{ijk}, b_k \in \mathbb{R}$ are the scalar *problem data* and the n, n-dimensional, vectors $x_i \in \mathbb{R}^n$ are called the *decision variables* ($[n] := \{1, 2, \ldots, n\}$). From a machine learning perspective, we can think about the decision variables x_i as embeddings for n different objects and we want to *learn* the embeddings that maximize the linear objective subject to linear constraints. For example, this formulation provides an intuitive path to canonical relaxations of NP-hard combinatorial optimization problems.

Example 1 (MaxCut)**.** *Given a graph $G = (V, E)$ the goal of the maximum cut problem is to find a partitioning of the vertices V such that the number of edges*

crossing the partitioning is maximized. This problem can be written formally as the following integer quadratic program

$$\max_{v_1,v_2,\ldots,v_n \in \{-1,1\}} \sum_{(i,j)\in E} \frac{1 - v_i v_j}{2}.$$

The decision variables use either -1 *or* 1 *to determine the side of the cut and the product of any two decision variables corresponding to an edge determines if the edge crosses the cut or not. Unfortunately, this problem is NP-hard to solve exactly, but we can relax it to the following well-known SDP*

$$\max_{v_1,v_2,\ldots,v_n \in \mathbb{R}^n} \sum_{(i,j)\in E} \frac{1 - v_i^\top v_j}{2}$$
$$\text{s.t. } \|v_i\|_2^2 = 1, \quad \forall i \in [n]$$

In this case, the n *different objects we want to embed are the vertices of a given graph. A high quality approximate solution can be obtained by solving this SDP and then using a randomized rounding scheme to select a cut [64].*

Using this intuitive formulation of SDPs it is natural to see how SDPs could be applied to various graph problems such as clustering [140, 167] and geometric representations of graphs [81, 104]. It should also be mentioned that we should recall the many other applications of SDPs mentioned in the introduction which do not obviously fit into this framework. For more applications of semidefinite programming from an operations research and control theory perspective, [105, 145] provide good surveys. In order to understand those other applications it is useful to formulate SDPs in their standard form.

1.1 Standard Form

While the SDP formulation provided in (1.1) is intuitive, it is not the standard form in which SDPs are normally written. In order to detail the standard form SDP we need to introduce some concepts and notation.

Decision Variables. First, the dot products $x_i^\top x_j$ in the objective of (1.1) are normally organized into a matrix $X \in \mathbb{R}^{n\times n}$ such that $X_{ij} = x_i^\top x_j$. Throughout the rest of this document we refer to X as the decision variable. First, notice that X is symmetric, i.e. $X = X^\top$ ($X_{ij} = X_{ji},\ \forall i, j \in [n]$). Denote the set of $n \times n$ symmetric matrices $\mathbb{S}^n$. The set of symmetric matrices is equipped with an inner product defined as $\langle A, B\rangle = \mathrm{tr}(A^\top B) = \sum_{i,j\in[n]} A_{ij}B_{ij}$ for any $A, B \in \mathbb{S}^n$, where $\mathrm{tr}(A)$ is the trace of A. The matrix X is referred to commonly as the *Gram matrix*. Furthermore, a matrix is a Gram matrix if and only if it is *positive semidefinite*, hence the name *semidefinite* programming. Before turning to the standard definition of semidefinite matrices we must introduce additional concepts.

Eigenvectors and Eigenvalues. Let $M \in \mathbb{S}^n$. An *eigenvector* of M is a vector $q \in \mathbb{R}^n$ that is parallel to Mq, i.e. $Mq = \lambda q$ where the scaling factor $\lambda \in \mathbb{R}$ is the corresponding *eigenvalue*. It is convention to scale q to have unit-norm. In addition, any symmetric matrix can be factorized into the following form

$$M = Q\Lambda Q^\top \tag{1.2}$$

where $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix (orthonormal columns, i.e. $Q^\top = Q^{-1}$) whose i^{th} column is the eigenvector q_i of M and Λ is a diagonal matrix with the corresponding eigenvalues oriented along the diagonal (i.e. $\Lambda_{ii} = \lambda_i$). This factorization is referred to as the *eigendecomposition* of a matrix. A matrix is *positive semidefinite* if and only if all of its eigenvalues are non-negative, which is written as $M \succeq 0$. We

denote the *cone* of $n \times n$ positive semidefinite matrices by $\mathbb{S}^n_+$. Given any positive semidefinite matrix, we can recover the vectors that make it a Gram matrix by taking it's eigendecomposition and distributing the square roots of the eigenvalues to the orthonormal eigenvector matrices. It is also important to observe that the trace of a matrix is equal to the sum of its eigenvalues, i.e. $\text{tr}(M) = \sum_{i\in[n]} M_{ii} = \text{tr}(\Lambda) = \sum_{i\in[n]} \lambda_i$, and the determinant of a matrix is the product of its eigenvalues. Given this, it is easy to see that the diagonal of a positive semidefinite matrix is also always non-negative.

Standard Form. Given this definition of semidefinite matrices, we can rewrite (1.1) as the following standard form SDP

$$\begin{aligned} \max_{X\in\mathbb{S}^n_+} \quad & \langle C, X\rangle \\ \text{s.t.} \quad & \mathcal{A}X = b \end{aligned} \tag{1.3}$$

where $C \in \mathbb{S}^n$ is the cost matrix (i.e. $c_{ij} = C_{ij} \in \mathbb{R}$), $\mathcal{A} : \mathbb{S}^n \to \mathbb{R}^m$ is a linear operator, and $b \in \mathbb{R}^m$. The constraint map $\mathcal{A}$ and it's adjoint $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{S}^n$, i.e. $\langle \mathcal{A}X, y\rangle = \langle X, \mathcal{A}^* y\rangle$ for any $X \in \mathbb{S}^n$ and $y \in \mathbb{R}^m$, 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 $A_i \in \mathbb{S}^n$. One observation about the standard form SDP (1.3) is that there are no inequality constraints, only equality constraints. This makes it easier to develop methodology and has the added benefit of still being able to express all of the problems expressed by (1.1) via slack variables as we will see soon. Unfortunately, this standard form does have its empirical drawbacks when it comes to solving problems with many inequality constraints.

Slack Variables. In order to incorporate inequality constraints, we need to add *slack variables*, i.e. add dimensions to the decision variable to take up the slack of the inequality constraint. Recall that a consequence of X being a positive semidefinite matrix is that the diagonal elements are all non-negative. So, in order to incorporate the following simple constraint

$$X_{ij} \geq 0,$$

we just need to expand X by a single dimension and rewrite this constraint as the following equality constraint

$$X_{ij} - X_{n+1,n+1} = 0.$$

Since $X_{n+1,n+1} \geq 0$ due to $X \succeq 0$, this equality constraint can be satisfied if and only if $X_{ij} \geq 0$. This is not too big of a problem if there are a small number of inequality constraints, but can become quite costly if the number of inequality constraints is large. For example, in the standard correlation clustering SDP, we have n^2 inequality constraints, meaning that the size of the decision variable would have over n^4 entries in slack variable form. Furthermore, we might want to have even more complicated constraints such as

$$|X_{ij}| + |X_{k\ell}| \leq 1.$$

In this case, we actually need to create six slack variables for this single constraint. First, we create need to create two slack variables Y_1 and Y_2 to remove the absolute value constraints to get the following constraints

$$-Y_1 \leq X_{ij} \leq Y_1$$

$$-Y_2 \leq X_{k\ell} \leq Y_2$$

$$Y_1 + Y_2 = 1$$

Then, we need to create four slack variables to handle the four inequalities above. This is a very small example, but we can see how slack variables could become practically problematic. There are some ways to deal with this, some of which we will visit later in this thesis.

Duality. As is the case with most constrained convex optimization, the dual is an important part of solving SDPs. The dual of the standard formulation is as follows

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & \langle b, y \rangle \\ \text{s.t.} \quad & C - \mathcal{A}^* y \preceq 0 \end{aligned} \tag{1.4}$$

Denote the solution sets of (1.3) and (1.4) by $\mathcal{X}_\star$ and $\mathcal{Y}_\star$, respectively. It is a standard assumption that (1.3) and (1.4) 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 [4] and many well-structured SDPs [47].

1.2 Solving SDPs

We will now briefly discuss several methods used for solving SDPs each with their own advantages and disadvantages. This is by no means a comprehensive review, but we give a broad overview of methods used to solve SDPs in practice with a special

focus on the methods relevant to this thesis (see [111] for additional information). We will describe each method in terms of how it solves the standard form SDP given in (1.3). For each method, we will discuss the trade-off made between convergence rate and scalability, the main challenge when it comes to solving SDPs.

1.2.1 Interior Point Methods

Primal-dual path-following interior point methods [3–5, 73] (i.e. *interior point methods*) are by far the most popular algorithm for solving SDPs. The main idea of these methods is to optimize the Lagrangian of the dual barrier function using Newton's method on a linearized version of the system of equations generated from the first-order optimality conditions of the Lagrangian. The most common barrier function used is the log-determinant barrier function. The contribution of the value of this barrier function is controlled by a scalar μ. All of the iterates are strictly feasible and converge to the optimal solution along a *central path*.

Interior point methods can solve small SDPs very efficiently and are the algorithm of choice for this reason. In fact, under standard assumptions interior point methods can solve SDPs in polynomial time. Interior point methods can find an ε-approximate solution in $\mathcal{O}(\log(1/\varepsilon))$ Newton's method iterations. Unfortunately, each iteration requires at least $\mathcal{O}(n^6)$ time to compute the update step. Additionally, there are several numerical stability issues and degeneration that can happen with interior point methods. Furthermore, it is unknown how to warm-start these methods effectively, making mixed-integer semidefinite programming more inefficient than its linear counterpart. This trade-off is fine and manageable for small problems, but has an extremely difficult time scaling. Regardless, there are several packages and implementations of interior point methods and are some of the most widely used in practice (e.g. Mosek [9]). As a forward pointer, we derive, implement, and utilize two

primal-dual path-following interior point methods in our proposed method which we detail in Appendix A.1.

1.2.2 Douglas-Rachford Splitting and ADMM

In order to be able to scale to larger problem sizes than can be reasonably solved with interior point methods, first-order methods have been utilized to solve SDPs. Among these, one of the most effective is splitting methods, e.g. Douglas-Rachford Splitting and ADMM [28, 51, 53, 57, 63]. While these methods can be applied to more general conic programs, they have been shown useful for solving moderately-sized SDPs to modest accuracy quickly. The most widely used method and implementation is the splitting conic solver [119, 121].

ADMM-based methods have several interpretations, but one that is commonly used in convex optimization is the concept of an augmented Lagrangian. We will explain a simpler, but conceptually relevant, algorithm for solving (1.3). First, consider the *augmented* standard SDP

$$\begin{aligned} \max_{X \succeq 0} \quad & \langle C, X \rangle - \frac{\rho}{2} \|\mathcal{A}X - b\|^2 \\ \text{s.t.} \quad & \mathcal{A}X = b \end{aligned}$$

The augmented Lagrangian the standard form SDP is just the Lagrangian of the augmented problem given as follows

$$\mathcal{L}_\rho(X, y) = \langle C, X \rangle + \langle y, \mathcal{A}X - b \rangle - \frac{\rho}{2} \|\mathcal{A}X - b\|_2^2. \tag{1.5}$$

The standard augmented Lagrangian approach is to alternate between optimizing with respect to the primal variable and the dual variable as follows

$$\begin{aligned} X_{t+1} &\leftarrow \arg\max_{X \in \mathbb{S}^n} \mathcal{L}_\rho(X, y_t), \\ y_{t+1} &\leftarrow \arg\min_{y \in \mathbb{R}^m} \mathcal{L}_\rho(X_{t+1}, y) \end{aligned} \tag{1.6}$$

In many situations, there are also rules for updating the penalty parameter $\rho > 0$ over time depending on the infeasibility of the current iterate relative to past iterates. This approach is much more efficient than interior point methods, but still struggles when n starts to become larger. This is because the primal step takes at least $\mathcal{O}(n^3)$ time since an eigendecomposition needs to be computed in order to project the primal iterate on to the semidefinite cone. Hence, splitting methods are suitable for solving moderate-sized SDPs to modest accuracy, but do not scale to large SDPs. It is also important to note that these methods can easily incorporate other types of constraints and are easily warm-started given a good primal-dual pair warm-start initialization.

1.2.3 SDPLR and ManiSDP

In many important applications, the SDPs are sizeable and have 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 also much less n^2, often being proportional to n or the number of non-zero entries in C. Often SDPs of this form are referred to as *weakly-constrained*. The most notable method is SDPLR [31], also know as the *Burer-Monteiro factorization heuristic*. The main idea is to replace the primal variable $X \succeq 0$ with a low-rank factorization $X = VV^\top$ where $V \in \mathbb{R}^{n \times r}$ is unconstrained and $r < n$ (often times $r \ll n$ for scalability reasons) which gives the following non-convex quadratically constrained quadratic program (QCQP)

$$\begin{aligned} \max_{V \in \mathbb{R}^{n \times r}} \quad & \langle C, VV^\top \rangle \\ \text{s.t.} \quad & \mathcal{A}(VV^\top) = b \end{aligned} \tag{1.7}$$

Connecting this problem back to our initial embedding formulation, we can think about V as the aggregation of r-dimensional embeddings for our n objects. This optimization problem eliminates the difficult constraint $X \succeq 0$ and enables memory

savings for large problems, but unfortunately introduces non-linear (and non-convex, in general) objective and constraints. In order to optimize (1.7), SDPLR simply applies the augmented Lagrangian method. The augmented Lagrangian for (1.7) is as follows

$$\mathcal{L}_\rho(V, y) = \langle C, VV^\top \rangle + \langle y, \mathcal{A}VV^\top - b \rangle - \frac{1}{2\rho}\|\mathcal{A}VV^\top - b\|_2^2. \tag{1.8}$$

In SDPLR, L-BFGS [102] is used to optimize the primal step $V_t \leftarrow \arg\max_{V \in \mathbb{R}^{n \times r}} \mathcal{L}_\rho(V, y_t)$. Recently, ManiSDP [154] instead use a Riemannian trust-region method in order to escape from saddle points when the problem is non-convex.

It is fairly well known that in order for (1.7) to be convex, the rank of the factorization must be $r = \mathcal{O}(\sqrt{m})$ [20, 124], which is not scalable in most cases. While this is not scalable for most problems, Huang and Gleich [80] propose to modify the rank r dynamically based on optimality bounds. In addition, there are specific instances of MaxCut where SDLPR have difficulties due to its non-convexity [153].

1.2.4 CGAL and SketchyCGAL

Still concerned about scalability and solving weakly constrained SDPs, [163] introduced the conditional gradient augmented Lagrangian (CGAL). This method does not make any compromises by factorizing the primal variable and handles the primal step of the augmented Lagrangian approach using a Frank-Wolfe (i.e. conditional gradient) step. As an additional assumption, this method assumes that $\operatorname{tr}(X_\star) \leq \alpha$ for some fixed and known $\alpha \in \mathbb{R}$ for all $X_\star \in \mathcal{X}_\star$. In this case, the primal step involves two steps:

$$\begin{aligned} H_t &\leftarrow \underset{H \succeq 0 : \operatorname{tr}(H) \leq \alpha}{\arg\max} \langle \nabla_X \mathcal{L}_\rho(X_t, y_t), H \rangle \\ &\qquad \in \operatorname{conv}\{\alpha v v^\top : v \text{ is a maximum eigenvector of } \nabla_X \mathcal{L}_\rho(X_t, y_t)\} \\ X_{t+1} &\leftarrow (1 - \eta_t) X_t + \eta_t H_t, \qquad \text{where } \eta_t = \frac{2}{t+1} \end{aligned} \tag{1.9}$$

where the update direction H_t is a rank-one matrix. Using an eigenvalue decomposition based argument we can see how the optimal update direction is determined. Yurtsever et al. [163] show that this algorithm converges to an optimal solution always without changing the problem structure or convexity. Other methods combine this Frank-Wolfe style update with Burer-Monteiro factorization heuristic for performance benefits [116].

This method, as described here, still requires storing the entire primal variable which is the memory requirement bottleneck for weakly-constrained problems. As an answer to this problem, Yurtsever et al. [164] introduce SketchyCGAL, showing how to incorporate matrix sketching techniques in order to store a low-rank projection of the primal variable, greatly reducing the memory requirements of the algorithm. This sketching technique, as we will describe in more detail later, allows the algorithm to use much less memory and still have provable guarantees about solution quality under some simple rank assumptions without sacrificing the convexity of the problem. In fact, on the MaxCut problems difficult for SDPLR to solve, SketchyCGAL is able to solve them in very little time and much less space than required by SDPLR. This will be our baseline throughout this thesis for our proposed method due to it's combined scale and provable convergence.

CHAPTER 2

UNIFIED SPECTRAL BUNDLING WITH SKETCHING

In this chapter, we will present Unified Spectral Bundling with Sketching (USBS), our proposed algorithm for solving SDPs that is fast, scalable, and can effectively utilize a warm-start initialization. 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.

This work was originally published as part of the following publication:

> **Fast, Scalable, Warm-Start Semidefinite Programming with Spectral Bundling and Sketching**
> Rico Angell and Andrew McCallum
> ICML 2024

2.1 Preliminaries

We begin with the notation necessary to define the semidefinite programming problem we seek to solve. For any matrix $M \in \mathbb{S}^n$ let $\lambda_{\max}(M)$ denote the maximum eigenvalue of M and let $\text{tr}(M)$ denote the trace of M. 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 subset of components corresponding to $\mathcal{I}$ from an arbitrary 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:

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

Note that while all SDPs of this form can technically be written as equality constrained SDPs (i.e. *standard-form* SDPs introduced in the previous chapter), doing so in practice is ill-advised since the problem dimension, n, grows by one for every inequality constraint. This increase in problem dimension is critically inefficient when solving problems of already high dimension. 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 $\text{dist}(\mathcal{A}X, \mathcal{K})$. We use $\text{dist}(z, \mathcal{Z})$ to denote the Euclidean distance from a point z to a closed set $\mathcal{Z}$ and $\text{proj}_{\mathcal{Z}}(z)$ to denote the Euclidean projection of z onto $\mathcal{Z}$ (i.e. the closest point in $\mathcal{Z}$ to z). Equivalently,

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

We denote the maximum nuclear norm of the primal solution set as $\mathfrak{N}(\mathcal{X}_\star) := \sup_{X_\star \in \mathcal{X}_\star} \|X_\star\|_*$. We denote the maximum norm of the dual given the level sets of f as $\mathfrak{N}(y_0) := \sup_{f(y) \leq f(y_0)} \{\|y\|\}$.

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 also much less n^2, often being proportional to n or the number of non-zero entries in C. Recall, we refer to SDPs with these properties as weakly-constrained. We would like to exploit these features with a provably fast and scalable optimization algorithm that can solve any SDP.

We use $\|\cdot\|$ to denote the ℓ^2-norm on vectors. We use $\|\cdot\|_\infty$ to represent the ℓ^∞-norm. We use $\|\cdot\|_{\mathrm{F}}$ to denote the Frobenius norm, $\|\cdot\|_*$ to denote the nuclear norm, and $\|\cdot\|_{\mathrm{op}}$ to denote the operator norm. We use $\mathcal{O}(\cdot)$ to represent standard big-O asymptotic notation. We use $\mathrm{nnz}(M)$ to denote the number of non-zero elements in a (sparse) matrix M.

2.1.1 Proximal Bundle Methods

Proximal bundle methods [52, 87, 96, 114] 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, $\hat{f}_t$:

$$\tilde{y}_{t+1} \leftarrow \operatorname*{arg\,min}_{y\in\mathbb{R}^m} \hat{f}_t(y) + \frac{\rho}{2}\|y - y_t\|^2 \tag{2.1}$$

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.

$$\beta(f(y_t) - \hat{f}_t(\tilde{y}_{t+1})) \le f(y_t) - f(\tilde{y}_{t+1}) \tag{2.2}$$

for some fixed $\beta \in (0, 1)$. The iterations where (2.2) 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 (2.2) is satisfied or not, $\tilde{y}_{t+1}$ is used to construct the next model $\hat{f}_{t+1}$.

2.1.2 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 $\partial f(y) := \{g : f(y') \ge f(y) + \langle g, y' - y\rangle,\ \forall y' \in \mathbb{R}^m\}$

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.***

$$\hat{f}_{t+1}(y) \leq f(y), \forall y \in \mathbb{R}^m \tag{2.3}$$

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

$$\hat{f}_{t+1}(y) \geq f(\tilde{y}_{t+1}) + \langle g_{t+1}, y - \tilde{y}_{t+1} \rangle, \forall y \in \mathbb{R}^m \tag{2.4}$$

3. ***Model subgradient lower bound.*** The first order optimality conditions for (2.1) 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,

$$\hat{f}_{t+1}(y) \geq \hat{f}_t(\tilde{y}_{t+1}) + \langle s_{t+1}, y - \tilde{y}_{t+1} \rangle, \forall y \in \mathbb{R}^m \tag{2.5}$$

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.

2.1.3 Convergence Guarantees.

Recently, Díaz and Grimmer [45] proved non-asymptotic convergence rates for the proximal bundle method under a wide variety of conditions. The following theorem summarizes the results relevant to this work (Theorems 2.1 and 2.3 from [45]).

