Advances in Sparse and Low Rank Matrix Optimization for Machine Learning Applications

by

Nicholas André G Johnson

B. S. E. Operations Research and Financial Engineering, Princeton University (2020)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Signature of Author .....
Sloan School of Management
August 9, 2024

Certified by .....
Dimitris J. Bertsimas
Boeing Leaders for Global Operations Professor of Management
Associate Dean for Business Analytics
Professor, Operations Research
Thesis Supervisor

Accepted by .....
Georgia Perakis
John C Head III Dean (Interim), MIT Sloan School of Management
Professor, Operations Management, Operations Research & Statistics
CoDirector, Operations Research Center

Advances in Sparse and Low Rank Matrix Optimization for Machine Learning Applications

by

Nicholas André G Johnson

Submitted to the Sloan School of Management
on August 9, 2024 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

ABSTRACT

Numerous fundamental problems in operations research, machine learning, and statistics exhibit natural formulations as cardinality or rank constrained optimization problems. Sparse solutions are desirable for their interpretability and storage benefits. Moreover, in the machine learning setting, sparse solutions exhibit superior model generalization and have a natural interpretation as conducting feature extraction in high-dimensional datasets. On the other hand, since the rank of a matrix is equivalent to the cardinality of the matrix's vector of singular values, rank can be interpreted as the matrix generalization of sparsity. Accordingly, low rank solutions inherit similar desirables properties as sparse solutions while allowing for very flexible modelling capability. Unfortunately, optimizing over cardinality and rank constraints is non-convex and NP-Hard in general which has led to strong reliance on convex relaxations and heuristic methods which yield sub-optimal solutions.

This thesis advances both the theory and application of sparse and low rank matrix optimization, focusing on problems that arise in statistics and machine learning. We develop algorithmic approaches to problems exhibiting cardinality and rank constraints by leveraging techniques from mixed-integer and mixed-projection optimization. The proposed algorithms outperform existing convex relaxations and heuristics. Our rigorous analysis and empirical validation aim to contribute to both the theoretical foundations of optimization and the development of practical tools for complex challenges in statistics and machine learning.

Chapter 2 studies the Sparse Plus Low Rank Matrix Decomposition problem. We present an alternating minimization algorithm that computes high quality feasible solutions and outperforms benchmark methods, scaling to dimension $n = 10000$ in minutes. We additionally design a custom branch and bound algorithm to globally solve problem instances of dimension up to $n = 25$ in minutes. Chapter 3 examines the Compressed Sensing problem, for which we present a custom branch and bound algorithm that can compute globally optimal solutions. Our approach produces solutions that are on average 6.22% more sparse on synthetic data and 9.95% more sparse on real world ECG data when compared to state of the art benchmark approaches. Moreover, our approach outperforms benchmark methods when used as part of a multi-label learning algorithm. Chapter 4 explores the problem of learning a partially observed matrix that is predictive of fully observed side information, which consists of an important generalization of the Matrix Completion problem. We reformulate this problem as a mixed-projection optimization problem and present an alternating

direction method of multipliers algorithm that can solve problems with $n = 10000$ rows and $m = 10000$ columns in less than a minute. On large scale real world data, our algorithm produces solutions that achieves 67% lower out of sample error than benchmark methods in 97% less execution time.

Thesis supervisor: Dimitris J. Bertsimas
Title: Boeing Leaders for Global Operations Professor of Management
Associate Dean for Business Analytics
Professor, Operations Research

Acknowledgments

I would like to first express immense thanks to my adviser Dimitris Bertsimas. I initially met Dimitris during February 2020 before I knew if I had been admitted to MIT. Ever since that first encounter, Dimitris has continually inspired me through his incredible passion for research and through his use of mathematical optimization to solve problems that matter in the world. He has fundamentally shaped my approach to research and challenged me to consistently push myself to grow by exploring new frontiers. I would like to thank Dimitris for providing me with ample research flexibility over the course of my doctorate to explore problem areas of interest (many of which did not lead to positive research results). Beyond research, Dimitris also meaningfully shaped my approach to technology entrepreneurship by distilling key lessons learned from his numerous entrepreneurial pursuits through countless conversations. Most importantly, I would like to express sincere thanks to Dimitris for his steadfast, unwavering support over the course of serious health complications I experienced during my doctoral studies.