Theorem 2. *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 (2.3), (2.4), *and* (2.5). *If* $D := \sup_{t\geq 0}\{\operatorname{dist}(y_t, \mathcal{Y}_\star)\} < +\infty$ *and* $G := \sup_{t\geq 0}\{\|g_{t+1}\|\} < +\infty$ *, then the iterates* y_t *have* $f(y_t) - f(y_\star) \leq \varepsilon$ *for all*

$$t \geq \mathcal{O}\left(\frac{\rho G^2 D^4}{\beta(1-\beta)^2\varepsilon^3}\right).$$

Additionally, if $f(y) - f(y_\star) \geq \mu \operatorname{dist}^2(y, \mathcal{Y}_\star)$ *for all* y *and some constant* $\mu > 0$*, then the bound improves to* $f(y_t) - f(y_\star) \leq \varepsilon$ *for all*

$$t \geq \mathcal{O}\left(\frac{G^2}{\beta(1-\beta)^2 \min\{\mu/\rho, 1\}\varepsilon}\right).$$

2.2 Unified Spectral Bundling

To apply the proximal bundle method, we need to reformulate the dual SDP as an unconstrained optimization problem. Begin by considering the following Lagrangian formulation of the model problem

$$p_\star = \max_{X\in\mathbb{S}^n_+} \min_{y\in\mathbb{R}^m : y_{\mathcal{I}}\geq 0} \mathcal{L}(X, y) = \min_{y\in\mathbb{R}^m : y_{\mathcal{I}}\geq 0} \max_{X\in\mathbb{S}^n_+} \mathcal{L}(X, y) = d_\star, \tag{2.6}$$

where the Lagrangian is defined as

$$\mathcal{L}(X, y) := \langle C - \mathcal{A}^* y, X\rangle + \langle b, y\rangle. \tag{2.7}$$

Let $\mathcal{Y} := \{y \in \mathbb{R}^m : y_{\mathcal{I}} \geq 0\}$. Since $\mathcal{X}_\star \subset \mathcal{X}$, we know

$$\max_{X\in\mathbb{S}^n_+} \min_{y\in\mathcal{Y}} \mathcal{L}(X, y) = \max_{X\in\mathcal{X}} \min_{y\in\mathcal{Y}} \mathcal{L}(X, y) = \min_{y\in\mathcal{Y}} \max_{X\in\mathcal{X}} \mathcal{L}(X, y). \tag{2.8}$$

Furthermore, it is easy to show the following fact [120],

$$\max_{X\succeq 0\,,\,\operatorname{tr}(X)\leq 1} \langle Z, X\rangle = \max\{\lambda_{\max}(Z), 0\}. \tag{2.9}$$

Incorporating (2.9) with (2.8) 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. Minimizing (pen-D) is equivalent to optimizing (D) in the sense that the optimal solution set and objective are the same. In the following subsections, we will detail how the model is constructed and how to compute the candidate iterates.

2.2.1 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 $\alpha \geq 2\mathfrak{M}(\mathcal{X}_\star)$, 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 $\iota_{\mathcal{Y}}(\cdot)$. Consider the dual form of the indicator function $\iota_{\mathcal{Y}}(\cdot)$

$$\iota_{\mathcal{Y}}(y) = \sup\{-\langle \nu, y\rangle : \nu_{\mathcal{I}} \leq 0, \nu_{\mathcal{I}'} = 0\}. \tag{2.10}$$

Define $\mathrm{N} := \{\nu \in \mathbb{R}^m : \nu_{\mathcal{I}} \leq 0, \nu_{\mathcal{I}'} = 0\}$. Substituting (2.10) into (pen-D) and utilizing (2.9) gives

$$f(y) = \sup_{(X,\nu)\in\mathcal{X}\times\mathrm{N}} \langle C - \mathcal{A}^* y, X\rangle + \langle b - \nu, y\rangle. \tag{2.11}$$

Remark 3. *This reformulation is inspired by the formulation introduced by Helmberg and Kiwiel [74]. In this work, we differ from [74] in the model, how we solve*

the subproblems, and the fact that we experiment with both equality and inequality constraints, not just inequality constraints. That being said this particular dual slack variable supremum reformulation is given almost exactly from [74]. As far as we know, this is the only other work that has incorporated inequality constraints explicitly.

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 (2.11) over a low-dimensional subspace of $\mathcal{X}$ such that (2.3), (2.4), and (2.5) 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

$$\widehat{\mathcal{X}}_t := \left\{ \eta\bar{X}_t + V_t S V_t^\top \;\middle|\; \begin{array}{l} \eta\,\mathrm{tr}(\bar{X}_t) + \mathrm{tr}(S) \leq \alpha \\ \eta \geq 0, S \in \mathbb{S}^k_+ \end{array} \right\}. \tag{2.12}$$

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 $k_c \geq 1$ current eigenvectors and $k_p \geq 0$ 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 $\mathrm{tr}(\bar{X}_t) \leq \alpha$ and enables the last model condition (2.5) to be satisfied. Hence, the model is

$$\hat{f}_t(y) := \sup_{(X,\nu)\in\widehat{\mathcal{X}}_t\times\mathrm{N}} \langle C - \mathcal{A}^*y, X\rangle + \langle b - \nu, y\rangle. \tag{2.13}$$

We show that this model satisfies the conditions (2.3), (2.4), and (2.5) in Section 2.2.3.

Remark 4. *Helmberg and Rendl [75] do not prove non-asymptotic convergence rates, but we follow a very similar proof technique method to Ding and Grimmer [46], which*

relies heavily on the new results of Díaz and Grimmer [45] for general proximal bundle methods.

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

$$\min_{y\in\mathbb{R}^m} \sup_{(X,\nu)\in\widehat{\mathcal{X}}_t\times\mathrm{N}} \langle C-\mathcal{A}^*y, X\rangle + \langle b-\nu, y\rangle + \frac{\rho}{2}\|y-y_t\|^2, \tag{2.14}$$

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

$$X_{t+1} = \eta_{t+1}\bar{X}_t + V_t S_{t+1} V_t^\top. \tag{2.15}$$

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

$$S_{t+1} = Q_{\overline{p}}\Lambda_{\overline{p}}Q_{\overline{p}}^\top + Q_{\underline{c}}\Lambda_{\underline{c}}Q_{\underline{c}}^\top, \tag{2.16}$$

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

$$\bar{X}_{t+1} \leftarrow \eta_{t+1}\bar{X}_t + V_t Q_{\underline{c}}\Lambda_{\underline{c}}Q_{\underline{c}}^\top V_t^\top. \tag{2.17}$$

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 $C - \mathcal{A}^*\tilde{y}_{t+1}$.

Remark 5. *For the sake of the theoretical results, we only need the accuracy of the top eigenvector to be extremely accurate. We use a thick-restart Lanczos algorithm [77, 155] to compute these top eigenvectors. The vast majority of the computation required*

to compute the eigenvectors comes from sparse matrix-vector multiplication. This is dependent on the number of non-zeros in $C - \mathcal{A}^* y_t$*. In addition, we expect to need* $\mathcal{O}(\log(n))$ *iterations for some large constant to guarantee a fixed level of accuracy in the eigenvectors [92]. In practice, it seems like using a relatively low number of matrix-vector multiplications and restarts seems to work well enough.*

Then, we set V_{t+1} to k orthonormal vectors that span the columns of $[V_t Q_{\overline{p}}; v_1, \ldots, v_{k_c}]$ as follows

$$V_{t+1} \leftarrow \texttt{orthonormalize}\Big([V_t Q_{\overline{p}}; v_1, \ldots, v_{k_c}]\Big), \tag{2.18}$$

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

Remark 6. *This is essentially the same model as presented by Ding and Grimmer [46], except for the fact that the penalized dual formulation allows for explicit inequality modeling. This model differs from the original spectral bundle work [74, 75] where* $k_c = 1$ *and* $k_p > 0$ *can vary and provide past spectral information to improve convergence. We also utilize and present the same model update steps as done in [46].*

2.2.2 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 (2.1)). Notice that the candidate iterate can be written as follows

$$\tilde{y}_{t+1} = \operatorname*{arg\,min}_{y \in \mathbb{R}^m} \sup_{(X,\nu) \in \widehat{\mathcal{X}}_t \times \mathrm{N}} F(X, \nu, y), \tag{2.19}$$

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

$$\tilde{y}_{t+1} = \operatorname*{arg\,min}_{y\in\mathbb{R}^m} \hat{f}_t(y) + \|y-y_t\|^2 = y_t - \frac{1}{\rho}(b-\nu_{t+1}-\mathcal{A}X_{t+1}), \tag{2.20}$$

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

$$(X_{t+1}, \nu_{t+1}) \in \operatorname*{arg\,max}_{(X,\nu)\in\widehat{\mathcal{X}}_t\times\mathrm{N}} \psi_t(X,\nu), \tag{2.21}$$

where

$$\psi_t(X,\nu) := \langle C, X\rangle + \langle b-\nu-\mathcal{A}X, y_t\rangle - \frac{1}{2\rho}\|b-\nu-\mathcal{A}X\|^2. \tag{2.22}$$

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:

$$\tilde{X} \leftarrow \operatorname*{arg\,max}_{X\in\widehat{\mathcal{X}}_t} \psi_t(X,\tilde{\nu}) \tag{2.23}$$

$$\tilde{\nu} \leftarrow \operatorname*{arg\,max}_{\nu\in\mathrm{N}} \psi_t(\tilde{X},\nu) = \mathrm{proj}_{\mathrm{N}}(b-\mathcal{A}\tilde{X}-\rho y_t) \tag{2.24}$$

Clearly ψ_t is smooth, and in this case the alternating maximization algorithm is known to converge at a $\mathcal{O}(1/\varepsilon)$ rate [22]. If ψ_t happens to also be strongly concave,

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`
  Update $\hat{f}_{t+1}$ and $\widehat{\mathcal{X}}_{t+1}$ using (2.12), (2.13), (2.17), (2.18).
return X_T, y_T

the rate improves to $\mathcal{O}(\log(1/\varepsilon))$ [107]. The function ψ_t is only guaranteed to be strongly concave if $\mathcal{A}$ is bijective.

We solve (2.23) using a primal-dual interior point method derived in Appendix A.1.2.

Remark 7. *For the sake of our theoretical convergence results, we advocate for solving for* $(X_{t+1}, \nu_{t+1}) \in \widehat{\mathcal{X}}_t \times \mathrm{N}$ *exactly. This differs from the other work which enables explicit inequality modeling [74]. Helmberg and Kiwiel [74] prove asymptotic results which do not need to solve this joint maximization problem exactly. We do need optimality in order to use the proximal bundle method results of Díaz and Grimmer [45]. We find that it is not a computational burden to solve this alternating maximization problem to full optimality. It is also easy to check for convergence by just checking the norm of the difference between* $\tilde{\nu}$ *as successive iterations.*

Feasibility of $\tilde{y}_{t+1}$. If we substitute $\nu_{t+1} = \mathrm{proj}_{\mathrm{N}}(b - \mathcal{A}X_{t+1} - \rho y_t)$ into (2.20), it can be seen that the candidate $\tilde{y}_{t+1}$ is always feasible

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

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

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

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 A.1.1. This interior point method has a per-iteration complexity of $\mathcal{O}(k^6)$ and converges at a rate $\mathcal{O}(\log(1/\varepsilon))$.

Remark 8. *Ding and Grimmer [46] claim that* (2.23) *could be solved using (accelerated) projected gradient descent and describe the method necessary for projection and proper scaling, but they use Mosek [9], 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* (2.23) *is difficult and varies between time steps t. We instead use (and advocate for) the primal-dual interior point method derived in Appendix A.1.2. This interior point method has a per-iteration complexity of $\mathcal{O}(k^6)$ and converges at a rate $\mathcal{O}(\log(1/\varepsilon))$.*

2.2.3 Convergence Analysis

In this section, we present the non-asymptotic convergence results for USBS. Under any setting of the algorithms parameters, Theorem 9 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. The results are summarized in the following theorem.

Theorem 9. *Suppose strong duality holds. For any fixed* $\rho > 0, \beta \in (0,1), k_c \geq 1, k_p \geq 0$, *and* $\alpha \geq 2\mathfrak{N}(\mathcal{X}_\star)$, *USBS produces iterates* $X_t \succeq 0$ *and* $y_t \in \mathcal{Y}$ *such that for any* $\varepsilon \in (0,1]$,

$$\begin{aligned}
&\textit{approximate penalized dual optimality: } f(y_t) - f(y_\star) \leq \varepsilon, \\
&\textit{approximate primal feasiblity: } \operatorname{dist}(\mathcal{A}X_t, \mathcal{K}) \leq \sqrt{\frac{2\rho}{\beta}}\varepsilon, \\
&\textit{approximate dual feasiblity: } \lambda_{\max}(C - \mathcal{A}^* y_t) \leq \frac{\varepsilon}{\mathfrak{N}(X_\star)}, \\
&\textit{approximate primal-dual optimality: } |\langle b, y_t\rangle - \langle C, X_t\rangle| \leq \frac{\alpha}{\mathfrak{N}(X_\star)}\varepsilon + \mathfrak{N}(y_0)\sqrt{\frac{2\rho}{\beta}}\varepsilon.,
\end{aligned}$$

for all iterations t

$$t \geq \mathcal{O}\left(\frac{\rho G^2 D^4}{\beta(1-\beta)^2\varepsilon^3}\right),$$

where $G \leq \|b\| + \alpha\|\mathcal{A}\|_\infty$. *And, if strict complementarity holds, then these conditions are achieved for all* t

$$t \geq \mathcal{O}\left(\frac{G^2}{\beta(1-\beta)^2 \min\{\mu/\rho, 1\}\varepsilon}\right).$$

Additionally, if Slater's condition holds and $y_\star$ *is unique, then the approximate primal feasibility and approximate primal-dual optimality respectively improve to*

$$dist(\mathcal{A}X_t, \mathcal{K}) \leq \rho\sqrt{2\mu}\,\varepsilon \quad \textit{and} \quad |\langle b, y_t\rangle - \langle C, X_t\rangle| \leq \left(\frac{\alpha}{\mathfrak{N}(X_\star)} + \mathfrak{N}(y_0)\rho\sqrt{2\mu}\right)\varepsilon.$$

Remark 10. *This theorem assumes perfect subproblem solving, namely accurately computing extreme eigenvectors, the penalized dual objective* $f(y)$, *the model* $\hat{f}(y)$, *and optimal values of* (2.21). *In practice, we cannot compute these quantities exactly and resort to highly accurate approximate methods. Anecdotally, the accuracy of the solutions to these subproblems is extremely important to the quality of the SDP solution. While we do not perform the analysis using approximate subproblem solutions,*

the error from each approximate step can be bounded. We leave this analysis for future work.

Note the following fact necessary to use Theorem 2.

Fact 11. *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 12. *Let* $y \in \mathcal{Y}$ *be arbitrary. Then,*

$$f(y) - f(y_\star) \geq \mu \operatorname{dist}^{2^d}(y, \mathcal{Y}_\star), \tag{2.25}$$

where $\mu > 0$ *is constant and* $d \in \{0, 1, \ldots, m\}$ *is the singularity degree* [49, 135, 138]*. 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 [49, 135, 138] ensures that

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

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

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

□

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 13. *At every descent step t,*

$$\operatorname{dist}(\mathcal{A}X_{t+1}, \mathcal{K}) \leq \sqrt{\frac{2\rho}{\beta}(f(y_t) - f(y_\star))}. \tag{2.26}$$

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

$$\operatorname{dist}(\mathcal{A}X_{t+1}, \mathcal{K}) \leq \rho\sqrt{2\mu}(f(y_t) - f(y_\star)). \tag{2.27}$$

Proof. For any descent step t, it is easy to verify from (2.20) and (2.24) 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 $\hat{f}_t(y) \le f(y)$ for all $y \in \mathcal{Y}$, and the definition of a descent step, which gives

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

Assume Slater's condition holds and $y_\star$ is unique. Then, using Lemma 12 gives

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

□

Lemma 14. *Suppose strong duality holds and $\alpha \ge 2\mathfrak{N}(\mathcal{X}_\star)$. Then, at every descent step t,*

$$\lambda_{\max}(C - \mathcal{A}^* y_t) \le \frac{1}{\mathfrak{N}(X_\star)}\left(f(y_t) - f(y_\star)\right). \tag{2.28}$$

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

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

We now utilize this bound to obtain the desired result

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

where the last inequality is implied by the assumption that $\alpha \geq 2\mathfrak{N}(\mathcal{X}_\star) \geq 2\|X_\star\|_*$.

□

Lemma 15. *At every descent step t,*

$$|\langle b, y_{t+1}\rangle - \langle C, X_{t+1}\rangle| \leq \frac{\alpha}{\mathfrak{N}(X_\star)}(f(y_t) - f(y_\star)) + \mathfrak{N}(y_0)\sqrt{\frac{2\rho}{\beta}\,(f(y_t) - f(y_\star))}. \quad (2.29)$$

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

$$|\langle b, y_{t+1}\rangle - \langle C, X_{t+1}\rangle| \leq \left(\frac{\alpha}{\mathfrak{N}(X_\star)} + \mathfrak{N}(y_0)\rho\sqrt{2\mu}\right)(f(y_t) - f(y_\star)). \quad (2.30)$$

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 13 it can be seen

$$\begin{aligned}|\langle \mathcal{A}X_{t+1} - b, y_{t+1}\rangle| &\leq \|\mathcal{A}X_{t+1} - b\|\|y_{t+1}\| \\ &\leq \|y_{t+1}\|\sqrt{\frac{2\rho}{\beta}\,(f(y_t) - f(y_\star))} \\ &\leq \mathfrak{N}(y_0)\sqrt{\frac{2\rho}{\beta}\,(f(y_t) - f(y_\star))}.\end{aligned}$$

Assume Slater's condition holds and $y_\star$ is unique. Then, using Lemma 12 gives

$$|\langle \mathcal{A}X_{t+1} - b, y_{t+1}\rangle| \leq \mathfrak{N}(y_0)\rho\sqrt{2\mu}\,(f(y_t) - f(y_\star))\,.$$

Since $X_{t+1} \succeq 0$ and $\mathrm{tr}(X_{t+1}) \leq \alpha$ by construction, we can use Lemma 14 to yield

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

The immediately yields the desired result. □

Using these lemmas and Theorem 2, we present the following proof of non-asymptotic convergence of USBS.

Proof of Theorem 9. The overall proof strategy is to use Theorem 2 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 13, 14, and 15 to obtain the stated convergence of primal feasibility, dual feasibility, and primal-dual optimality.

To apply the result of Theorem 2, we showed that the norms of the iterates and subgradients of f are bounded (Fact 11), and so all we need to do is show that the model satisfies the conditions (2.3), (2.4), and (2.5).

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

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

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

$$f(y) = \sup_{X \in \mathcal{X}} \langle C - \mathcal{A}^* y, X\rangle + \langle b, y\rangle \quad \text{and} \quad \hat{f}_{t+1}(y) = \sup_{X \in \widehat{\mathcal{X}}_{t+1}} \langle C - \mathcal{A}^* y, X\rangle + \langle b, y\rangle. \tag{2.31}$$

Verifying (2.3)**.** The condition (2.3) follows from (2.31), since $\widehat{\mathcal{X}}_{t+1} \subseteq \mathcal{X}$.

Verifying (2.4)**.** 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 $S = \alpha s s^\top$ implies $\alpha v v^\top \in \widehat{\mathcal{X}}_{t+1}$. Hence,

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

Verifying (2.5)**.** From the first-order optimality conditions and the update step (2.20) we know that

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

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 $y \in \mathcal{Y}$,

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

□

2.3 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* [62, 71, 97, 144] 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 feasbility of the primal problem), we only need to store $\langle C, \bar{X}_t\rangle$, $\operatorname{tr}(\bar{X}_t)$, and $\mathcal{A}\bar{X}_t$. This means that our representation of $\bar{X}_t$ is independent from the operations taken by USBS. In addition, these values can be efficiently maintained given low-rank updates to $\bar{X}_t$ due to the linearity of the operations,

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

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

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

2.3.1 Nyström Sketch

Consider any iterate $\bar{X}_t \in \mathbb{S}_+^n$. Let $r \in \mathbb{N}$ be a parameter that trades of accuracy for scalability. To construct the Nyström sketch, we sample a random projection matrix $\Omega \in \mathbb{R}^{n\times r}$ such that $\Omega_{ij} \sim \mathcal{N}(0,1)$ are sampled i.i.d. The projection of $\bar{X}_t$, or *sketch*, can be computed as follows

$$P_t = \bar{X}_t \Omega \in \mathbb{R}^{n\times r}. \tag{2.32}$$

We can efficiently maintain P_t as $\bar{X}_t$ evolves with low-rank updates,

$$P_{t+1} \leftarrow \Big(\eta_{t+1}\bar{X}_t + V_t Q_{\underline{c}} \Lambda_{\underline{c}} Q_{\underline{c}}^\top V_t^\top\Big)\Omega = \eta_{t+1}P_t + V_t Q_{\underline{c}} \Lambda_{\underline{c}}\Big(Q_{\underline{c}}^\top V_t^\top \Omega\Big). \tag{2.33}$$

Note that computing P_{t+1} in this manner is equivalent to applying the projection matrix to X_{t+1}. Given the sketch P_t and the projection matrix Ω, we can compute a rank-r approximation to $\bar{X}_t$. The approximation is as follows

$$\widehat{\bar{X}_t} := P_t(\Omega^\top P_t)^+ P_t^\top = (\bar{X}_t\Omega)(\Omega^\top \bar{X}_t \Omega)^+ (\bar{X}_t\Omega)^\top, \tag{2.34}$$

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. [164]. The Nyström approximation yields a provably good approximation for any sketched matrix. See the following fact from [144],

Fact 16. *Fix $X \in \mathbb{S}^n_+$ arbitrarily. Let $P := X\Omega$ where $\Omega \in \mathbb{R}^{n\times r}$ has independently sampled standard normal entries. For each $r' < r-1$, the Nyström approximation $\widehat{X}$ as defined in* (2.34) *satisfies*

$$\mathbb{E}_\Omega \left\| X - \widehat{X} \right\|_* \leq \left(1 + \frac{r'}{r - r' - 1}\right) \left\| X - [\![X]\!]_r \right\|_*, \tag{2.35}$$

where $\mathbb{E}_\Omega$ is the expectation with respect to Ω and $[\![X]\!]$ 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 [115]*. If we replace $\widehat{X}$ with $[\![\widehat{X}]\!]_r$, this error bound still remains valid.*

This means that by applying this Nyström sketching technique the only approximation error is in the approximation $\widehat{\bar{X}_t}$, and all of the error is constant (i.e. the error does not compound over time) and comes only from the projection matrix (and the choice of r).

2.3.2 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 Ω 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.

2.4 Discussion of Algorithmic Contributions

USBS extends previous work with several theoretical and practical contributions. The main high-level contribution is to extend the algorithm presented by Ding and Grimmer [46] to handle inequality constraints. We utilize some of the insight provided by Helmberg and Kiwiel [74] to incorporate inequality constraints into the penalized dual objective function via the dual of the indicator function for the feasibility of y. Helmberg and Kiwiel [74] differs from this work since they use a much simpler model and they do not solve (2.22) exactly, which we need to be exact (high-precision) for our proofs of non-asymptotic convergence rates. Lastly, we derive two novel primal-dual interior point methods for solving (2.23) and $\hat{f}_t(\tilde{y}_{t+1})$, respectively. We found that it is empirically important to solve the subproblems to high precision and that these primal-dual interior point methods are the most reliable way to solve these subproblems. Having an efficient standalone implementation with these algorithmic innovations (including sketching) all contribute to the speed and scalability of USBS in practice. Moreover, a standalone implementation is the only way to utilize a GPU to improve empirical performance. For more details on implementation and experimental details, see the implementation in pure JAX and the next chapter.

CHAPTER 3

USBS FOR CLASSIC COMBINATORIAL OPTIMIZATION PROBLEMS

We first evaluate USBS on two different classic SDP problem types, MaxCut and Quadratic Assignment Problems, with and without warm-starting strategies against the scalable semidefinite programming algorithm CGAL [163, 164] (with sketching). Yurtsever et al. [164] perform an extensive evaluation against strong SDP solvers, including SeDuMi [137], SDPT3 [143], Mosek [9], SDPNAL+[160], and the BM factorization heuristic [31]. They show on several applications that CGAL with sketching is by far the standard against which to compare for scalable semidefinite programming.

This work was originally published as part of the following publication:

> **Fast, Scalable, Warm-Start Semidefinite Programming with Spectral Bundling and Sketching**
> Rico Angell and Andrew McCallum
> ICML 2024

3.1 General Experimental Setup and Details

Following Yurtsever et al. [164], we scale the problem data for all problems such that the following condition holds,

$$\|C\|_{\mathrm{F}} = \mathrm{tr}(X_\star) = 1. \tag{3.1}$$

In both 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 (3.1), we can set $\alpha = 2$ for all problem types.

We consider the primal iterate an ε-approximate solution if the relative primal objective suboptimality and relative infeasibility is less than ε, i.e.

$$\frac{\langle C, X\rangle - \langle C, X_\star\rangle}{1 + \langle C, X\rangle} \leq \varepsilon \quad \text{and} \quad \frac{\operatorname{dist}(\mathcal{A}X, \mathcal{K})}{1 + \|b\|} \leq \varepsilon. \tag{3.2}$$

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 $y \in \mathcal{Y}$,

$$\begin{aligned} f(y) &= \alpha \max\{\lambda_{\max}(C - \mathcal{A}^* y), 0\} + \langle b, y\rangle \geq \langle C - \mathcal{A}^* y, X_\star\rangle - \langle b, y\rangle = \langle C, X_\star\rangle, \\ &\implies \langle C, X\rangle - \langle C, X_\star\rangle \leq \langle C, X\rangle - f(y). \end{aligned} \tag{3.3}$$

We use this upper bound to approximate the relative primal objective suboptimality in our experiments for USBS. We use the upper bound given in [164] 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 [77, 155] 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 $\uparrow$ to indicate that *higher is better* for that particular metric and $\downarrow$ to indicate *lower is better* for that particular metric.

3.2 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

$$\max \quad \frac{1}{4} x^\top L x \quad \text{s.t.} \quad x \in \{\pm 1\}^n, \tag{3.4}$$

where L is the graph Laplacian. This optimization problem is known to be NP-hard [85], but rounding the solution to the SDP relaxation can yield very strong approximate solutions [64].

3.2.1 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

$$\max \quad \frac{1}{4} \operatorname{tr}(LX) \quad \text{s.t.} \quad \operatorname{diag}(X) = \mathbf{1} \text{ and } X \succeq 0, \tag{3.5}$$

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 (3.5) 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. [64]), but this one works quite well in practice.

Notice that (3.5) 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.

3.2.2 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 [164], we evaluate the size of r cuts given by the column vectors of $\mathrm{sgn}(U)$ and choose the largest.

3.2.3 Experimental Setup

We evaluate the CGAL and USBS with and without warm-starting on ten instances from the DIMACS10 [13] where n and the number of edges in each graph are shown in Table 3.1. 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.

3.2.4 Experimental Results

Figure 3.1 displays the time (in seconds) taken by CGAL and USBS, with and without warm-starting on 99% of the problem instance, to achieve an ε-approximate solution where $\varepsilon = 10^{-1}$. In all cases, we optimize until an ε-approximate solution is found or 72 hours has elapsed (the same stopping criteria is used for creating a warm-start initialization). The bars marked with † indicate that an ε-approximate solution was not achieved in 72 hours. As can be seen, USBS without warm-starting converges to an ε-approximate solution faster the CGAL with and without warm-starting. On 7 out of 10 data instances, CGAL fails to achieve an ε-approximate solution in 72 hours while USBS reaches an ε-approximate solution for all data instances within 72 hours.

On 1 out of 10 data instances, USBS fails to leverage the warm-start initialization improve convergence time to acheive an ε-approximate solution (problem instance 598a). In all other cases, warm-starting either helps or at least does not hurt the convergence time. On a moderate size problem instance, fe_body ($> 10^9$ decision variables), USBS is able to achieve an ε-approximate solution in 5 minutes while CGAL takes 45 hours to achieve the same ε-approximate solution, yielding over a 500x speedup. On a massive problem instance, 333SP ($> 10^{13}$ decision variables), USBS is able to achieve an ε-approximate solution in 28 hours.

Figure 3.2 plots three different convergence measures over time for a large data instance, 144 ($>10^{10}$ decision variables), 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 does not drop in value when warm-starting whereas the weight of the cut produced by CGAL drops in value before rising again. 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.

Table 3.1: 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
nnz(L)	115K	149K	372K	405K	983K	1.6M	2.3M	7.1M	7.1M	25.9M

Figures 3.3 and 3.4 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. As a general trend, we find that the infeasibility gap of the solution produced by USBS improves when k_c is larger. If choosing between allocating resources between k_c and

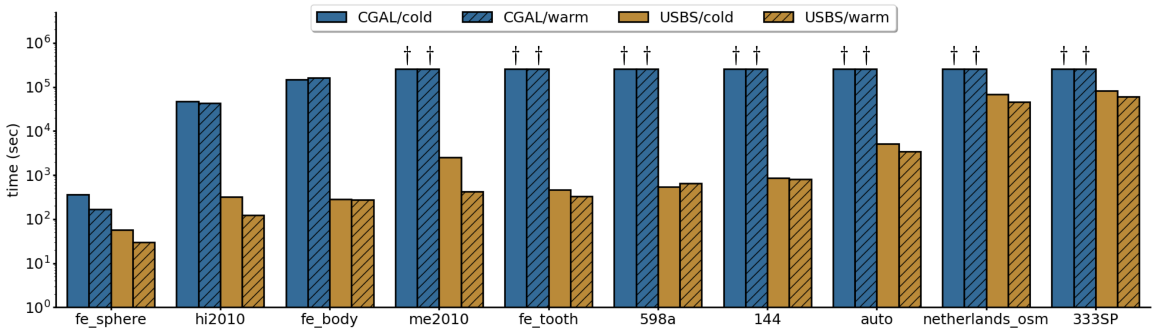


Figure 3.1: Convergence time (sec) to moderate relative error tolerance (↓). The time (in seconds) for CGAL and USBS to achieve an ε-approximate solution for $\varepsilon = 10^{-1}$ with and without warm-starting on 99% of initial data on ten `DIMACS10` MaxCut instances. The bars marked with † indicate 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.

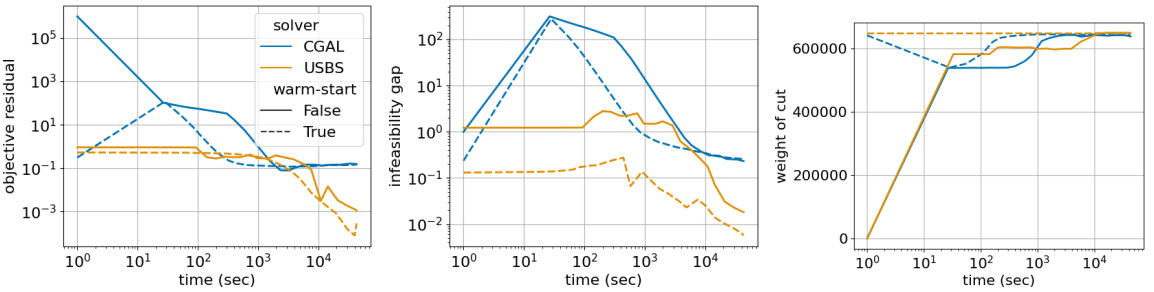
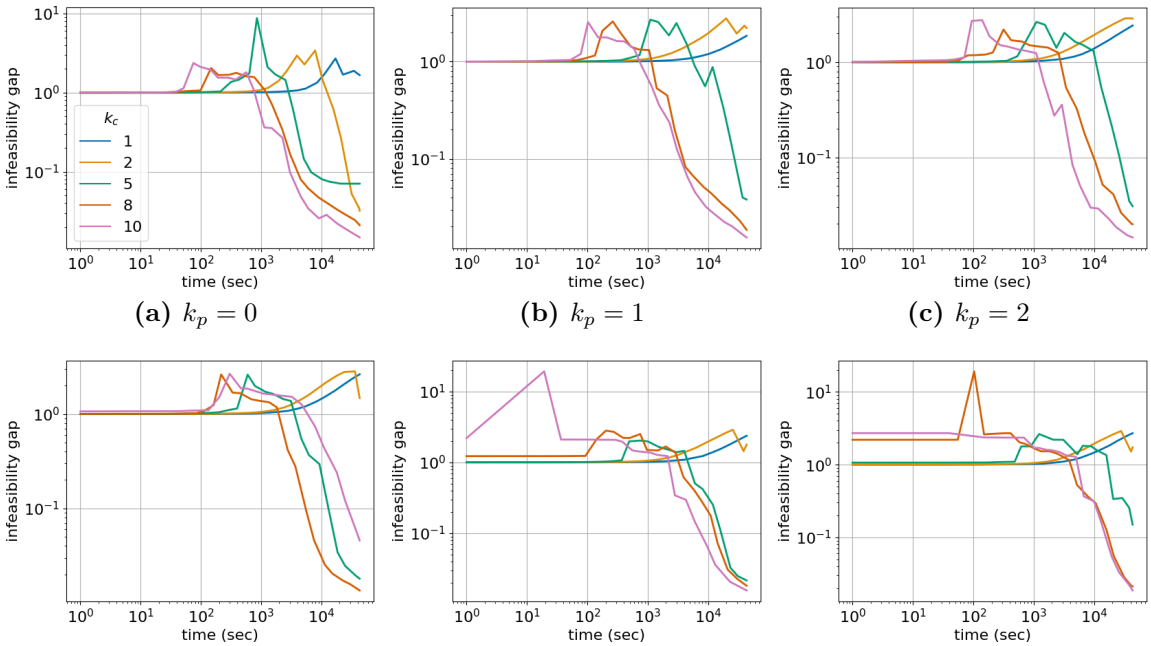


Figure 3.2: Convergence measures on instance `144`. We solve instance `144` from DIMACS10 and plot the primal objective sub-optimality (objective residual, ↓), relative infeasibility (infeasibility gap, ↓), and weight of the cut (↑) produced by the rounding procedure. The warm-started runs use 99% of 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.

k_p we find more resources should be given to k_c. The infeasibility gap of the solution produced by USBS is also generally better when $k_c > k_p$.

Figures 3.5 and 3.6 compare USBS's objective residual against time for different settings of k_c and k_p on the `144` instance from `DIMACS10` on both CPU and GPU. As a general trend, we find that the objective residual of the solution produced by USBS



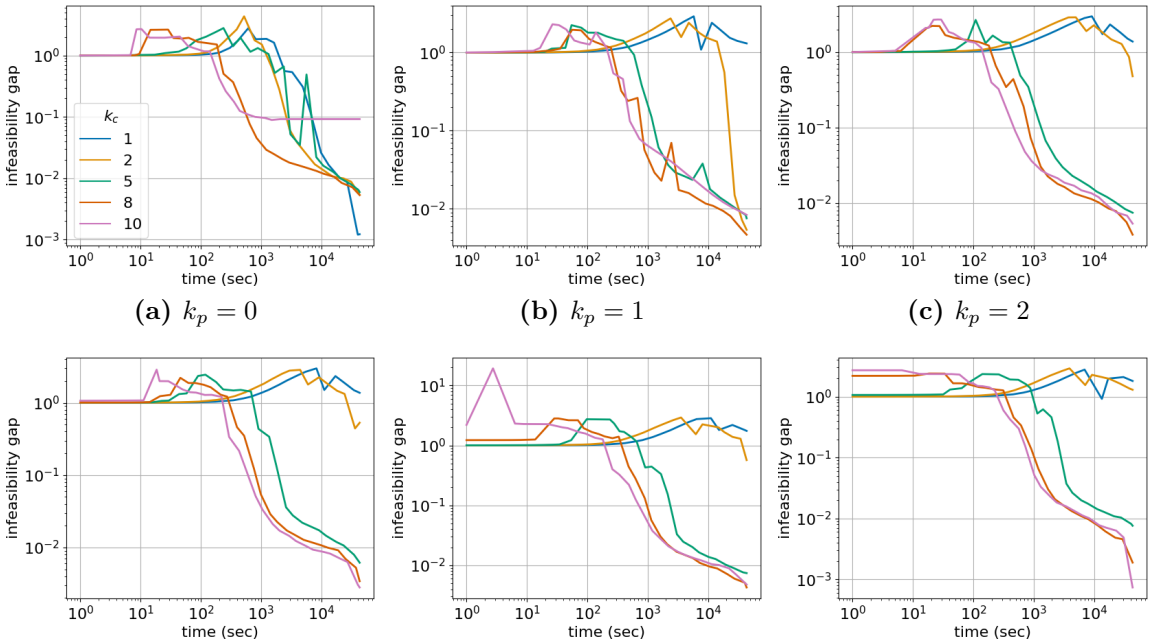
(a) $k_p = 0$ **(b)** $k_p = 1$ **(c)** $k_p = 2$

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

Figure 3.3: 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 $k_c \geq k_p$.

improves when k_c is larger. The value of k_p seems to have less of an impact on the objective residual of the solution produced by USBS than with the infeasibility gap. These results might vary with different settings of ρ.

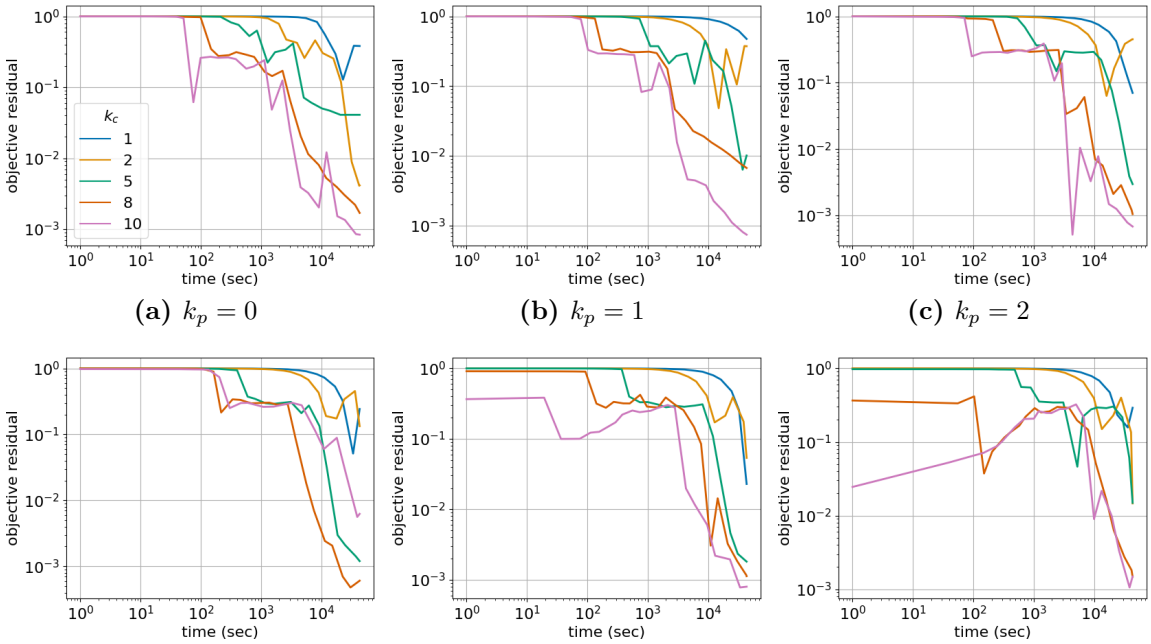
Figures 3.7 and 3.8 compare CGAL and USBS on CPU and GPU for different settings of k_c and k_p on the `144` instance from `DIMACS10`. We observe approximately a 25x speedup on a single NVIDIA GeForce 1080 Ti GPU as compared to a CPU instance with 16 cores and 128GB of RAM for both CGAL and USBS. We also observe that USBS performs best when $k_c \geq k_p$. In addition, USBS executed on CPU significantly outperforms CGAL executed on GPU.



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

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

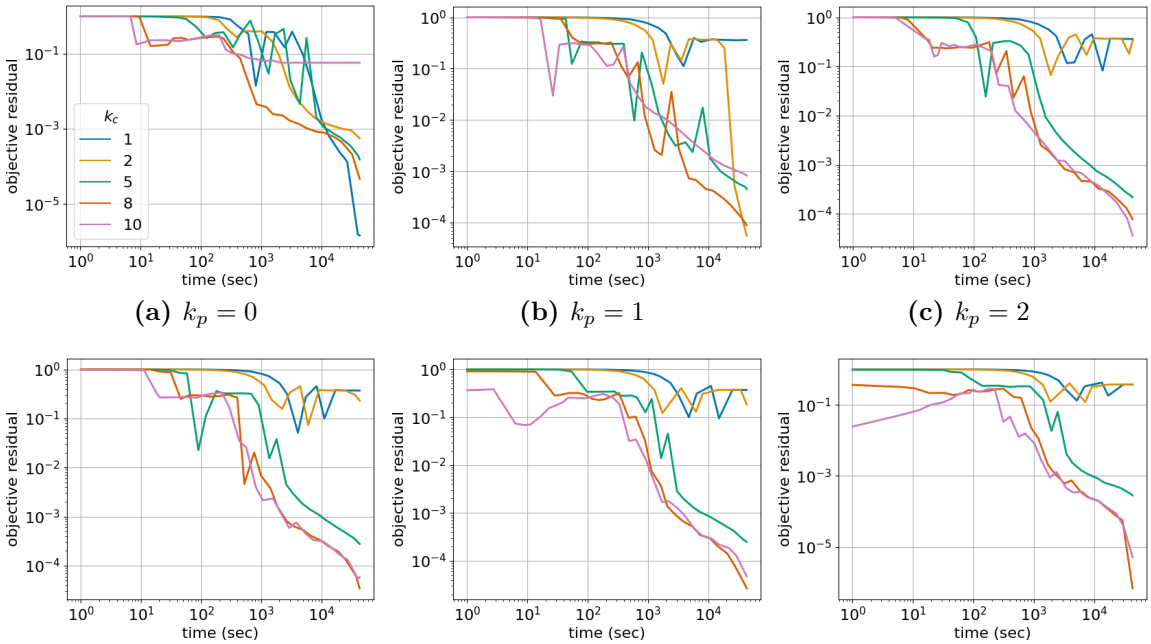
Figure 3.4: Infeasibility gap vs. time for different settings of k_c and k_p. In each plot, the x-axis is time (up to 12 hours) and the y-axis is the relative infeasibility gap. In every case, USBS is cold-started on the `144` instance from the `DIMACS10` dataset. Each plot considers one value of k_p 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)** $k_p = 10$

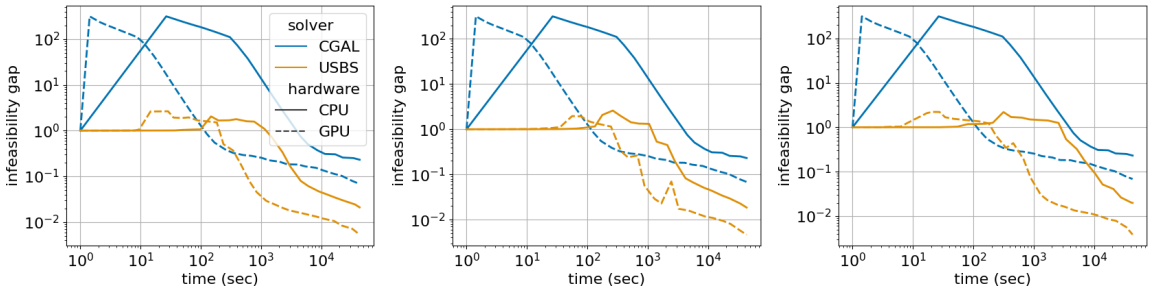
Figure 3.5: Objective residual vs. time for different settings of k_c and k_p. In each plot, the x-axis is time (up to 12 hours) and the y-axis is the primal objective suboptimality. In every case, USBS is cold-started on the `144` instance from the `DIMACS10` dataset. Each plot considers one value of 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 $k_c \geq k_p$.