I would like to express thanks to Rahul Mazumder and Andy Sun for serving on my thesis committee. It has been a pleasure to work with both of you. Thank you also to Bart Van Parys for serving on my general exam committee. I am deeply appreciative of Georgia Perakis and Patrick Jaillet for their leadership as co-directors of the ORC over the course of my 4 years at MIT, and I am grateful that Laura Rose and Andrew Carvalho always kept the ORC running seamlessly. Thank you to Dimitris, Patrick, Pablo Parrilo, David Gamarnik, Alexandre Jacquillat, Lorenzo Rosasco, Tomaso Poggio, Dick den Hertog, Martin Wainright

and Sasha Rakhlin for engaging coursework.

During my time at MIT, I have been fortunate to collaborate with exceptional peers who significantly influenced my PhD experience. Accordingly, I would like to express thanks in particular to Ryan Cory-Wright, Michael Li, Alex Paskov and Wes Gurnee. I am moreover immensely grateful for the many great conversations, laughs and friendship shared with other students at the ORC. Among many others, my time at the ORC was made better by Prem Talwai, El Ghali Zerhouni, Ivan Paskov, Arthur Delarue, Kim Villalobos Carballo, Vassilis Digalakis Jr., Angelos Koulouras and Baptiste Rossi.

I would also like to express deep thanks to my friends from outside of the ORC who have supported and inspired me through these past 4 years. I am particularly grateful for Jadal Williams, Aaron Michael West Jr., Junior Pena, Myles Sampson, Clyde-Blaise Niba, Menelik Graham, William Pugh, Todd Baldwin Jr., Oluwatomi Lawal, Angel Onuoha, Milliam Gehrer, Yiwen Li, Jesse Thibodeau, Alexander Sinora, Perry Chuinkam and Jean-Pierre Ngezigihe.

My undergraduate advisers and mentors were instrumental in preparing me for my MIT journey and beyond. I would like to express special thanks to William Massey for encouraging me to pursue PhD studies in OR and to Miklos Racz for providing me with exceptional guidance during my undergraduate thesis work. Ronnie Sircar, Ludovic Tangpi, Amir Ali Ahmadi and Jianqing Fan were also instrumental in cultivating my interest in the field of OR. I would also like to thank several important people from my time at Princeton outside of the ORFE department: Momo Wolapaye, Martin Semelhack, Ilhan Aksay, Dannelle Gutarra Cordero, Alexis Andres and Sandra Bermann.

My 12 years at Selwyn House School in Montreal were unforgettable and instrumental in preparing me for life. I am immensely grateful to the entirety of the Selwyn House community. Special thanks to Hal Hannaford, Michael Downey, Carol Manning and Kathy Funamoto for their leadership during my tenure. Special thanks also to David Grier, Catherine Lumsden, Samara Sayegh and Pat Shannon for cultivating my interest in math, science

and the arts. Thanks to George Levtchouk and Lefong Hua for helping me explore my passion for chess, and thank you to Mrs. Edelmera Harrison for pushing me to compete in math competitions.

Finally, I would like to express my deepest thanks for my family, to whom I am dedicating this thesis. To my late maternal grandmother Gloria and maternal grandfather Cyril, thank you for instilling in me the value of education and the importance of ownership. To my paternal grandparents Mavis and George, thank you for continually reminding me that through Christ all things are possible and to remain disciplined. To my sister Anastasia, thank you for instilling in me the confidence to unapologetically pursue my passions and strive for excellence in all that I do. To my parents Anita and Dexter, thank you for imparting the values of hard work, perseverance and accountability onto me. Thank you both for trusting and supporting me on this 8 year adventure in a foreign country. Words cannot express how grateful I am for everything that you have given me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Numerous central problems from the operations research, machine learning, and statistics literatures exhibit natural formulations as cardinality and/or rank constrained optimization problems. Among others, sparse regression/classification, sparse principal component analysis and sparse inverse covariance matrix estimation are canonical statistics and machine learning problems that exhibit cardinality constraints to reflect model parsimony. Furthermore, fundamental problems such as factor analysis, low rank kernel learning and optimal control contain explicit rank constraints to enforce low complexity or low dimensionality.