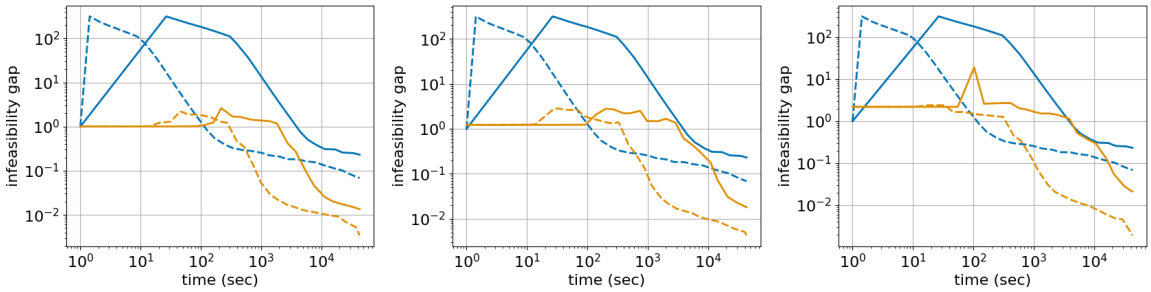
(a) $k_p = 0$ **(b)** $k_p = 1$ **(c)** $k_p = 2$

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

Figure 3.6: Objective residual 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 $k_c \geq k_p$.

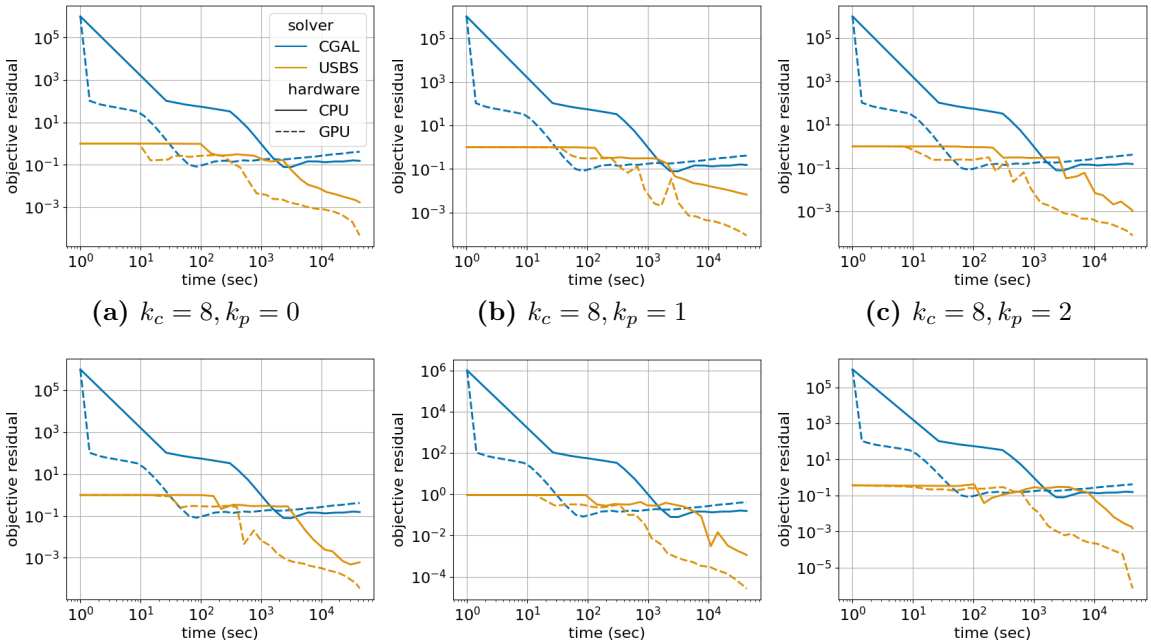


(a) $k_c = 8, k_p = 0$ **(b)** $k_c = 8, k_p = 1$ **(c)** $k_c = 8, k_p = 2$



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

Figure 3.7: 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 3.8: 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 $k_c \geq k_p$.

3.3 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 [103]. SDP relaxations have been shown to facilitate finding good solutions to large QAPs [166]. The QAP can be formally defined as follows

$$\min \quad \operatorname{tr}\left(W\Pi D\Pi^\top\right) \quad \text{s.t.} \quad \Pi \text{ is an } \mathtt{n}\times\mathtt{n} \text{ permutation matrix}, \tag{3.6}$$

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.

3.3.1 SDP Relaxation

There are many SDP relaxtions for QAPs, but we consider the one presented by [164] and inspired by [30, 79] which is formulated as follows

$$\begin{aligned}
\min \quad & \operatorname{tr}\big((D\otimes W)\,\mathsf{Y}\big) \\
\text{s.t.} \quad & \operatorname{tr}_1(\mathsf{Y}) = I,\ \operatorname{tr}_2(\mathsf{Y}) = I,\ \mathcal{G}(\mathsf{Y}) \geq 0, \\
& \operatorname{vec}(\mathsf{B}) = \operatorname{diag}(\mathsf{Y}),\ \mathsf{B}\mathbf{1} = \mathbf{1},\ \mathbf{1}^\top\mathsf{B} = \mathbf{1}^\top,\ \mathsf{B} \geq 0, \\
& X := \begin{bmatrix} 1 & \operatorname{vec}(\mathsf{B})^\top \\ \operatorname{vec}(\mathsf{B}) & \mathsf{Y} \end{bmatrix} \succeq 0,\ \operatorname{tr}(\mathsf{Y}) = \mathtt{n}
\end{aligned} \tag{3.7}$$

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 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).

3.3.2 Rounding

We adopt the rounding method presented in [164] to convert an approximate solution of (3.7) 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) [83, 93, 117]. This procedure yields a feasible point to the original QAP (3.6). To get an upper bound for (3.6), we perform the rounding procedure on each column of U and choose the permutation matrix Π which minimizes the objective $\operatorname{tr}\big(W\Pi D\Pi^\top\big)$.

3.3.3 Experimental Setup

We evaluate CGAL and USBS on select large instances from `QAPLIB` [32] and `TSPLIB` [128] 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 (3.7). 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 [164] is shown to obtain significantly smaller `best relative gap` than CSDP [30] and PATH [165] 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 $\mathtt{n} - 1$ 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.

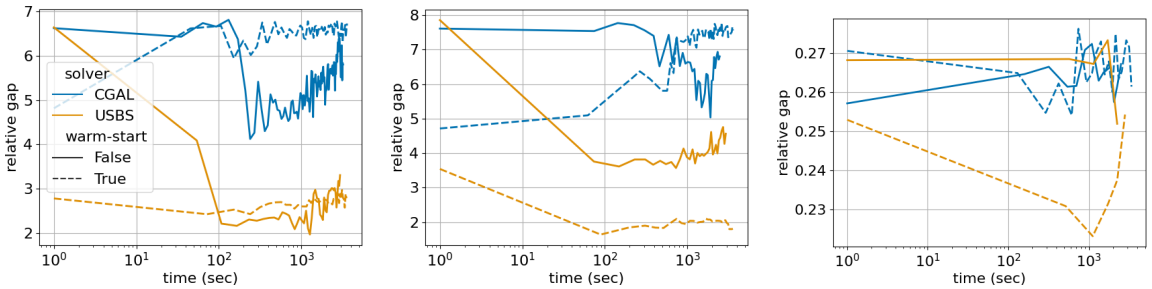
Following [164], we apply the following scaling of the problem variables in (3.7)

$$\|C\|_{\mathrm{F}} = \mathrm{tr}(X_\star) = \|\mathcal{A}\|_{\mathrm{op}} = 1 \quad \text{and} \quad \|A_1\|_{\mathrm{F}} = \cdots = \|A_m\|_{\mathrm{F}}. \tag{3.8}$$

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.

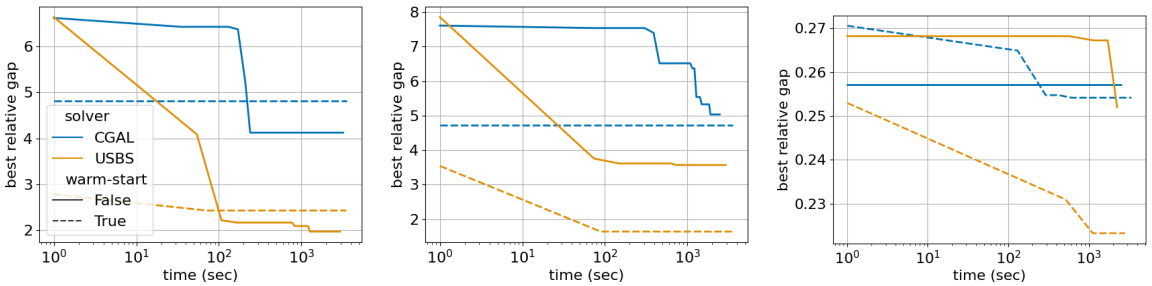
3.3.4 Experimental Results

Figure 3.9 and Figure 3.10 plot `relative gap` and `best relative gap` against time (in seconds) for three instances, respectively. Figure 3.9 shows that even for



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

Figure 3.9: `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 3.10: `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.

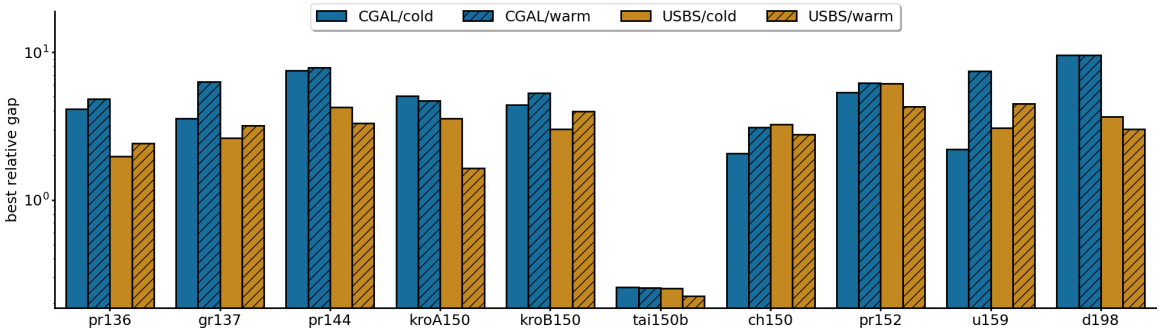


Figure 3.11: `best relative gap` (↓). The `best relative gap` obtained in one hour of optimization is shown for ten instances from `QAPLIB` and `TSPLIB`. We observe that on most problem instance that USBS produces a better relative gap than CGAL and that a warm-start initialization helps USBS obtain a better relative gap. CGAL is much less reliable in leveraging a warm-start initialization.

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 3.11 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 (3.7) 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 (3.7) from each of those warm-start initializations.

CHAPTER 4

USBS FOR INTERACTIVE ENTITY RESOLUTION WITH ∃-CONSTRAINTS

Real-world deployed machine learning systems will inevitably be imperfect despite ever growing labeled training data and model sizes. Machine learning models will encounter novel inputs and patterns and noise unseen at training which will lead to incorrect predictions. This lack of precision is amplified in applications where training data is limited or the objective is misaligned with the task. Fortunately, human users interacting with machine learning systems over time have the ability to provide feedback to correct the model outputs and thus improve the precision and safety of the system.

Techniques and approaches for gathering, representing, and utilizing human feedback have been developed for tasks in both supervised and unsupervised machine learning. For classification, human-in-the-loop active labeling techniques have been employed to gather labels for problematic inputs and fine-tune the model [132]. Weak supervision enables noisy label gathering by allowing users to write labeling rules for data where labeled data is scarce [127]. Integrating human feedback about relevant features instead of input example labels into classification models has been shown useful [33, 41, 48, 59, 100]. Human preferences about agent actions in complicated environments have been integrated into reinforcement learning systems to improve policy learning [34, 70]. Tasks with difficult to define objectives, such as learning to summarize text [136], have been shown to benefit from human feedback. Interactive models have been used for applications such as knowledge base construction [91] and

entity resolution [90]. Human user feedback has also been used to correct both flat and hierarchical clusterings [16, 25, 43, 113, 118, 149].

Clustering is an area of machine learning that could particularly benefit from human feedback. Since clustering is often unsupervised and used for discovery of entities, patterns, and groups, it generally needs additional input or side information to guide predictions. In past work on clustering with human feedback, users interact with the predicted clustering and provide feedback in the form of pairwise point constraints [151], relative comparisons among data points [55], and cluster sizes [158]. Existing approaches either directly integrate this feedback into existing clustering frameworks [88, 133, 152] or adapt the similarity measure of the data to produce a more accurate predicted clustering [94, 98, 99, 106, 156].

In this chapter, we 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 [1, 2, 6, 26, 159]. 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 [14]: active clustering (also know as semi-supervised

clustering) [113, 130, 149, 150, 157] and interactive clustering [21, 35, 36, 50][1]. There are several paradigms of human feedback used to steer a clustering algorithm [55, 88, 94, 98, 99, 106, 133, 152, 156], the most common of which is pairwise constraints [151] (i.e. statements about whether two points or mentions must or cannot be clustered together).

In this chapter, we introduce *existential cluster constraints* (i.e. $\exists$-*constraints*), a new paradigm of interactive feedback for correcting entity resolution predictions and present a novel SDP relaxation as part of a heuristic algorithm for satisfying $\exists$-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. More specifically, human users view the predicted clustering and make statements about the existence of a cluster containing a specific subset of discrete features and not containing another disjoint subset of discrete features. We posit this enables users to convey cluster-wise preferences more naturally with respect to a user's mental model of the output clustering. Furthermore, existential cluster constraints are more efficient in terms of the number of human feedback inputs to reach an ideal output clustering both in theory and practice. Lastly, unlike prior forms of feedback, existential cluster constraints allow users to provide feedback when data privacy is a concern without viewing the raw data or cluster sizes.

We show theoretically that existential cluster constraints are able to express feedback with far fewer human inputs than pairwise point constraints. Unsurprisingly, we also show that it is NP-hard in general to find a clustering which satisfies all of the existential cluster constraints.

---

[1]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.

We introduce an inference algorithm for clustering with existential cluster constraints within the correlation clustering framework. The algorithm begins by solving a semidefinite programming (SDP) relaxation of an integer linear program which jointly clusters the data points and existential cluster constraints by combining the traditional correlation clustering optimization problem with extra optimization constraints formed from the existential cluster constraints. Next, we construct a DAG-shaped hierarchical clustering (i.e., sparse cluster trellis [108]) over the data points and existential cluster constraints using the solution of SDP relaxation as an affinity matrix. Finally, we extract the optimal flat clustering consistent with the trellis constructed in the previous step using a dynamic programming algorithm which memoizes optimal partial clusterings.

We evaluate the efficiency of existential cluster constraints in terms of human feedback inputs and the efficacy of our inference algorithm on publicly available author coreference datasets. Author coreference is a real world clustering task where the predictions are viewed by human users on websites such as Google Scholar[2], Semantic Scholar[3], and OpenReview[4]. We simulate the process of human feedback in a series of rounds in which an oracle generates one or more constraints to correct the previous round's predictions and run our inference algorithm on the running set of constraints. We compare existential cluster constraint directly against must-link and cannot-link constraints. On every dataset, we find existential cluster constraints require nearly half as many user feedback inputs than must-link and cannot-link constraints to reach the ground truth clustering.

[2]scholar.google.com

[3]semanticscholar.org

[4]openreview.net

This work was originally published as part of the following publications:

Fast, Scalable, Warm-Start Semidefinite Programming with Spectral Bundling and Sketching
Rico Angell and Andrew McCallum
ICML 2024

Interactive Correlation Clustering with Existential Cluster Constraints
Rico Angell, Nicholas Monath, Nishant Yadav, and Andrew McCallum
ICML 2022

4.1 Problem Setup

In this section, we will describe the problem setting that will be the foundation from which we will detail our contributions. We will define the correlation clustering objective used throughout the remainder of the paper, the domain of the vertices and cluster features used to express existential cluster constraints, and our assumptions about a latent ground truth clustering.

4.1.1 Correlation Clustering

Correlation clustering is a graph-based clustering framework that has several advantages over other clustering objectives [17, 44, 140]. For example, correlation clustering does not require the user to specify the number of clusters or a threshold for choosing a particular flat clustering. There are several correlation clustering objectives with different properties, but we will use a version of the max-agree correlation clustering objective.

In our setting, we are given a graph $G = (V, E)$ of n vertices where each edge $e = (u, v)$ has a weight $w_e \in \mathbb{R}$ (sometimes written as w_{uv}) representing the affinity between u and 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 $\bigcup_{C \in \mathcal{C}} C = V$. The goal of the correlation clustering problem [17] is to find a clustering of the vertices

which maximizes the sum of positive edge-weights within each cluster minus the negative edge-weights across the clusters. Let E^+ and E^- be the subsets of edges with positive and negative edge weights, respectively. Formally, the max-agree correlation clustering optimization problem can be written as follows:

$$\begin{aligned} \max \sum_{(u,v)\in E^+} w_{uv}\, x_u \cdot x_v - \sum_{(u,v)\in E^-} w_{uv}(1 - x_u \cdot x_v) \\ \text{s.t.} \quad x_v \in \{e_1, e_2, \ldots, e_n\}, \quad v \in V \end{aligned} \tag{4.1}$$

where $e_i \in \mathbb{R}^n$ is the ith standard basis vector. A complete assignment to the decision variables x_v defines a clustering of the vertices: $x_v = e_i$ implies vertex v is assigned to cluster i. Conversely, any clustering of the vertices can be expressed in an assignment to the decision variables. Additionally, it is well known that Problem (4.1) is intractable to optimize exactly, unless P = NP [17].

Alternatively, consider the following optimization problem:

$$\begin{aligned} \max \sum_{(u,v)\in E} w_{uv}\, x_u \cdot x_v \\ \text{s.t.} \quad x_v \in \{e_1, e_2, \ldots, e_n\}, \quad v \in V \end{aligned} \tag{4.2}$$

This objective considers only the *intra*-cluster edges of a clustering regardless of the sign of the weights. By only considering intra-cluster edges, the partial objective computed over a subset of vertices $S \subset V$ is independent from the choice of partitioning the remaining set of vertices $V \setminus S$. This property makes Problem (4.2) more favorable than Problem (4.1) for optimization.

In general, clustering objectives can be viewed as a way to rank candidate solutions. Interestingly, Problem (4.1) and Problem (4.2) are *order-equivalent* (See the supplementary material of Greenberg et al. [69] for the proof). In other words, these objectives generate the same complete ranking of candidate clusterings. We will use

Problem (4.2) as our correlation clustering objective since it has desirable properties for optimization which we will utilize in Section 4.3.2.

4.1.2 Vertex and Cluster Features

In order to specify clustering constraints in terms of the features of desired clusters, we need to define the features of vertices and clusters we will consider in our setting. We assume that each of the n vertices has an associated set of discrete features. 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 $v \in V$. Furthermore, for any subset of vertices $S \subseteq V$, we define the set of features associated with S as $\Phi(S) = \bigcup_{v \in S} \Phi(v)$. We make no assumptions about the relationship between features and edge weights.

4.1.3 Ground Truth Clustering

We assume that there exists a latent ground truth clustering $\mathcal{C}^*$ which may be different from the clustering that optimizes the correlation clustering objective. Let k be the number of ground truth clusters. We refer to this problem of achieving the latent ground truth clustering through interactive feedback as the *interactive correlation clustering problem with discrete features.*

4.2 Existential Cluster Constraints

Most commonly, feedback in clustering is provided in the form of must-link and cannot-link constraints where users specify whether pairs of points must or cannot be in the same cluster, respectively. We can easily integrate these pairwise point constraints into the correlation clustering framework by re-assigning the edge weight for each must-link and cannot-link constraint to ∞ and $-\infty$, respectively. Despite being easy to incorporate into the correlation clustering framework, pairwise point constraints have several practical downsides: (1) they can be inefficient in expressing large sweeping changes to whole clusters; (2) it might be unnatural for users to express

desired attributes of output clusters when expressing feedback at the level of pairs of points; (3) we may not want users to interact with the raw data points directly.

In response, we propose ∃-*constraints* (pronounced *"there-exists constraints"*), a novel form of feedback which enables users to express statements about the existence of a cluster with and without particular cluster features. Formally, a ∃-constraint $\xi \subseteq \{+,-\} \times \Phi$ uniquely characterizes a constraint asserting there exists a cluster $C \in \mathcal{C}^*$ such that all of the positive features $\xi^+ := \{\phi \,|\, (+,\phi) \in \xi\}$ are contained in the features of C (i.e. $\xi^+ \subseteq \Phi(C)$) and none of the negative features $\xi^- := \{\phi \,|\, (-,\phi) \in \xi\}$ 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 $\xi^- \cap \Phi(S) = \emptyset$. Observe that it is perfectly feasible for more than one ground truth cluster to 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 be S is compatible with ξ just as long as $\Phi(S) \cap \xi^-$ is empty.

Example Use Case of ∃-constraints. Consider the problem of author coreference where there is a database of papers, grants, and other professional information about authors of scientific papers and we want to decide which of these belong to the same individual. This problem can be framed as a clustering problem where the output is some aggregation of the information (features) into clusters. Now consider the following example of how ∃-constraints could be used to correct errors made by a

system to solve this problem. Suppose there are two different authors "Alice Smith" and "Alice R. Smith" where "Alice Smith" has a co-author "Bob Johnson" on multiple papers and "Alice R. Smith" never co-authored a paper with anyone named "Bob Johnson". Now suppose the current output of the clustering algorithm predicts that the records of "Alice Smith" and "Alice R. Smith" belong to the same individual. To correct the clustering error, a human could provide feedback in the form of a ∃-constraint as follows:

$$\begin{aligned}\xi = \{&(+, \texttt{first\_name} : \text{“Alice”}),\\ &(+, \texttt{middle\_initial} : \text{“R”}),\\ &(+, \texttt{last\_name} : \text{“Smith”}),\\ &(-, \texttt{coauthor\_name} : \text{“Bob Johnson”})\}.\end{aligned}$$

Incorporating this ∃-constraint will cause the cluster to be split such that "Alice R. Smith" and "Alice Smith" are no longer in the same cluster (there might still be other errors which need to be fixed, but these can be fixed with additional feedback from the user). Note how this could be much more efficient in terms of human feedback as compared to must-link and cannot-link constraints. If "Alice Smith" wrote many papers with "Bob Johnson", the user might have to provide many more cannot-link constraints in order to split "Alice Smith" and "Alice R. Smith".

4.2.1 Efficiency of Feedback

The purpose of interaction in clustering is to enable users to correct mistakes in the predicted clustering by providing guidance to the algorithm through feedback. When providing feedback, we assume users are able to view the cluster features of predicted clusters and then provide one or more ∃-constraints. To ease the burden on the user, the feedback mechanism should ideally be both natural for the user to specify given their mental model of the latent ground truth clustering as well as

efficient in terms of the number of user feedback inputs they need to provide. We posit that ∃-constraints are both more efficient and natural to specify than previous feedback mechanisms.

Intuitively, specifying an existential cluster constraint is natural for a user. When viewing the features of a predicted cluster, a user has a mental model of the corresponding latent ground truth cluster in terms of the salient features that both belong and do not belong in the ground truth cluster. By better aligning the feedback mechanism with the user's mental model, we ease the burden on the user, and thus, make the process of providing feedback more efficient.

Existential cluster constraints are also more efficient along another dimension, namely, the number of human feedback inputs. Each existential cluster constraint is a compressed representation of a powerful logical statement about the output clustering. Theoretically, we provide a worst-case lower bound on how much more powerful ∃-constraints can be when compared to must-link and cannot-link constraints in our setting. Note the following proposition.

Proposition 17. *The interactive correlation clustering problem with discrete features will require $\Omega(k^2)$ more pairwise node constraints than total features required by ∃-constraints to achieve the latent ground truth clustering in the worst case, where k is number of clusters.*

Proof. Consider a complete graph with k ground truth clusters each with ℓ nodes where v_{ij} denotes the jth node in the ith cluster. Let the total number of features be $k + \ell$ and the features for each node v_{ij} be $\Phi(v_{ij}) = \{\phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_k, \phi_{k+j}\}$. Let the edge labels and weights be as follows:

$$\begin{cases} k^2\ell^2 & (v_{ij}, v_{i'j}) \\ 1 & (v_{ij}, v_{ij'}) \\ -(1+\varepsilon) & \text{o.w.} \end{cases}$$

where $\varepsilon > 0$. The optimal clustering with respect to the correlation clustering objective is ℓ clusters, each of the form $\{v_{1,j}, v_{2,j}, \ldots, v_{k,j}\}$. The cost of this clustering is $k\ell^2 - k\ell$, the sum of all of the ground truth within cluster edges, all of which are cut in the optimal predicted clustering. With clear reasoning, we can see why this is the optimal clustering. We never want to cut edges of the form $(v_{ij}, v_{i'j})$, since cutting one of these edges will incur a cost of $k^2\ell^2$, more than the sum of the weights of all other edges. Furthermore, we never want to add any edges of the form $(v_{ij}, v_{ij'})$ to the predicted clustering since doing so will incur a cost of at least 2ε for every one of these edges that we include.

We will now show that this construction requires at least $k^2 + k\ell - 2k$ pairwise node constraints to achieve the ground truth clustering. To accomplish this, we will show that to achieve the ground truth clustering using pairwise node constraints for each pair of ground truth clusters either we need to put cannot-link constraints on all the edges of the form $(v_{ij}, v_{i'j})$ or the pairwise node constraints need to fully specify the ground truth partitioning of that pair of ground truth clusters (i.e. there is exactly one partitioning which satisfies all of the pairwise node constraints and it is the ground truth partitioning).

For any arbitrary pair of ground truth clusters, assume the pairwise node constraints do not fully specify the ground truth partitioning and the optimal partitioning given the pairwise node constraints is the ground truth partitioning. For the sake of contradiction, suppose that the set of pairwise node constraints does not include cannot-link constraints for each edge of the form $(v_{ij}, v_{i'j})$. This means that there is a partitioning which satisfies all of the constraints which includes an edge of the form $(v_{ij}, v_{i'j})$. But, the partitioning which includes the edge of the form $(v_{ij}, v_{i'j})$ actually has a lower cost than the ground truth partitioning which does not contain that edge. Hence, the optimal partitioning is not the ground truth partitioning and we have a contradiction.

There are exactly $k^2\ell - k\ell$ edges of the form $(v_{ij}, v_{i'j})$. One set of pairwise node constraints which will clearly result in the ground truth clustering is placing a cannot-link constraint on all edges of the form $(v_{ij}, v_{i'j})$. But, we can actually achieve the ground truth clustering with fewer pairwise node constraints. We can place $\ell-1$ must-link constraints in each cluster to force the ground truth clusters to be connected and place $k^2 - k$ across ground truth cluster cannot-link constraints. This is the smallest set of pairwise node constraints which fully separates the ground truth clusters. This results in a total of $k^2 + k\ell - 2k$ pairwise node constraints which for $\ell > 1$ is less than $k^2\ell - k\ell$.

Lastly, we will show that there is a set of $\exists$-constraints with a total of $k\ell+k-\ell-1$ features which ensures the ground truth clustering will be returned by the correlation clustering objective. For each of $k-1$ of the ground truth clusters, create a $\exists$-constraint for cluster i with the form $\{(-, \phi_i), (+, \phi_{k+1}), (+, \phi_{k+2}), \ldots, (+, \phi_{k+\ell})\}$. In order to satisfy this $\exists$-constraint, every node from the ith ground truth cluster will be required to be contained in a predicted cluster and none of the nodes from any other ground truth cluster will be contained in the same predicted cluster. The cluster we do not explicitly create a $\exists$-constraint for will be predicted since it is separated from all of the other clusters by the set of $\exists$-constraints. Observe that there are $\ell + 1$ features in each constraint, so the total number of features over all of the $\exists$-constraints is exactly $k\ell + k - \ell - 1$. The result follows.

□

We will show empirical analysis of efficiency of $\exists$-constraints compared with must-link and cannot-link constraints in Section 4.4.4.

4.2.2 Hardness of Satisfaction

Given the representational power and potentially combinatorially many ways a $\exists$-constraint could be satisfied, it seems intractable to efficiently infer a clustering which

satisfies all $\exists$-constraints Ξ in general. In fact, we show that even without considering the clustering objective it is NP-hard to return a clustering of the vertices that satisfies all of the $\exists$-constraints Ξ. Note the following proposition.

Proposition 18. *Given a graph $G = (V, E)$, a discrete feature map Φ, and a set of $\exists$-constraints Ξ, it is* NP*-hard to determine if there exists a clustering of the vertices V which satisfies all of the $\exists$-constraints.*

Proof. We will show a polynomial time reduction from SAT-CNF. Suppose we have an algorithm that can determine if there exists a clustering of the vertices of a graph which satisfies all of some set of $\exists$-constraints. Additionally, suppose we have an arbitrary SAT-CNF formula f. For every disjunctive clause c in f, create a feature ϕ_c and add it to the set of possible features. For every boolean variable x in f, create two vertices in the graph, namely v_x and $v_{\neg x}$. In addition, create a unique feature for each vertex in the graph (e.g. ϕ_x for v_x and $\phi_{\neg x}$ for $v_{\neg x}$). For every clause c in f that contains x (and similarly $\neg x$), add the corresponding clause feature ϕ_c to $\Phi(v_x)$ (and similarly $\Phi(v_{\neg x})$). Create the *all-clause* $\exists$-constraint $\{(+, \phi_c) : c \in f\}$; this constraint is satisfied if and only if each disjunctive clause evaluates to `True`. Lastly, create the set of *valid-assignment* $\exists$-constraints $\{(+, \phi_x), (-, \phi_{\neg x})\}$ for every boolean variable x.

Clearly this is a polynomial time reduction. If there exists a clustering of the vertices which satisfies all of the $\exists$-constraints, then the set of vertices in the cluster which satisfies the *all-clause* $\exists$-constraint directly corresponds to an assignment to the boolean variables that will be a satisfying assignment to f. For every vertex v in the *all-clause* cluster, if v corresponds to a variable x, then the satisfying assignment includes setting $x =$ `True`. Otherwise, v corresponds to the negation of a variable $\neg x$, and the satisfying assignment includes setting $x =$ `False`. The *valid-assignment* $\exists$-constraints ensure that the assignment will be valid since one of v_x or $v_{\neg x}$ can be in the cluster satisfying the *all-clause* $\exists$-constraint , but not both. If there does not exist a clustering of the vertices which will satisfy both the *all-clause* $\exists$-constraint and

valid-assignment ∃-constraints, then there exists at least one feature ϕ_c that cannot be satisfied in the *all-clause* cluster without violating the *valid-assignment* ∃-constraints. That means that clause c cannot be satisfied in the formula f. Hence, there does not exist a valid assignment to the boolean variables for which f is satisfied. The result follows. □

This directly implies the following corollary.

Corollary 19. *There does not exist an exact polynomial time algorithm that can guarantee all valid ∃-constraints will be satisfied by the inferred clustering, unless* P = NP.

This follows from a simple contradiction argument. If there existed a polynomial time clustering algorithm that can guarantee all valid ∃-constraints will be satisfied by the inferred clustering, then we could use this algorithm to solve the decision problem determined to be NP-hard in Proposition 18.

4.3 Algorithm

In this section, we present our inference algorithm for correlation clustering with ∃-constraints. We start by integrating the ∃-constraints into the optimization problem presented in (4.2). Our general approach is to jointly cluster the vertices and the ∃-constraints, so we create additional 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 $\mathbf{S}_\phi^\xi := \{v \in V : \phi \in \Phi(v) \wedge \Phi(v) \cap \xi^- = \emptyset\}$ be the set of candidate vertices that

could satisfy $\phi \in \xi^+$. The optimization problem with $\exists$-constraints integrated can be written as follows:

$$\begin{aligned} \max & \sum_{(u,v)\in E} w_{uv}\, x_u \cdot x_v \\ \text{s.t.} \quad & x_s \in \{e_1, e_2, \ldots, e_n\}, \quad s \in V \cup \Xi \\ & \sum_{v \in \mathbf{S}_\phi^\xi} x_v \cdot x_\xi \geq 1, \quad \phi \in \xi^+, \xi \in \Xi \\ & x_s \cdot x_\xi = 0, \quad s \in V \cup \Xi, \xi \in \Xi : s \perp \xi \end{aligned} \tag{4.3}$$

The second 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 optimization problem is intractable to optimize in general by Corollary 19. Our general approach is to relax Problem (4.3) to an SDP and then round the fractional solution from the SDP to obtain a predicted clustering of the vertices and the $\exists$-constraints.

4.3.1 SDP Relaxation

We relax (4.3) following the standard SDP relaxation used for max-agree correlation clustering. The decision variables in the optimization problem are changed from standard basis vectors whose dot products determine whether or not two elements (vertices or $\exists$-constraints) are in the same cluster to a positive semi-definite 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 semi-definite 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 $\exists$-constraint satisfaction are similar to (4.3).

Formally, the SDP relaxation is as follows:

$$\begin{aligned}
\max \sum_{(u,v)\in E} & w_{uv} X_{uv} \\
\text{s.t.} \quad & X \succeq 0, \\
& X_{ss} = 1, \quad s \in V \cup \Xi \\
& X_{st} \geq 0, \quad s,t \in V \cup \Xi \\
& \sum_{v \in \mathbf{S}_\phi^\xi} X_{v\xi} \geq 1, \quad \phi \in \xi^+, \xi \in \Xi \\
& X_{s\xi} = 0, \quad s \in V \cup \Xi, \xi \in \Xi : s \perp \xi
\end{aligned} \tag{4.4}$$

4.3.2 Trellis-based Rounding Algorithm

After we have solved the SDP relaxation, we need to round the fractional solution to maximize not only the number of $\exists$-constraints satisfied, but also the correlation clustering objective. We accomplish this by constructing a data structure over the potential (partial) clusterings of $V \cup \Xi$ and then selecting the global optimal clustering from that data structure using a dynamic programming algorithm.

4.3.2.1 Sparse Cluster Trellis

A *sparse cluster trellis* (or just *trellis*, for short) $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a directed acyclic graph where the vertices $\mathcal{V} \subseteq \mathcal{P}(V \cup \Xi)$ [69, 108]. The set of vertices always includes the *leaves* $\{s\}_{s \in V \cup \Xi}$ and the *root* $V \cup \Xi$. There exists a directed edge from one vertex $S_\ell \in \mathcal{V}$ (the *child*) to another $S_p \in \mathcal{V}$ (the *parent*) if S_ℓ is a maximal subset of S_p. For every child S_ℓ of S_p there also exists a *complimentary child* $S_r = S_p \setminus S_\ell \in \mathcal{V}$. We define the set of child, complementary child pairs for a vertex $S_p \in \mathcal{V}$ as $\texttt{splits}_{\mathcal{T}}(S_p) := \{(S_\ell, S_r) \in \mathcal{V}^2 : S_\ell \cup S_r = S_p \wedge S_\ell \cap S_r = \emptyset\}$. A sparse cluster trellis is a generalization of a hierarchical clustering tree which allows for more clusterings to be represented. In practice, we construct a sparse cluster trellis by first constructing

Algorithm 2 TrellisCut$(G, \Phi, \Xi, \mathcal{T})$

Initialize assignment map, $a[\,\cdot\,] : \mathcal{V} \to \mathcal{P}(\mathcal{V})$
Initialize constraint satisfaction map, $s[\,\cdot\,] : \mathcal{V} \to \mathbf{Z}_+$
Initialize objective value map, $o[\,\cdot\,] : \mathcal{V} \to \mathbf{R}$
for $S_p \in \texttt{toposort}_{\mathcal{T}}(\mathcal{V})$ **do**
  $a[S_p] \leftarrow \{S_p\}$
  $s[S_p] \leftarrow \sum_{\xi \in S_p \cap \Xi} \mathbb{1}\{S_p \cap V \text{ satisfies } \xi\}$
  $o[S_p] \leftarrow \sum_{u,v \in S_p \cap V} w_{uv}$
  for $(S_\ell, S_r) \in \texttt{splits}_{\mathcal{T}}(S_p)$ **do**
    if $(s[S_\ell] + s[S_r] > s[S_p]$
      or $(s[S_\ell] + s[S_r] = s[S_p]$
        and $o[S_\ell] + o[S_r] > o[S_p]))$ **then**
      $a[S_p] \leftarrow a[S_\ell] \cup a[S_r]$
      $s[S_p] \leftarrow s[S_\ell] + s[S_r]$
      $o[S_p] \leftarrow o[S_\ell] + o[S_r]$
return $a[V \cup \Xi]$

one or more binary hierarchical clusterings of $V \cup \Xi$ guided by the solution to the SDP, X^*, and then combining them into a single data structure [108].

4.3.2.2 Trellis-Consistent Partition

Given a trellis $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, a *trellis-consistent partition* is a clustering $\mathcal{C}$ of $V \cup \Xi$ such that $C \in \mathcal{V}$ for all $C \in \mathcal{C}$. A $\exists$-constraint ξ is satisfied in a trellis-consistent partition $\mathcal{C}$ if ξ is satisfied by $C \cap V$ for $\xi \in C \in \mathcal{C}$. The number of trellis-consistent partitions encoded by a trellis is exponential in $|\mathcal{V}|$. Since 4.3 only considers intra-cluster edges, we can extract the trellis-consistent partition which maximizes the objective using a dynamic programming algorithm which scales linearly in $|\mathcal{V}|$. We refer to the process of extracting a trellis-consistent partition as *cutting* the trellis.

4.3.2.3 Trellis Cut Algorithm

Once the trellis is constructed, we need to efficiently extract the optimal trellis-consistent partition. This is a multi-objective optimization problem: we want to maximize both the number of $\exists$-constraints satisfied in the trellis-consistent partition

(we know from Section 4.2.2 that it is intractable to satisfy all the $\exists$-constraints in general) and, secondarily, maximize the correlation clustering objective. We extract the optimal trellis-consistent partition using a dynamic programming algorithm where we memoize the best partial solution at each vertex in the trellis. To efficiently memoize the best partial solution at each vertex, we compute the partial solutions for each vertex in the trellis in topological order from leaves to root. Let $\texttt{toposort}_{\mathcal{T}}(\mathcal{V})$ be an ordered set of trellis vertices topologically ordered from leaves to root.

Algorithm 2 details the dynamic programming algorithm we use to optimally cut the trellis. The algorithm takes as input the graph $G = (V, E)$ including edge weights, a discrete feature map Φ, a set of $\exists$-constraints Ξ, and a trellis $\mathcal{T} = (\mathcal{V}, \mathcal{E})$. For each vertex S in the trellis, we initialize the following: an assignment map $a[\,\cdot\,] : \mathcal{V} \to \mathcal{P}(\mathcal{V})$ which stores the optimal partitioning of S, a constraint satisfaction map $s[\,\cdot\,] : \mathcal{V} \to \mathbf{Z}_+$ which stores the number of constraints satisfied by the optimal partitioning, and an objective value map $o[\,\cdot\,] : \mathcal{V} \to \mathbf{R}$ which stores the correlation clustering objective value for the optimal partitioning. For each vertex $S_p \in \mathcal{V}$ in topological order, we start by assigning the optimal partitioning to be all the vertices $s \in S_p$ to be in a single cluster and store the corresponding number of satisfied constraints and correlation clustering objective value in their respective maps. Then, for each child and complementary child pair (S_ℓ, S_r) we check if the union of optimal partitionings from S_ℓ and S_r is more favorable than the previously stored partitioning for S_p. After iterating over all vertices in the trellis in topological order, the algorithm returns the optimal partitioning of the root.

Proposition 20. *Given a graph $G = (V, E)$, a discrete feature map Φ, a set of $\exists$-constraints Ξ, and a trellis $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, Algorithm 2 returns the trellis-consistent partition which maximizes the number of $\exists$-constraints satisfied a priori, then maximizes the (4.3) value subject to maintaining the number of $\exists$-constraints satisfied.*

Proof. We proceed via induction over the trellis vertices in topological order from leaves to root. The base case is the children S_ℓ and S_r are leaves and the parent S_p represents the set of two vertices each contained as singletons in S_ℓ and S_r. There can be no $\exists$-constraints satisfied by the partial clustering $\{S_\ell, S_r\}$, so if any $\exists$-constraints are satisfied in S_p (the case where one of S_ℓ or S_r represents the set of a $\exists$-constraint and the other satisfies that $\exists$-constraint), then S_p will be chosen as the memoized clustering stored in $a[S_p]$. If no $\exists$-constraints are satisfied in S_p, then we just need to determine which partial clustering has the higher objective value, either $\{S_\ell, S_r\}$ or $\{S_p\}$. The algorithm will choose the partial clustering which has the higher objective values. The base case follows.

The inductive step is next. Without loss of generality, suppose S_p only has pair of children (S_ℓ, S_r). By assumption, $a[S_\ell]$ and $a[S_r]$ memoize the best (according to our objective stated in the proposition) partial clusterings of S_ℓ and S_r, respectively. Algorithm 2 will choose either $\{S_p\}$ or $a[S_\ell] \cup a[S_r]$ as the partial clustering to memoize in $a[S_p]$. Suppose $\{S_p\}$ does not have more satisfied $\exists$-constraints or a higher objective value than $s[S_\ell] + s[S_r]$ and $o[S_\ell] + o[S_r]$, respectively. Algorithm 2 will choose $a[S_\ell] \cup a[S_r]$ as the partial clustering to memoize in $a[S_p]$. Out of all of the trellis-consistent partial clusterings of S_p where no element $x_\ell \in S_\ell$ is contained in the same cluster as any element $x_r \in S_r$, the partial clustering $a[S_\ell] \cup a[S_r]$ will be the best, since the objective value and the number of $\exists$-constraints satisfied in a partial clustering decompose given this required bifurcation of the set of elements. Thus, the inductive hypothesis has been shown and the result follows. □

The critical idea of the above proof is that both objectives, the number of $\exists$-constraints satisfied by a trellis-consistent partition and the correlation clustering objective, decompose over the structure of the trellis. This enables the dynamic programming approach to leverage optimal partial solutions to efficiently compute the globally optimal trellis-consistent partition.

4.4 Experiments

We empirically evaluate the effectiveness of ∃-constraints and our inference algorithm in achieving the latent ground truth clustering on several *author coreference* datasets (sometimes referred to as *author name disambiguation*). We directly compare the benefits of ∃-constraints with must-link and cannot-link constraints. We accomplish this by simulating a series of rounds of a user generating one or more constraints based on the predicted clustering produced in the previous round and integrating the new constraint(s) into the next prediction. The simulation terminates when the latent ground truth clustering is predicted.