Sparse solutions to optimization problems are desirable due to their increased interpretability and storage benefits. Moreover, in the machine learning setting, sparse solutions exhibit superior model generalization properties and have a natural interpretation as conducting feature extraction for high-dimensional datasets. On the other hand, since the rank of a matrix is equivalent to the cardinality of the matrix's vector of singular values, rank can be interpreted as the matrix generalization of sparsity. Accordingly, low rank solutions inherit the previously mentioned desirable properties of sparse solutions. Furthermore, rank constraints allow for very flexible modelling power. For example, any non-convex quadratic optimization problem can be equivalently reformulated as a convex optimization problem intersected with a rank 1 constraint. This framework captures all binary quadratic opti-

mization problems [72] as well as problems that involve optimizing the dynamics of certain physicals systems like power grids through the alternating current optimal power flow problem [96].

Techniques for sparse optimization have been well studied by the mixed-integer optimization community. Most approaches reformulate cardinality constrained optimization problems as binary optimization problems with "big-M" constraints and obtain globally optimal solutions via branch and bound, branch and cut or a similar algorithmic framework. Despite recent efforts [22], the machine learning community has yet to widely adopt such approaches due to poor scaling of their worst case asymptotic computational complexity. This, however, is a poor reason to discount these methods, since average case wall clock runtime is in general a far more relevant measure of computational complexity than worst case asymptotic scaling. On the other hand, although rank constrained optimization has previously been regarded as intractable by the optimization and machine learning communities in part because it cannot be modelled using mixed-integer convex optimization [100], recent efforts have made substantial progress in obtaining tractable reformulations of rank constraints [16, 19, 20].

In this thesis, we make theoretical and applied contributions to further the state of sparse and low rank matrix optimization with a specific focus on statistics and machine learning problems. We continue to bridge the gap between the mixed-integer optimization literature and the machine learning literature by leveraging techniques from binary optimization to tackle machine learning problems that previously have been solved using approximate techniques. Whereas cardinality constraints and rank constraints have previously been treated independently by the global optimization community, we tackle problems that simultaneous enforce both constraints and develop methods that outperform existing convex relaxations and heuristics. Through rigorous analysis and empirical validation, the proposed work aims to contribute not only to the theoretical foundations of optimization but also to the development of practical tools that can be applied to address complex challenges in statistics and machine learning.

In the remainder of this chapter, we provide an overview of the structure of this thesis while highlighting the core contributions of each chapter. We also introduce the notation we use throughout this thesis.

1.1 Thesis Structure and Contributions

Chapter 2 In this chapter, we consider the *Sparse Plus Low Rank* (SLR) decomposition problem which is the problem of approximately decomposing a corrupted data matrix $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ into a sparse matrix $\boldsymbol{Y}$ that models the data corruptions of $\boldsymbol{D}$ plus a low-rank matrix $\boldsymbol{X}$ that models the uncorrupted principal components of $\boldsymbol{D}$.

Formally, given a target rank k_0 and a target sparsity level k_1, we solve:

$$\min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2 \quad \text{s.t. } \mathrm{Rank}(\boldsymbol{X}) \le k_0, \|\boldsymbol{Y}\|_0 \le k_1, \tag{1.1}$$

where $\lambda, \mu > 0$ are parameters that control sensitivity to noise and are to be cross-validated by minimizing a validation metric to obtain strong out-of-sample performance. Our formulation (1.1) is well-justified from an information-theoretic perspective. Indeed, several authors [2, 67] have demonstrated that, for special cases of Problem (1.1), when the ground truth is sparse and/or low-rank, exact sparse and/or low-rank formulations recover the ground truth at least as accurately as any polynomial time method, and indeed there is a gap between the amount of data required for an "exact" sparse plus low-rank formulation to recover the ground truth, and the amount of data required for a polynomial time approach [67].