4.4.1 Datasets

We evaluate our constraints and algorithm on several author coreference datasets. Each author coreference dataset is composed of *mentions* (or *records*) of authors. The goal is to cluster the mentions such that each cluster contains all of the mentions of exactly one individual author. The mentions are preprocessed into *blocks*, easy to partition mentions. Commonly, each block is uniquely defined by the first initial and last name of an author. If two author mentions have completely different first initials or last names, they are safely assumed to be from two distinct authors. We use three publicly available datasets from S2AND [139], a state-of-the-art author coreference system[5].

To produce the edge weights on the graph for each block, we train the pairwise S2AND LightGBM [86] classifier model using the pairwise features detailed in Subramanian et al. [139]. We use the default and recommended 80/10/10 train/val/test split for each dataset, tuning all of the hyperparameters on the validation set. For each test set, we construct the complete graph for each block and for each edge assign edge weights based on the trained pairwise classifier output logits shifted by a thresh-

[5]https://github.com/allenai/S2AND

Table 4.1: Author Coreference Data Test Set Statistics.

	PubMed	**QIAN**	**SCAD-zbMATH**
# vertices	315	410	1,196
# clusters	34	77	166
# features	14,093	10,366	8,203
# blocks	5	38	120
min block size	12	2	2
mean block size	63.0	10.8	31.5
max block size	142	100	133

old tuned on the validation set. If the classifier score is above (below) the threshold, the edge weight will be positive (negative). The resulting graph will be taken as input into the simulation of clustering with human feedback. Test set statistics for each dataset can be seen in Table 4.1.

We extract discrete features from each record given the text features for each field in the record. Features we extract include, but are not limited to, names, affiliation n-grams, email address n-grams, paper title n-grams, co-author names, co-author affiliation n-grams, etc. All of these discrete features are visible in the predicted cluster features and can be used by the constraint generator simulating human feedback.

4.4.2 Constraint Generation

We implement an oracle constraint generator for both $\exists$-constraints and must-link and cannot-link constraints to efficiently and effectively simulate a human user providing feedback. At each round, the generator has access to latent ground truth clustering, the predicted clustering, and the discrete features for all of the mentions. To generate a constraint (or set of constraints), the generator first selects a predicted cluster and a target gold cluster. The generator selects the gold cluster that has the lowest similarity with its most similar predicted cluster (computed based on Jaccard similarity of the sets of vertices). This pair of gold and predicted clusters has the most potential for improving the clustering.

After choosing a pair of gold and predicted clusters, the generator constructs constraint(s) which aim to correct the predicted cluster to look more like the gold cluster. There are three sets of vertices in the pair of gold and predicted clusters: vertices in both the gold and predicted clusters (overlapping vertices), vertices in the gold cluster and not in the predicted cluster (positive vertices), and vertices in the predicted cluster and not in the gold cluster (negative vertices). To generate a $\exists$-constraint, the generator picks a salient feature from the overlapping vertices, a salient feature from each predicted cluster containing any of the positive vertices, and a salient feature from each gold cluster containing any of the negative vertices. We have two modes for generating must-link and cannot-link constraints: batched and single. To generate a batch of must-link and cannot link constraints, the generator picks a vertex from the overlapping vertices and creates a must-link constraint to a vertex in each predicted cluster containing any of the positive vertices, and creates a cannot-link constraint to a vertex in each gold cluster containing any of the negative vertices. To generate a single must-link or cannot link constraint, we generate a batch of must-link and cannot-link constraints and pick the constraint with the potential to correct the clustering of the highest number of vertices. Hence, the constraint generator will never generate a constraint (of any type) which is already satisfied.

4.4.3 Metrics

We use several metrics to compare $\exists$-constraints with must-link and cannot-link constraints. To directly compare the amount of human feedback input, we define *constraint units* (CU) to be two units for every must-link and cannot-link constraint and one unit for every feature in each $\exists$-constraint. The number of CUs is representative of how many selections users need to make to specify both types of constraints. We will use CU as the metric for comparing the efficiency of both types of constraints.

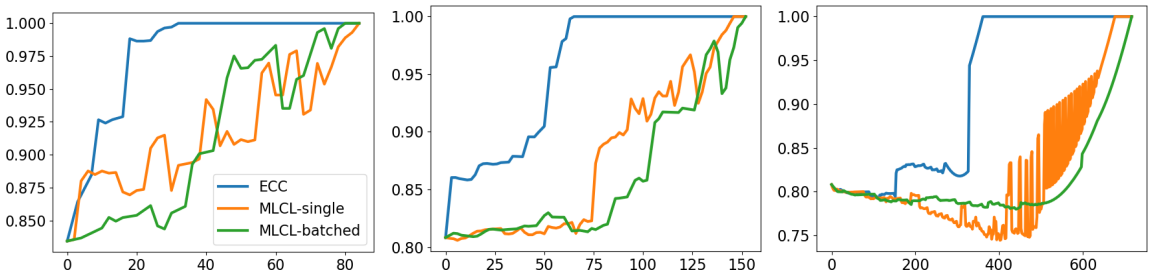
Table 4.2: Initial Clustering Metrics and Constraint Quantities Utilized to Achieve Latent Ground Truth Clustering.

	PubMed	**QIAN**	**SCAD-zbMATH**
Initial F1	95.6	96.0	96.8
Initial Rand Index	83.4	80.8	80.8
Initial FMC	92.2	87.9	91.2
# ECC	14	27	139
# MLCL-single	42	73	338
# MLCL-batched	40	76	358
CU ECC	32	65	361
CU MLCL-single	84	146	676
CU MLCL-batched	80	152	716

We use three different metrics to measure the quality of the predicted clustering in each round. First, we want a measure of the accuracy of the features of the predicted clusters. We define the *feature match coefficient* (FMC) to be the average Jaccard similarity between each gold cluster features and its most similar (by Jaccard similarity) predicted cluster features. The two other metrics we use are standard clustering metrics for measuring the accuracy of the clustering of the vertices: F1 (sometimes referred to as V-measure in the clustering literature) and Adjusted Rand Index.

4.4.4 Feedback Efficiency of ∃-constraints

In Table 4.2 we detail the initial clustering metrics (without any constraints) for each of the three author coreference datasets. We also show both the number of constraints and number of constraint units generated by the constraint generator before achieving the latent ground truth clustering. As can be seen in the table, existential cluster constraints required nearly half as many constraint units as must-link and cannot-link constraints to achieve the ground truth clustering. Figures 4.1&4.2 show the trajectories of both Adjusted Rand Index and FMC, respectively, with respect to the number of CUs as constraints were generated. As can be seen, we find that the



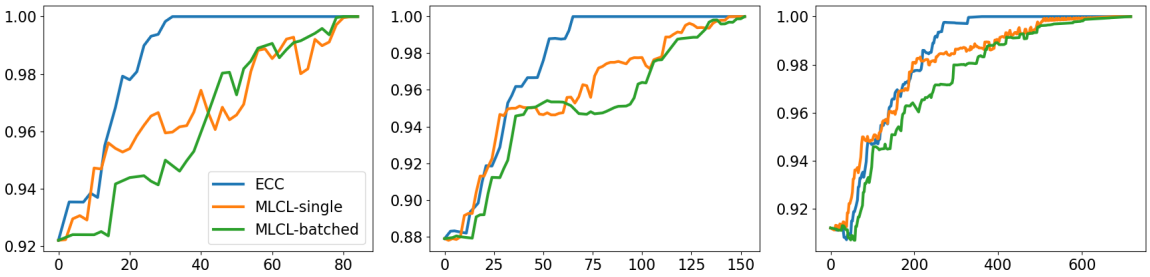
(a) PubMed **(b)** QIAN **(c)** SCAD-zbMATH

Figure 4.1: Adjusted Rand Index (↑) vs. CU. In each plot, the x-axis is the number of constraint units (CU) used by each feedback constraint method and the y-axis is the corresponding Adjusted Rand Index of the resulting clustering after incorporating the generated constraints. The blue line is existential cluster constraints (ECC), the orange line is must-link and cannot-link constraints generated one at a time (MLCL-single), and the green line is must-link and cannot link constraints generated in batches (MLCL-batched) to mimic ECC.

existential clustering constraints provide for efficiency improvements in terms of the number of constraint units compared to the must-link and cannot-link constraints. These trends seem to hold across each of the metrics considered.

4.4.5 USBS for Interactive Entity Resolution with ∃-constraints

In this section, we evaluate the performance of the CGAL and USBS with and without warm-starting on the same three author coreference datasets. 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 [7, 139]. 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



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

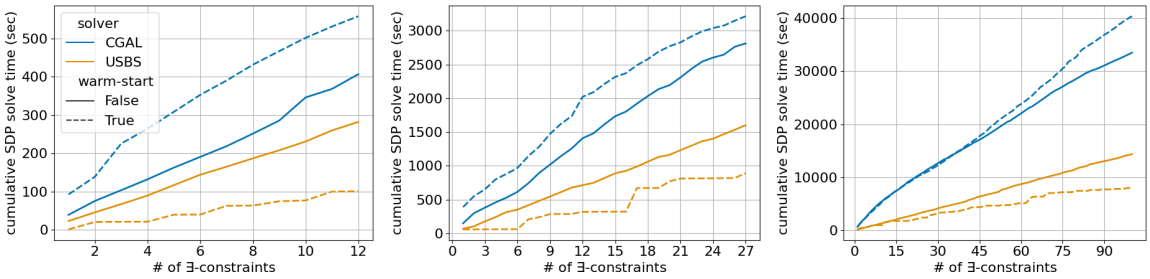
Figure 4.2: FMC (↑) vs. CU. In each plot, the x-axis is the number of constraint units (CU) used by each feedback constraint method and the y-axis is the corresponding feature match coefficient (FMC) of the resulting clustering after incorporating the generated constraints. The blue line is existential cluster constraints (ECC), the orange line is must-link and cannot-link constraints generated one at a time (MLCL-single), and the green line is must-link and cannot link constraints generated in batches (MLCL-batched) to mimic ECC.

Table 4.3: Merged author coreference dataset statistics.

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

one. We then sparsify this highly structured similarity matrix using a spectral sparsifier [134]. Table 4.3 details the dataset statistics including the number of non-zeros in the pairwise similarity matrix after spectral sparsification, denoted nnz(W).

We simulate ∃-constraint generation using the same oracle implemented in the previous sections. 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 ∃-constraint which is satisfied by the ground truth clustering, but is not satisfied by the predicted clustering. The oracle generates ∃-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 (3.2) and the following absolute infeasibility condition are satisfied



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

Figure 4.3: 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 4.3 for dataset sizes and details.

$$\|\mathcal{A}X - \text{proj}_{\mathcal{K}}(\mathcal{A}X)\|_\infty \leq \varepsilon. \tag{4.5}$$

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 ∃-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 ∃-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 ∃-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 ∃-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 ∃-constraints and mentions, we find that scaling up the norms of A_i's corresponding to

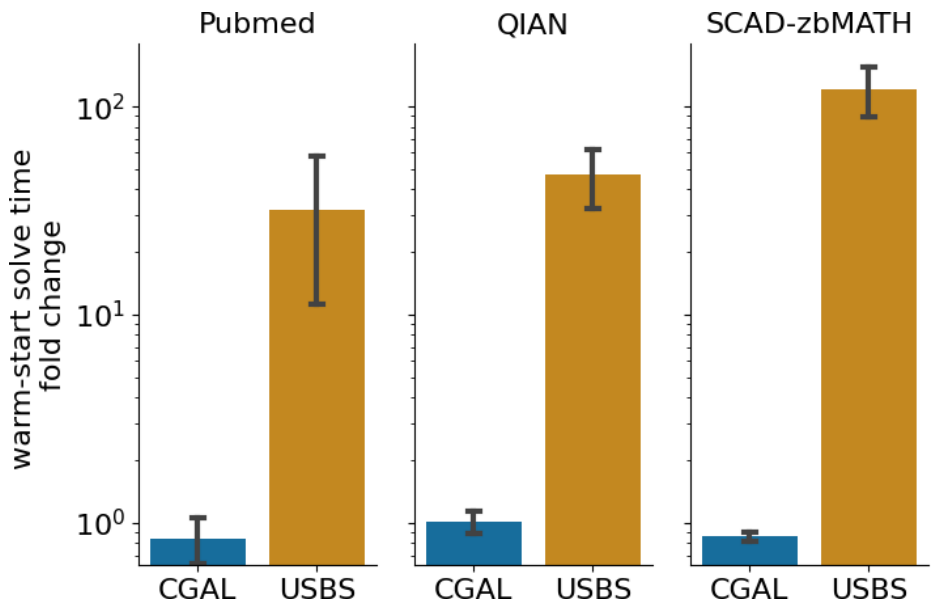


Figure 4.4: Average warm-start fold change (↑). Warm-start fold change represents the ratio of SDP solve time without warm-starting divided by the SDP solve time with warm-starting per ∃-constraint. The error bars indicate one standard deviation from the mean. We observe that USBS is able to much more reliably leverage a warm-start initialization. In addition, we observe that the performance gap between CGAL and USBS grows as the problem size grows.

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 ∃-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.

We simulate ∃-constraint generation using the same oracle as used in the previous sections. Figure 4.3 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.

Figure 4.4 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 4.3. 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 4.3. 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 $\exists$-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 4.4 shows that warm-starting provides more of a benefit the larger the SDP we are trying to solve.

4.5 Conclusion

In this chapter, we introduced a new feedback paradigm for interactive clustering in which users provide existential constraints on the features of clusters. We advocated for existential cluster constraints as a natural form of feedback that does not require users to inspect many individual data points, but rather use their mental model of what the output clustering should look like to define clustering constraints. We introduced an inference algorithm for incorporating existential cluster constraints into the correlation clustering framework. We demonstrated that our proposed feedback method provides efficiency improvements as compared to must-link and cannot-link constraints. In real-world settings, data is arriving incrementally all the time, so we showed that USBS is an efficient method for incorporating new data and existential cluster constraints as they arrive via warm-starting.

CHAPTER 5

TOWARDS MIXED-INTEGER SEMIDEFINITE OPTIMIZATION WITH USBS

Mixed-integer programming (MIP) is the subset of optimization problems where some of the decision variables are required to be integers [38]. The most common and well-studied form of MIPs are mixed-integer linear programs (MILP) [15, 65, 67, 84]. This is due to many important real-world problems being easily relaxed into LPs [39] and a rich understanding of LPs [40, 66]. MILP solvers employ LP solvers as subroutines of heuristic algorithms such as cutting planes [67], branch and bound [95], branch and cut [122], and column generation [18]. Moreover, there are efficient methods for solving LPs and it is understood how we can warm-start initialize such solvers. Warm-starting is critical to the practical success of MILP solvers, since every solver successively adds additional constraints to the relaxed problem in order to satisfy the integral constraints. These algorithms enable efficient practical approaches for real-world MILPs, but do have a worst case exponential running time, since MILPs are NP-hard to solve in general [85].

There are several real-world problems (e.g. Sparse PCA [24]) that can more efficiently and effectively be expressed as mixed-integer semidefinite programs (MISDP), i.e. semidefinite programs where some of the decision variables are constrained to be integers [58]. While there has been some work on methods for solving MISDPs [58, 78, 89, 112, 125], progress has been slowed for several reasons, including not knowing how to warm-start interior point methods, the standard method for solving SDPs. The ability to warm-start the simplex method for solving LPs enabled a vast number of heuristic methods for practically solving MILPs, including cutting

planes and column generation techniques, to name a few. The existence of a SDP solver that can effectively utilize a warm-start initialization would be a first step towards practical solving of MISDPs. We have already shown in a variety of scenarios how USBS can be warm-started, but not as a subroutine of a branch and bound algorithm. Additionally, Gally et al. [58] do note that spectral bundle methods could be a viable option for solving MISDPs as part of a branch and bound algorithm due to their theorized amenability to warm-starting. In this chapter, we provide evidence that USBS can be effectively warm-started as part of a branch and bound algorithm for solving MISDPs.

5.1 Existing Approaches to Solving MISDPs

Before discussing how USBS can be utilized within a standard branch and bound framework, we begin by discussing the fairly small amount of previous work on solving MISDPs. We can formulate all MISDPs in the following primal form

$$\begin{aligned} \max_{X \in \mathbb{S}^n_+} \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \mathcal{A}_{\mathcal{I}'} X = b_{\mathcal{I}'} \\ & \mathcal{A}_{\mathcal{I}} X \leq b_{\mathcal{I}} \\ & X_{ij} \in \mathbb{Z}, \quad \forall (i,j) \in \mathcal{Q} \end{aligned} \tag{5.1}$$

where $\mathcal{Q}$ is an index set for all of the entries in the primal variable X which need to be integers. In many practical applications, these additional integrality constraints are actually binary constraints, i.e. $X_{ij} \in \{0, 1\}$. The existing work in this area tends to prefer to use the following dual form MISDP

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & \langle b, y \rangle \\ \text{s.t.} \quad & C - \mathcal{A}^* y \preceq 0 \\ & y_{\mathcal{I}} \geq 0 \\ & y_i \in \mathbb{Z}, \quad \forall i \in \mathcal{R} \end{aligned} \tag{5.2}$$

where $\mathcal{R} \subseteq [m]$ is the subset of components of the dual variable required to be integers. Matter and Pfetsch [112] show how this dual form can be converted into the primal form and vice versa. We will present prior work in terms of the dual formulation, but will use the primal formulation for describing our experimental setup and results due to its ease of understanding.

Gally et al. [58] present a unifying framework and implementation for solving MISDPs using a branch and bound algorithm combined with an interior point method as the SDP solver. They also present theoretical results demonstrating that strong duality holds even after adding branching constraints in certain instances and heuristics for tightening the relaxations at nodes in the search tree such as dual fixing. Finally, they present empirical results of their SCIP-based [27] implementation on three applications: truss topology design, cardinality constrained least squares, and minimum k-partitioning. The authors do mention that interior point methods used in their SDP-SCIP solver for MISDPs cannot be warm-started, but that spectral bundle methods could potentially be a better fit due to their potential to be warm-started. Several follow-up works provide improvements to this initial framework including presolving [112], detecting symmetries in the problem [78], and conflict analysis [125].

Kobayashi and Takano [89] present a very different branch and cut approach. They also focus on the dual form of MISDPs above. The first main idea is that they relax the positive semidefinite constraint on $C - \mathcal{A}^* y$ using the following observation,

$$C - \mathcal{A}^* y \succeq 0 \implies \operatorname{diag}(C - \mathcal{A}^* y) \geq 0.$$

By temporarily applying this relaxed constraint on the feasible region for the dual decision variable y, they can leverage existing MILP solvers to solve the relaxed form of the problem. Then, in order to make sure the positive semidefinite constraint is satisfied in the end, they progressively add linear constraints of the following form

$$d^\top (C - \mathcal{A}^* y) d \geq 0,$$

where d is a unit-normed minimum eigenvector of $C - \mathcal{A}^* y$. They repeat this process until $\lambda_{\min}(C - \mathcal{A}^* y) \geq -\varepsilon$ for some error tolerance ε. The largest benefit of this approach is that it utilizes all of the progress made on solving MILPs. Unfortunately, the naive implementation is very expensive since potentially $n + 1$ large MILPs need to be solved. Kobayashi and Takano [89] make this point clear and provide a clever implementation that leverages callback features available in many modern MILP solvers such as CPLEX and Gurobi to generate cuts during the MILP branch and bound procedure. The authors demonstrate the empirical effectiveness of this approach on synthetic and real-world MISDPs. This procedure is clever, easy to implement, and avoids the potential of numerical instabilities of SDP solvers, but could also be potentially very slow in practice, especially if the number of integrality constraints is very small compared to the problem size as in the clustering with $\exists$-constraints problem.

5.2 Importance of Explicit Inequality Modeling

One of the important algorithmic contributions of USBS is explicit modeling of both equality and inequality constraints. Before demonstrating the ability to warm-start USBS in the context of a branch and bound algorithm, we want to address the importance of inequality modeling. Inequality modeling is extremely important in the context of MISDPs since inequalities are used in almost every viable solving method,

e.g. cutting planes. While it is possible to express every SDP in standard form with only equality constraints, it might not be the most efficient and straightforward method. This section aims to provide evidence for explicit inequality modeling over putting SDPs in standard form.

To demonstrate the importance of explicit inequality modeling, we utilize the correlation clustering SDP and datasets from Chapter 4. We reiterate the precise SDP we aim to solve. Given a graph $G = (V, E)$ with edge weights w_{uv} edge $(u, v) \in E$ we define the correlation clustering SDP relaxation as follows

$$\begin{aligned} \max \sum_{(u,v)\in E} & w_{uv} X_{uv} \\ \text{s.t.} \quad & X \succeq 0, \\ & X_{vv} = 1, \quad v \in V \\ & X_{uv} \geq 0, \quad u, v \in V \end{aligned} \tag{5.3}$$

In this SDP, we have $|V|$ equality constraints and $|E|$ inequality constraints. In order to put this SDP in standard form, we need to increase the dimension of the decision X by $|E|$. Since X is positive semidefinite, all of its diagonal entries must be greater than zero. Hence, we can rewrite each inequality constraint as $X_{uv} - X_{\sigma_{uv},\sigma_{uv}} = 0$ where σ_{uv} is the index of the slack variable in the expanded primal variable X. Using standard form, we avoid having to do any explicit modeling of the inequalities, but it does put more of the burden on solving for eigenvalues and eigenvectors since the dimension has increased by one for each inequality constraint. One other obvious concern is the growth in the size of X. But, using similar ideas to those used to realize sketching, we can track a structured subset of X so the memory requirements do not need to be a burden on the standard form approach. In this section, we are mostly concern with the speed and accuracy of each of these approaches and present

evidence in favor of explicit inequality modeling. Note that this has already been somewhat hinted at by Helmberg and Kiwiel [74].

We empirically compare solving the standard form of (5.3) to the inequality formulation. Note the at a high level (not implementation level due to how subproblems are solved) USBS reduces to SpecBM [46] when there are not any inequality constraints. We use the same three datasets as in Chapter 4. The results are shown in Figures 5.1, 5.2, and 5.3 for Pubmed, QIAN, and SCAD-zbMATH, respectively. We use the hyperparameters $k_c = 8, k_p = 2, \rho = 0.1, \beta = 0.1$. We plot the objective residual, infeasibility gap, and max infeasibility against time (seconds) over a period of one hour. We observe that in general USBS with inequality constraints is faster and more stable than using slack variables with the standard form of the SDP. We note that all of these experiments were run on a compute node with a single NVIDIA A100 GPU making it much more favorable for the standard form than if these experiments were run on CPU.

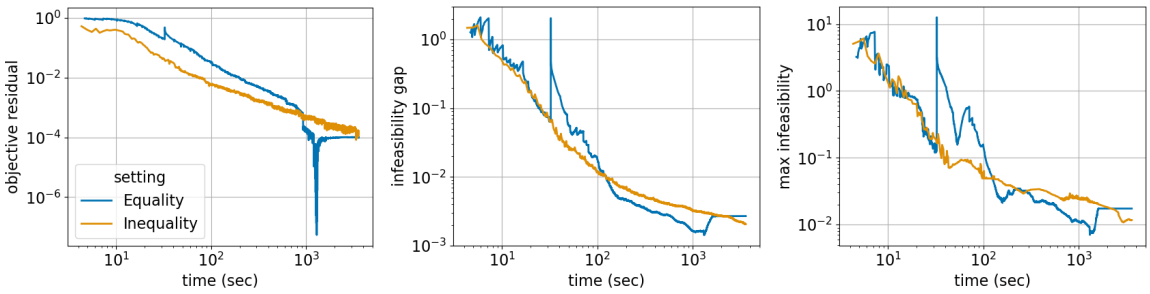


Figure 5.1: Equality Only vs. Inequality Correlation Clustering on Pubmed We compare SpecBM [46] (equality constraints only) to USBS on the correlation clustering problem without $\exists$-constraints . For SpecBM we have to add a dimension for each slack variable corresponding to the non-negativity inequality constraints in the standard correlation clustering SDP. In these plots, USBS and SpecBM are executed with $k_c = 8, k_p = 2, \rho = 0.1, \beta = 0.1$. All runs were executed on a compute node with a single NVIDIA A100 GPU. We see that USBS is much more stable and provides a higher quality solution quicker.

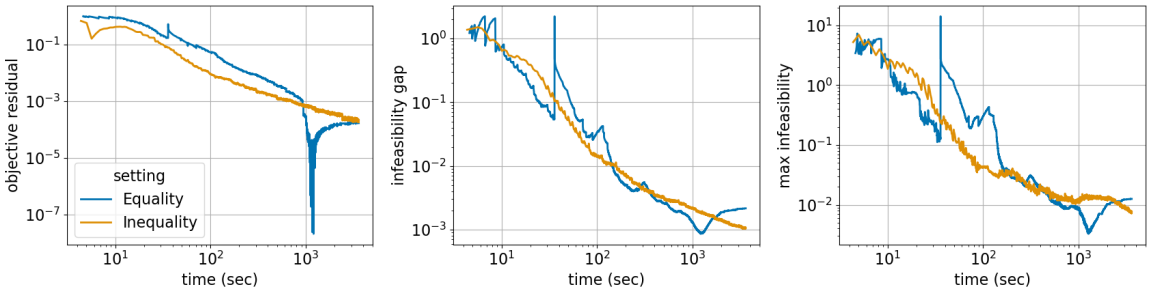


Figure 5.2: Equality Only vs. Inequality Correlation Clustering on QIAN We compare SpecBM [46] (equality constraints only) to USBS on the correlation clustering problem without ∃-constraints . For SpecBM we have to add a dimension for each slack variable corresponding to the non-negativity inequality constraints in the standard correlation clustering SDP. In these plots, USBS and SpecBM are executed with $k_c = 8, k_p = 2, \rho = 0.1, \beta = 0.1$. All runs were executed on a compute node with a single NVIDIA A100 GPU. We see that USBS is much more stable and provides a higher quality solution quicker.

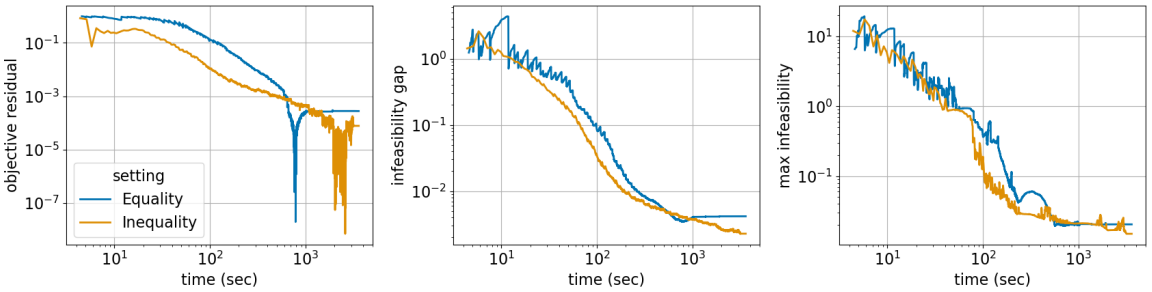


Figure 5.3: Equality Only vs. Inequality Correlation Clustering on SCAD-zbMATH We compare SpecBM [46] (equality constraints only) to USBS on the correlation clustering problem without ∃-constraints . For SpecBM we have to add a dimension for each slack variable corresponding to the non-negativity inequality constraints in the standard correlation clustering SDP. In these plots, USBS and SpecBM are executed with $k_c = 8, k_p = 2, \rho = 0.1, \beta = 0.1$. All runs were executed on a compute node with a single NVIDIA A100 GPU. We see that USBS is much more stable and provides a higher quality solution quicker.

5.3 Warm-starting for Branch and Bound

We now discuss how USBS can be warm-started to improve the efficiency of a branch and bound algorithm used to solve the correlation clustering with ∃-constraints problem. Recall the correlation clustering with ∃-constraints SDP from Chapter 4 as follows

$$
\begin{aligned}
\max \sum_{(u,v)\in E} & w_{uv} X_{uv} \\
\text{s.t.} \quad & X \succeq 0, \\
& X_{ss} = 1, \quad s \in V \cup \Xi \\
& X_{st} \geq 0, \quad s,t \in V \cup \Xi \\
& \sum_{v \in \mathbf{S}_\phi^\xi} X_{v\xi} \geq 1, \quad \phi \in \xi^+, \xi \in \Xi \\
& X_{s\xi} = 0, \quad s \in V \cup \Xi, \xi \in \Xi : s \perp \xi
\end{aligned}
\tag{5.4}
$$

Recall also that even with the rounding procedure detailed in Algorithm 2, we are not guaranteed to satisfy all of the ∃-constraints specified by the user, and in fact it is unlikely especially if the number of vertices in $\mathbf{S}_\phi^\xi$ is large. Moreover, it is NP-hard to even decide whether all ∃-constraints can even be satisfied or not. But, for practical purposes we want to try to find a clustering which satisfies all of the ∃-constraints. To this end, we will strive to solve the following MISDP instead

$$
\begin{aligned}
\max \sum_{(u,v)\in E} & w_{uv} X_{uv} \\
\text{s.t.} \quad & X \succeq 0, \\
& X_{ss} = 1, \quad s \in V \cup \Xi \\
& X_{st} \geq 0, \quad s,t \in V \cup \Xi \\
& \sum_{v \in \mathbf{S}_\phi^\xi} X_{v\xi} \geq 1, \quad \phi \in \xi^+, \xi \in \Xi \\
& X_{v\xi} \in \{0,1\}, \quad \forall v \in \mathbf{S}_\phi^\xi, \phi \in \xi^+, \xi \in \Xi \\
& X_{s\xi} = 0, \quad s \in V \cup \Xi, \xi \in \Xi : s \perp \xi
\end{aligned}
\tag{5.5}
$$

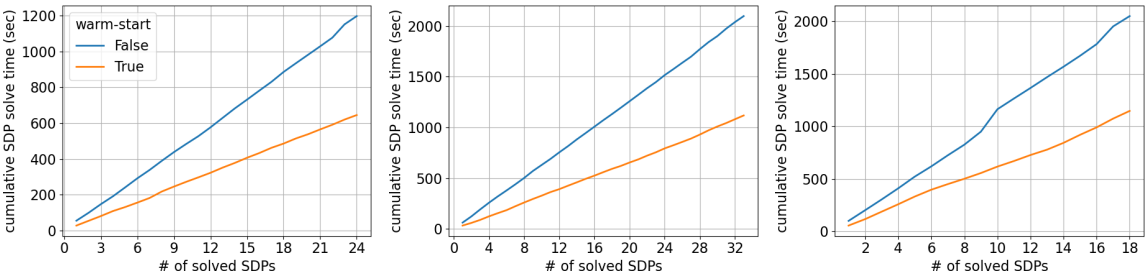
In this MISDP, we have added a constraint to every vertex that could possible satisfy every positive feature of any ∃-constraint in the set Ξ. Note that this is the only constraint that needs to be added to satisfy all of the ∃-constraints if possible. The

negative features will never be violated due to the nature of ∃-constraints, but the positive features might not all be satisfied in the original SDP.

In order to solve the optimization proposed in 5.5, we propose a very simple branch and bound procedure. Each node in the search tree branches on the representative vertex which satisfies a particular positive feature in a particular ∃-constraint. Branching involves adding an additional constraint to the original SDP which forces $X_{v\xi} = 1$ for some $v \in \mathbf{S}_{\phi}^{\xi}$. If the optimal SDP objective is smaller than the best known objective for a rounded solution where all ∃-constraints are satisfied, we can stop branching off of that node and bound the search process. We do not worry about whether the other decision variables, i.e. $X_{u\xi}$ for $u \in \mathbf{S}_{\phi}^{\xi} \setminus \{v\}$, take on binary values or not, since we just need one of these vertices to satisfy this particular positive feature in the particular ∃-constraint ξ.

Just as in Chapter 4, we simulate ∃-constraints being generated and solve problem 5.5 after each ∃-constraint is added. The goal of this setup is to test how well USBS can be warm-started as part of the branch and bound procedure described above. In the search tree, we can easily warm-start a child node from the solution to the parent node. The only change we make is in the warm-start solution is we set $y_i^* = 1$ where i is the index of the dual variable corresponding to what used to be $X_{v\xi} \geq 0$ in the parent SDP and now corresponds to the constraint $X_{v\xi} \geq 1$ which forces $X_{v\xi} = 1$ given the other constraints. We will then show the difference in speed to reach an approximate solution to the SDP at each node in the search by optimizing from both a cold-start solution (no warm-start information) and a warm-start initialization. Note that we already have some idea that this should work given the results in Chapter 4 where we warm-start after each ∃-constraint is added to the problem.

Figure 5.4 shows the timing results for cold-starting vs. warm-starting USBS during the MISDP branch and bound algorithm, aggregated over all ∃-constraints. We see that warm-starting USBS leads to roughly a 2x speed up in convergence to a



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

Figure 5.4: Cumulative SDP solve time (↓) vs. number of solved SDPs. In each plot (one for each author coreference dataset), the x-axis is the number of SDPs solved in the cumulative search trees 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, USBS is initialized using the solution from the previous SDP (i.e. the parent node). ∃-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 is consistent for all dataset sizes.

ε-approximate solution when warm-starting from the parent node in the search across all three datasets. We also note that in the experiments in Chapter 4 many more ∃-constraints needed to be generated in order to achieve the ground truth clustering due to not all of the ∃-constraints being satisfied at the time of their addition to the problem using the algorithm presented in Chapter 4. It is important to note that many fewer and less targeted ∃-constraints are required when using this branch and bound MISDP approach to correlation clustering with ∃-constraints .

5.4 Discussion

We have shown that USBS can be warm-started to speed up a branch and bound algorithm for solving MISDPs. Although other work has hinted at using a spectral bundle method as part of a MISDP solver, this is the first push towards actually showing that it is possible. One problem that likely still needs to be solved is addressing the issue of finding a high quality solution to the SDP at each node in the search tree. It is a problem that USBS and spectral bundle methods in general have

a difficult time reaching as high quality of a solution as interior point methods. For some applications though, it might not be that big of an issue, such as the type of problem we used here, clustering with ∃-constraints. There is still a lot of work to do in terms of heuristics such as cutting planes column generation, but we show evidence that USBS could serve as a good base for a MISDP solver especially for larger problems with fewer integrality constraints such as clustering with ∃-constraints . We note that improvements to hyperparameters and warm-start strategies could improve performance as well. There is still a lot of work to be done before this is practical for most real-world problems, but in this chapter we have shown evidence that USBS could be part of a MISDP solution.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we have provided evidence for USBS being a practical algorithm used to solve certain types of SDPs. First, we have shown that USBS is fast and scalable for solving large, sparse, and weakly constrained SDPs. This is the type of problem that we have considered most throughout this thesis. We have shown this evidence on both canonical and novel SDPs for a specific knowledge base application. In addition, to the best of our knowledge, we show the first and most thorough empirical evidence for a spectral bundle method being able to effectively utilize a warm-start initialization.

We also show how this can be extended into the mixed-integer semidefinite optimization domain. There has been little work done so far on mixed-integer semidefinite optimization for a variety of reasons – one of the main reasons being that there is not a fast and efficient method that can be reasonably warm-started. We show through its use in a branch and bound algorithm that USBS can be efficiently warm-started in this scenario. While for most applications we do not need an extremely accurate solution, there are scenarios where it is NP-hard to find a feasible integer solution, let alone a high quality one. In clustering with existential cluster constraints, a problem of this type, we find that using a branch and bound algorithm with USBS enables the need for fewer user feedback constraints, since the constraints will always be satisfied.

Semidefinite programming still has practical limitations, but this thesis starts to show practical evidence that certain classes of SDPs can be solved in reasonable time at scale. This evidence provides a spark for practical use of spectral bundle methods.

In this chapter, we discuss some of the potential improvements, other applications, and future work given the results and findings presented in this thesis.

6.1 More and Less Amenable Applications of USBS

In this section, we will discuss the applications for which USBS is likely to be beneficial and also applications where USBS is likely to struggle. As stated throughout this thesis, USBS is likely to work well for weakly-constrained SDPs where we do not need an extremely accurate solution. Weakly-constrained SDPs are most commonly the result of some combinatorial optimization problem relaxation. We demonstrate the empirical effectiveness of USBS on three types of combinatorial optimization problems of this nature. Other combinatorial optimization problems include various relaxations of graph problems such as Lovàsz Theta [104], Minimum Bisection [61], μ-conductance [81], and CutNorm [54] (see Lovász [105] for more potential applications. USBS is also amenable to problems where, in addition to the above requirements, the constraints and problem data is dynamic, especially due to the fact that USBS can be warm-started reliably. For example, many clustering problems where constraints are involved would benefit from using USBS [37, 60, 68, 82, 141, 161].

In contrast, there are several SDPs where USBS will likely not be the best algorithm for those particular applications. USBS struggles quite a bit with problems containing many constraints and slack variables. One example of a problem with many constraints and slack variables is the k-SparsePCA problem [24]. This problem requires several slack variables and has many constraints and we found that SCS [119] outperformed USBS on this task. We also expect USBS to struggle with problems where we want a high accuracy solution. USBS struggles in practice to converge to a high accuracy solution. Some of this might be due to the theoretical convergence rate and some of this might be due to error propagation from the subproblem solving (see Remark 10). We might be able to overcome this with higher accuracy subproblem

solvers and sufficiently large enough k given the linear convergence results of Ding and Grimmer [46], but further work needs to explore this further.

6.2 Sum of Squares

Throughout this thesis, we have mostly focused on the benefits of USBS for combinatorial optimization problems. While these are important applications of semidefinite programming, another very important application is problems related to sum-of-squares polynomial optimization [123]. Similar to combinatorial optimization, semidefinite programming applied to sum of squares optimization problems is mostly used as a theoretical tool. But, there are a whole swath applications in the sum of squares optimization landscape where improvements in semidefinite programming, especially on the scalability front, would be very useful [12, 72].

One fundamental problem in sum of squares is determining if a polynomial is globally non-negative [123]. Equivalently, this problem can be reformulated to find the exact global minimum of a polynomial. This amounts to solving a particular semidefinite program, but is very interesting because the answer to the problem we are trying to solve does not require us to store the primal decision variable at all. USBS is quite amenable to this type of problem since it does not require storing the primal matrix X_t explicitly. This property is what allows us to use the matrix sketching technique for a reduction in memory requirements. This property, in combination with faster convergence as a first order method, makes USBS particularly attractive as a solver for these types of sum of squares problems.

In addition, it has been shown that some large-scale polynomial optimization problems have a constant trace property that can be exploited for more efficient solving [109, 110]. In these works, CGAL has been the method of choice for solving these problems, and hence, as evidenced by the work in this thesis, USBS could be a better alternative for solving these problems. There is still work to be done on

this front, but USBS and improvements on it could be viable methods for solving polynomial optimization problems.

6.3 Improvements for General MISDO

In this work, we have only shown a proof of concept for using USBS as part of a mixed-integer semidefinite programming solver. We have shown with a very simplistic branch and bound algorithm that USBS can be successfully warm-started using the solution to the parent node SDP in the search tree. While we have have shown that this works consistently and effectively, there could be improvements to the warm-start initialization or scheme to improve performance. Additionally, a greater understanding of the convergence properties and warm-start dynamics would be beneficial in designing a MISDP solver. These warm-start improvements would be great, but not sufficient for building a practical MISDP solver.

In any modern MISDP solver, we would need many other components and heuristics in order to make the solver practical. These components include cutting planes, handling symmetries, pre-solving, and other algorithmic improvements to the search algorithm [58, 78, 112, 125]. Some of this work has been done for MISDP solvers already, but only in combination with interior point methods which are impractically slow for problems with any reasonable size.

It would also be interesting to consider solving problems where the problem size is large, but the number of integrality constraints is relatively small. In this case, any solver based on interior point methods would inevitably fail to scale to the large problem size. However, a solution based with USBS as the SDP solver could have a better chance of solving these types of problems. One example of this is actually the clustering with existential cluster constraints problem considered in this thesis. A generalization of this is any problem that is NP-hard to find a feasible integer

solution, i.e. it is NP-hard to round. We are unaware of other problems of this type, but advocate for using USBS as part of a MISDP solver in this scenario.

6.4 Algorithmic Improvements to USBS

There are several algorithmic improvements that could be considered to improve the performance of USBS and scalable semidefinite programming in general. First, we anecdotally find that using another method like CGAL to warm-start USBS can improve convergence. In general, CGAL can find a reasonable solution very quickly and USBS finds a higher quality solution in less time. So the convergence curves overlap somewhere around a low accuracy solution, depending on the problem size. We could sensibly combine these approaches to obtain an even better hybrid solution, especially given the warm-start capabilities of USBS. This could extend well beyond CGAL, namely we could use any solver to warm-start USBS including SDPLR [31], SDPLR+ Huang and Gleich [80], ManiSDP [154], among others. We could also consider using USBS to relieve SDPLR from local minima.

Several modifications could be made to improve the convergence of USBS itself. Approximate second-order methods could be used to scalably improve the convergence of USBS [76], as is also mentioned in [46]. One of the biggest bottlenecks in the current approach is the alternating maximization for incorporating inequality constraints. Techniques for improving the performance of this alternating maximization method would greatly speed up the convergence of USBS. One simple change to the algorithm that would improve theoretical performance would be to use an adaptive smoothing parameter ρ_t, i.e. change ρ over time [45]. An adaptive schedule for ρ could hurt warm-start performance since we would not necessarily know what ρ should be initially. Lastly, the current warm-start initialization strategies are somewhat primitive and not principled other than the inherent properties of USBS. Improved understanding of warm-start dynamics could be the subject of interesting future work.

APPENDIX

A.1 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 (2.23). Before we can derive the algorithms, it is necessary to define the svec operator and symmetric Kronecker product [5, 131, 142].

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

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

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

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

For any $M \in \mathbb{R}^{n\times n}$, let $\text{vec}(M)$ 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 $\text{vec}(A) \mapsto \text{svec}(A)$ for any $A \in \mathbb{S}^n$. Let $u_{ij,kl}$ be the entry in the row which defines element a_{ij} in $\text{svec}(A)$ and the column that is multiplied with the element a_{kl} in $\text{vec}(A)$. 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

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

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 $\text{svec}(A)$ for $A \in \mathbb{S}^n$ as follows

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

Alternatively, but equivalently [131], 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.

A.1.1 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

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

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 (A.1). We follow the well known technique for deriving primal-dual interior point methods [73]. We start by defining the Lagrangian of the dual barrier problem of (A.1),

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

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 (A.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 (A.2) are the following system of equations

$$\nabla_S L_\mu = g_1 - \text{svec}(T) + \omega v_I = 0 \tag{A.3}$$

$$\nabla_\eta L_\mu = g_2 - \zeta + \omega = 0 \tag{A.4}$$

$$\nabla_T L_\mu = S - \mu T^{-1} = 0 \tag{A.5}$$

$$\nabla_\zeta L_\mu = \eta - \mu\zeta^{-1} = 0 \tag{A.6}$$

$$\nabla_\omega L_\mu = 1 - v_I^\top \text{svec}(S) - \eta - \mu\omega^{-1} = 0 \tag{A.7}$$

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 (A.5), (A.6), and/or (A.7) to solve for μ,

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

The fundamental idea of primal-dual interior point methods is to use Newton's method to follow the central path to a solution of (A.1). Before we can apply Newton's method, we must linearize the non-linear equations (A.5), (A.6), and (A.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 [5] or [73] for more on the choice of linearization). We choose one standard method linearization, detailed in the following system of equations

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

The solution $\theta_\star$ to this system of equations $\mathrm{F}_\mu(\theta) = 0$ satisfies the first-order optimality conditions (A.3)-(A.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 (A.9))

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

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

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

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

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

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

$$\left(T \otimes_s S^{-1} + \frac{\zeta\eta^{-1}}{\kappa_1\zeta\eta^{-1}+1} v_I v_I^\top\right) \operatorname{svec}(\Delta S) = v_I g_2 - v_I \mu\eta^{-1} - g_1 + \mu \operatorname{svec}(S^{-1}) \\ + v_I \zeta\eta^{-1}(-\kappa_1\zeta\eta^{-1} - 1)^{-1}\left(-\kappa_1(\mu\eta^{-1} - g_2 - \omega) + \mu\omega^{-1} + v_I^\top \operatorname{svec}(S) + \eta - 1\right) \tag{A.15}$$

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

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

where, as done in [73], we use (A.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 [75], we compute a non-increasing estimate of the barrier parameter

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

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

A.1.2 Solving (2.23)

The subproblem (2.23) can be rewritten as the following (small) quadratic semidefinite program

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

For this subproblem, unlike the subproblem solved in subsection A.1.1, 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

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

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 A.1.1, so we proceed by including the details which differ from the previous section. The Lagrangian of the

dual barrier problem of (A.16) is as follows

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

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

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

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

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

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

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

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

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

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

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

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

We make a single change to the initialization as compared with the interior point method presented in subsection A.1.1. 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.

BIBLIOGRAPHY

[1] Dhruv Agarwal, Rico Angell, Nicholas Monath, and Andrew McCallum. Entity linking and discovery via arborescence-based supervised clustering. *arXiv preprint arXiv:2109.01242*, 2021.

[2] Dhruv Agarwal, Rico Angell, Nicholas Monath, and Andrew McCallum. 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.

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

[4] Farid Alizadeh, Jean-Pierre A Haeberly, and Michael L Overton. Complementarity and nondegeneracy in semidefinite programming. *Mathematical programming*, 1997.

[5] Farid Alizadeh, Jean-Pierre A Haeberly, and Michael L Overton. Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results. *SIAM Journal on Optimization*, 1998.

[6] Rico Angell, Nicholas Monath, Sunil Mohan, Nishant Yadav, and Andrew McCallum. Clustering-based inference for biomedical entity linking. *arXiv preprint arXiv:2010.11253*, 2020.

[7] Rico Angell, Nicholas Monath, Nishant Yadav, and Andrew Mccallum. Interactive correlation clustering with existential cluster constraints. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[8] Pierre Apkarian, Dominikus Noll, and Olivier Prot. A trust region spectral bundle method for nonconvex eigenvalue optimization. *SIAM Journal on Optimization*, 2008.

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

[10] Sanjeev Arora, Elad Hazan, and Satyen Kale. log n) approximation to sparsest cut can be found in o (n2) time. In *Proc. of 45th FOCS*, 2004.

[11] Sanjeev Arora, Elad Hazan, and Satyen Kale. 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.

[12] Francis Bach, Elisabetta Cornacchia, Luca Pesce, and Giovanni Piccioli. Theory and applications of the sum-of-squares technique. *arXiv preprint arXiv:2306.16255*, 2023.

[13] David A. Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. 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.

[14] Juhee Bae, Tove Helldin, Maria Riveiro, Sławomir Nowaczyk, Mohamed-Rafik Bouguelia, and Göran Falkman. Interactive clustering: A comprehensive review. *ACM Computing Surveys (CSUR)*, 2020.

[15] Egon Balas. An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 13(4):517–546, 1965.

[16] Maria-Florina Balcan and Avrim Blum. Clustering with Interactive Feedback. In *International Conference on Algorithmic Learning Theory*, 2008.

[17] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation Clustering. *Machine Learning*, 2004.

[18] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[19] Burak Bartan and Mert Pilanci. Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time. *arXiv preprint arXiv:2101.02429*, 2021.

[20] Alexander I. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, 13:189–202, 1995.

[21] Sumit Basu, Danyel Fisher, Steven Drucker, and Hao Lu. Assisting users with clustering tasks by combining metric learning and classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

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

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

[24] Dimitris Bertsimas, Ryan Cory-Wright, and Jean Pauphilet. Solving large-scale sparse pca to certifiable (near) optimality. *Journal of Machine Learning Research*, 23(13):1–35, 2022.

[25] Adel Bibi, Baoyuan Wu, and Bernard Ghanem. Constrained k-means with general pairwise and cardinality constraints, 2019.

[26] Olivier Binette and Rebecca C Steorts. (Almost) all of entity resolution. *Science Advances*, 2022.

[27] Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. The SCIP Optimization Suite 9.0. ZIB-Report 24-02-29, Zuse Institute Berlin, February 2024. URL https://nbn-resolving.org/urn:nbn:de:0297-zib-95528.

[28] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 2011.

[29] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

[30] José FS Bravo Ferreira, Yuehaw Khoo, and Amit Singer. Semidefinite programming approach for the quadratic assignment problem with a sparse graph. *Computational Optimization and Applications*, 2018.

[31] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical programming*, 2003.

[32] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. Qaplib–a quadratic assignment problem library. *Journal of Global optimization*, 1997.

[33] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, 2007.

[34] Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

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

[36] Anni Coden, Marina Danilevsky, Daniel Gruhl, Linda Kato, and Meena Nagarajan. A method to accelerate human in the loop clustering. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 237–245. SIAM, 2017.

[37] Mihai Cucuringu, Ioannis Koutis, Sanjay Chawla, Gary L. Miller, and Richard Peng. Scalable constrained clustering: A generalized spectral method. *ArXiv*, abs/1601.04746, 2015. URL https://api.semanticscholar.org/CorpusID:12795732.

[38] Robert J Dakin. A tree-search algorithm for mixed integer programming problems. *The computer journal*, 8(3):250–255, 1965.

[39] George B Dantzig. On the significance of solving linear programming problems with some integer variables. *Econometrica, Journal of the Econometric Society*, 1960.

[40] George Bernard Dantzig and Mukund Narain Thapa. *Linear programming: Theory and extensions*, volume 2. Springer, 2003.

[41] Sanjoy Dasgupta, Akansha Dey, Nicholas Roberts, and Sivan Sabato. Learning from discriminative feature feedback. In *NeurIPS*, pages 3959–3967, 2018.

[42] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, et al. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *Advances in Neural Information Processing Systems*, 2020.

[43] Ian Davidson and S Basu. Clustering with constraints., 2009.

[44] Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation Clustering in General Weighted Graphs. *Theoretical Computer Science*, 2006.

[45] Mateo Díaz and Benjamin Grimmer. Optimal convergence rates for the proximal bundle method. *SIAM Journal on Optimization*, 2023.

[46] Lijun Ding and Benjamin Grimmer. Revisiting spectral bundle methods: Primal-dual (sub) linear convergence rates. *SIAM Journal on Optimization*, 2023.

[47] Lijun Ding and Madeleine Udell. On the simplicity and conditioning of low rank semidefinite programs. *SIAM Journal on Optimization*, 2021.

[48] Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. In *Proceedings of the 2009 conference on Empirical methods in natural language processing*, 2009.

[49] Dmitriy Drusvyatskiy and Henry Wolkowicz. The many faces of degeneracy in conic optimization. *Foundations and Trends® in Optimization*, 2017.

[50] Avinava Dubey, Indrajit Bhattacharya, and Shantanu Godbole. A cluster-level semi-supervision model for interactive clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010.

[51] Jonathan Eckstein. *Splitting methods for monotone operators with applications to parallel optimization*. PhD thesis, Massachusetts Institute of Technology, 1989.

[52] Stefan Feltenmark and Krzysztof C Kiwiel. Dual applications of proximal bundle methods, including lagrangian relaxation of nonconvex problems. *SIAM Journal on Optimization*, 2000.

[53] Michel Fortin and Roland Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Elsevier, 2000.

[54] Alan M. Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19:175–220, 1999. URL https://api.semanticscholar.org/CorpusID:15231198.

[55] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *2007 IEEE 11th International Conference on Computer Vision*, 2007.

[56] Roy Frostig, Matthew James Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 2018.

[57] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1):17–40, 1976.

[58] Tristan Gally, Marc E Pfetsch, and Stefan Ulbrich. A framework for solving mixed-integer semidefinite programs. *Optimization Methods and Software*, 33 (3):594–632, 2018.

[59] Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 2010.

[60] Pierre Gançarski, T. Dao, Bruno Crémilleux, Germain Forestier, and Thomas Andrew Lampert. Constrained clustering: Current and new trends. In *A Guided Tour of Artificial Intelligence Research*, 2020. URL https://api.semanticscholar.org/CorpusID:218952844.

[61] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1:237–267, 1976. URL https://api.semanticscholar.org/CorpusID:205090369.

[62] Alex Gittens. *Topics in randomized numerical linear algebra*. California Institute of Technology, 2013.

[63] Roland Glowinski and Americo Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.

[64] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 1995.

[65] Ralph Gomory. Essentials of an algorithm for integer solutions to linear programs. *Recent Advances in Mathematical Programming*, 1958.

[66] Ralph E Gomory. On the relation between integer and noninteger solutions to linear programs. *Proceedings of the National Academy of Sciences*, 53(2): 260–265, 1965.

[67] RE Gomory. An algorithm for integer solutions to linear programs. rl graves and p. wolfe (eds.) recent advances in mathematical programming, 1963.

[68] Germ'an Gonz'alez-Almagro, Daniel Peralta, Eli De Poorter, José Ramón Cano, and Salvador García. Semi-supervised constrained clustering: An in-depth overview, ranked taxonomy and future research directions. *ArXiv*, abs/2303.00522, 2023. URL https://api.semanticscholar.org/CorpusID:257254995.

[69] Craig Greenberg, Nicholas Monath, Ari Kobren, Patrick Flaherty, Andrew McGregor, and Andrew McCallum. Compact Representation of Uncertainty in Clustering. In *Advances in Neural Information Processing Systems*, 2018.

[70] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. Georgia Institute of Technology, 2013.

[71] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 2011.

[72] Georgina Hall. Engineering and business applications of sum of squares polynomials. *arXiv preprint arXiv:1906.07961*, 2019.

[73] Christoph Helmberg. An interior point method for semidefinite programming and max-cut bounds. 1994.

[74] Christoph Helmberg and Krzysztof C Kiwiel. A spectral bundle method with bounds. *Mathematical Programming*, 2002.

[75] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 2000.

[76] Christoph Helmberg, Michael L Overton, and Franz Rendl. The spectral bundle method with second-order information. *Optimization Methods and Software*, 2014.

[77] Vicente Hernández, Jose E Román, Andrés Tomás, and Vicente Vidal. Krylov-schur methods in slepc. *Universitat Politecnica de Valencia, Tech. Rep. STR-7*, 2007.

[78] Christopher Hojny and Marc E Pfetsch. Handling symmetries in mixed-integer semidefinite programs. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 69–78. Springer, 2023.

[79] Qixing Huang, Yuxin Chen, and Leonidas Guibas. Scalable semidefinite relaxation for maximum a posterior estimation. In *International Conference on Machine Learning*. PMLR, 2014.

[80] Yufan Huang and David F Gleich. Suboptimality bounds for trace-bounded sdps enable a faster and scalable low-rank sdp solver sdplr+. *arXiv preprint arXiv:2406.10407*, 2024.

[81] Yufan Huang, C Seshadhri, and David F Gleich. Theoretical bounds on the network community profile from low-rank semi-definite programming. In *International Conference on Machine Learning*, pages 13976–13992. PMLR, 2023.

[82] Yuheng Jia, Junhui Hou, and Sam Tak Wu Kwong. Constrained clustering with dissimilarity propagation-guided graph-laplacian pca. *IEEE Transactions on Neural Networks and Learning Systems*, 32:3985–3997, 2020. URL https://api.semanticscholar.org/CorpusID:221358266.

[83] Roy Jonker and Ton Volgenant. 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.

[84] Michael Jünger, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A Wolsey. *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009.

[85] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, 1972.

[86] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 2017.

[87] Krzysztof C Kiwiel. Efficiency of proximal bundle methods. *Journal of Optimization Theory and Applications*, 2000.

[88] Dan Klein, Sepandar D Kamvar, and Christopher D Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, 2002.

[89] Ken Kobayashi and Yuich Takano. A branch-and-cut algorithm for solving mixed-integer semidefinite optimization problems. *Computational Optimization and Applications*, 2020.

[90] Ari Kobren, Nicholas Monath, and Andrew McCallum. Integrating user feedback under identity uncertainty in knowledge base construction. In *Automated Knowledge Base Construction (AKBC)*, 2019.

[91] Bernhard Kratzwald, Guo Kunpeng, Stefan Feuerriegel, and Dennis Diefenbach. IntKB: A verifiable interactive framework for knowledge base completion. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

[92] Jacek Kuczyński and Henryk Woźniakowski. Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications*, 13(4):1094–1122, 1992.

[93] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

[94] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 2009.

[95] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 1960.

[96] Claude Lemarechal, Jean-Jacques Strodiot, and André Bihain. On a bundle algorithm for nonsmooth optimization. In *Nonlinear programming 4*. Elsevier, 1981.

[97] Huamin Li, George C Linderman, Arthur Szlam, Kelly P Stanton, Yuval Kluger, and Mark Tygert. Algorithm 971: An implementation of a randomized algorithm for principal component analysis. *ACM Transactions on Mathematical Software (TOMS)*, 2017.

[98] Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th international conference on Machine learning*, 2008.

[99] Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Constrained clustering via spectral regularization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[100] Percy Liang, Michael I Jordan, and Dan Klein. Learning from measurements in exponential families. In *Proceedings of the 26th annual international conference on machine learning*, 2009.

[101] Feng-Yi Liao, Lijun Ding, and Yang Zheng. An overview and comparison of spectral bundle methods for primal and dual semidefinite programs. *arXiv preprint arXiv:2307.07651*, 2023.

[102] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[103] Eliane Maria Loiola, Nair Maria Maia De Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European journal of operational research*, 2007.

[104] László Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

[105] László Lovász. Semidefinite programs and combinatorial optimization. In *Recent advances in algorithms and combinatorics*, pages 137–194. Springer, 2003.

[106] Zhengdong Lu and Miguel A Carreira-Perpinan. Constrained spectral clustering through affinity propagation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[107] Zhi-Quan Luo and Paul Tseng. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 1993.

[108] Sebastian Macaluso, Craig Greenberg, Nicholas Monath, Ji Ah Lee, Patrick Flaherty, Kyle Cranmer, Andrew McGregor, and Andrew McCallum. Cluster Trellis: Data Structures & Algorithms for Exact Inference in Hierarchical Clustering. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021.

[109] Ngoc Hoang Anh Mai, Jean B Lasserre, Victor Magron, and Jie Wang. Exploiting constant trace property in large-scale polynomial optimization. *ACM Transactions on Mathematical Software*, 48(4):1–39, 2022.

[110] Ngoc Hoang Anh Mai, Jean-Bernard Lasserre, and Victor Magron. A hierarchy of spectral relaxations for polynomial optimization. *Mathematical Programming Computation*, 15(4):651–701, 2023.

[111] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):331–360, 2020.

[112] Frederic Matter and Marc E Pfetsch. Presolving for mixed-integer semidefinite optimization. *INFORMS Journal on Optimization*, 5(2):131–154, 2023.

[113] Arya Mazumdar and Barna Saha. Clustering with noisy queries, 2017.

[114] Robert Mifflin. An algorithm for constrained optimization with semismooth functions. *Mathematics of Operations Research*, 1977.

[115] Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 1960.

[116] Renato DC Monteiro, Arnesh Sujanani, and Diego Cifuentes. A low-rank augmented lagrangian method for large-scale semidefinite programming based on a hybrid convex-nonconvex approach. *arXiv preprint arXiv:2401.12490*, 2024.

[117] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 1957.

[118] Feiping Nie, Han Zhang, Rong Wang, and Xuelong Li. Semi-supervised clustering via pairwise constrained optimal graph. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021.

[119] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. SCS: Splitting Conic Solver, version 3.1.1. https://github.com/cvxgrp/scs, November 2021.

[120] Michael L Overton. Large-scale optimization of eigenvalues. *SIAM Journal on Optimization*, 1992.

[121] Brendan O'donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169:1042–1068, 2016.

[122] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[123] Pablo A Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96:293–320, 2003.

[124] Gábor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23 (2):339–358, 1998.

[125] Marc E Pfetsch. Dual conflict analysis for mixed-integer semidefinite programs. *Preprint, Optimization Online*, 2023.

[126] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *Advances in neural information processing systems*, 2018.

[127] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*. NIH Public Access, 2017.

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

[129] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. Sesync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 2019.

[130] Yusuke Sato and Makoto Iwayama. Interactive constrained clustering for patent document set. In *Proceedings of the 2nd international workshop on Patent information retrieval*, pages 17–20, 2009.

[131] Kathrin Schacke. On the kronecker product. *Master's thesis, University of Waterloo*, 2004.

[132] Burr Settles. Active learning literature survey. 2009.

[133] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing gaussian mixture models with em using equivalence constraints. *Advances in neural information processing systems*, 2004.

[134] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008.

[135] Stefan Sremac, Hugo J Woerdeman, and Henry Wolkowicz. Error bounds and singularity degree in semidefinite programming. *SIAM Journal on Optimization*, 2021.

[136] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. *Advances in Neural Information Processing Systems*, 2020.

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

[138] Jos F Sturm. Error bounds for linear matrix inequalities. *SIAM Journal on Optimization*, 2000.

[139] Shivashankar Subramanian, Daniel King, Doug Downey, and Sergey Feldman. S2AND: A Benchmark and Evaluation System for Author Name Disambiguation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2021*, 2021.

[140] Chaitanya Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *SODA*, 2004.

[141] Tian Tian, Jie Zhang, Xiang Lin, Zhi Wei, and Hakon Hakonarson. Model-based deep embedding for constrained clustering analysis of single cell rna-seq data. *Nature Communications*, 12, 2021. URL https://api.semanticscholar.org/CorpusID:232365973.

[142] Michael J Todd, Kim-Chuan Toh, and Reha H Tütüncü. On the nesterov–todd direction in semidefinite programming. *SIAM Journal on Optimization*, 1998.

[143] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. Sdpt3—a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 1999.

[144] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Fixed-rank approximation of a positive-semidefinite matrix from streaming data. *Advances in Neural Information Processing Systems*, 2017.

[145] Levent Tunçel. Some applications of semidefinite optimization from an operations research viewpoint. 2009.

[146] Lieven Vandenberghe and Stephen Boyd. Applications of semidefinite programming. *Applied Numerical Mathematics*, 1999.

[147] Lieven Vandenberghe, Stephen Boyd, and Abbas El Gamal. Optimizing dominant time constant in RC circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1998.

[148] Lieven Vandenberghe, Stephen Boyd, and Shao-Po Wu. Determinant maximization with linear matrix inequality constraints. *SIAM journal on matrix analysis and applications*, 1998.

[149] Sharad Vikram and Sanjoy Dasgupta. Interactive bayesian hierarchical clustering. In *International Conference on Machine Learning*, 2016.

[150] Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. Large language models enable few-shot clustering. *arXiv preprint arXiv:2307.00524*, 2023.

[151] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 2000.

[152] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, 2001.

[153] Irene Waldspurger and Alden Waters. Rank optimality for the burer–monteiro factorization. *SIAM journal on Optimization*, 2020.

[154] Jie Wang and Liangbing Hu. Solving low-rank semidefinite programs via manifold optimization. *arXiv preprint arXiv:2303.01722*, 2023.

[155] Kesheng Wu, Andrew Canning, HD Simon, and L-W Wang. Thick-restart lanczos method for electronic structure calculations. *Journal of Computational Physics*, 1999.

[156] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 2002.

[157] Caiming Xiong, David M Johnson, and Jason J Corso. Active clustering with model-based uncertainty reduction. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[158] Linli Xu, Wenye Li, and Dale Schuurmans. Fast normalized cut with linear constraints. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[159] Nishant Yadav, Nicholas Monath, Rico Angell, and Andrew McCallum. 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.

[160] Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. Sdpnal+: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 2015.

[161] Shuai Yuan, Pang-Ning Tan, Kendra Spence Cheruvelil, Sarah M. Collins, and Patricia A. Soranno. Spatially constrained spectral clustering algorithms for region delineation. *ArXiv*, abs/1905.08451, 2019. URL https://api.semanticscholar.org/CorpusID:160010118.

[162] Alp Yurtsever, Quoc Tran Dinh, and Volkan Cevher. A universal primal-dual convex optimization framework. *Advances in Neural Information Processing Systems*, 2015.

[163] Alp Yurtsever, Olivier Fercoq, and Volkan Cevher. A conditional-gradient-based augmented lagrangian framework. In *International Conference on Machine Learning*, 2019.

[164] Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 2021.

[165] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[166] Qing Zhao, Stefan E Karisch, Franz Rendl, and Henry Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 1998.

[167] Yubo Zhuang, Xiaohui Chen, and Yun Yang. Sketch-and-lift: scalable subsampled semidefinite program for k-means clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 9214–9246. PMLR, 2022.