A key characteristic of Problem (1.1) is that it directly employs a sparsity constraint on $\boldsymbol{Y}$ and a rank constraint on $\boldsymbol{X}$. These constraints are non-convex, which make (1.1) a difficult problem to solve exactly, both in practice—where the best-known exact algorithms cannot certify optimality beyond $n = 10$ [97]—and in theory, where the problem is NP-hard by reduction from low-rank matrix approximation [70].

In this chapter, we present three main contributions.

- First, we introduce a novel formulation (1.1) for the SLR decomposition problem that directly exploits the underlying discreteness of the problem. Our formulation is inspired by incorporating robustness against adversarial perturbations in the input data in SLR, which is useful in noisy settings.

- Our second main contribution is the development of a heuristic that obtains high quality feasible solutions to Problem (1.1) and the derivation of a convex relaxation of (1.1) that provides high quality bounds for the solutions returned by our heuristic. Our relaxation can be interpreted as a novel reverse Huber penalty which penalizes the sparse and low-rank matrices in a convex manner. Moreover, we present a branch and bound framework that solves (1.1) to certifiable near-optimality for small problem instances.

- Third, we extensively benchmark our proposed algorithms and demonstrate that they outperform state-of-the-art methods by obtaining sparser and lower rank matrices with a lower mean-squared error than via prior attempts, in a comparable amount of computational time. Moreover, our approach scales to successfully solve problem instances with 10000×10000 matrices.

The work in this chapter is based on the paper "Sparse Plus Low Rank Matrix Decomposition: A Discrete Optimization Approach" [15], which is joint work with Dimitris Bertsimas and Ryan Cory-Wright.

Chapter 3 We consider the *Compressed Sensing* (CS) problem in this chapter, a fundamental problem in statistics and machine learning which arises in numerous applications such as medical resonance imaging, holography, climate monitoring, natural resource mining and electrocardiogram signal acquisition among many others. CS seeks to find the sparsest vector $\boldsymbol{x} \in \mathbb{R}^n$ that is consistent with a collection of m linear equalities.

Formally, given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^m$, CS is given by [62]:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \|\boldsymbol{x}\|_0 \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \tag{1.2}$$

In the presence of noisy measurements, it is necessary to relax the equality constraint in (1.2), leading to the following formulation for $\epsilon > 0$:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \|\boldsymbol{x}\|_0 \quad \text{s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon. \tag{1.3}$$

The vast majority of existing approaches to CS either rely on ℓ_1 based convex approximations to (1.3) or are greedy heuristics whereas the use of integer optimization techniques has gone relatively unexplored. In this chapter, we formulate CS as:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \|\boldsymbol{x}\|_0 + \frac{1}{\gamma}\|\boldsymbol{x}\|_2^2 \quad \text{s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon, \tag{1.4}$$

where $\gamma > 0$ is a regularization parameter that in practice can either take a default value (e.g. $\sqrt{n}$) or be cross-validated by minimizing a validation metric to obtain strong out-of-sample performance. A defining characteristic of the approach we present in this work is that we leverage techniques from mixed-integer second order cone optimization to exploit the inherent discreteness of formulation (1.4) rather than relying on more commonly studied approximate methods.

In this chapter, we present three main contributions.

- First, we derive a second order cone relaxation of (1.4) and show that under mild conditions on the regularization parameter, the resulting relaxation is equivalent to the well studied basis pursuit denoising problem and the closely related Lasso problem [131].

- Second, we present a semidefinite relaxation that strengthens our second order cone relaxation and develop a custom branch and bound algorithm that leverages our second

order cone relaxation to solve instances of CS to certifiable optimality.

- Third, we benchmark our approach extensively on both synthetic and real world data. When compared against solutions produced by three state of the art benchmark methods on synthetic data, our numerical results show that our approach produces solutions that are on average 6.22% more sparse. On real world ECG data, for a given ℓ_2 reconstruction error our approach produces solutions that are on average 9.95% more sparse than benchmark methods, while for a given sparsity level our approach produces solutions that have on average 10.77% lower reconstruction error than benchmark methods. When used as a component of a multi-label classification algorithm, our approach achieves greater classification accuracy than benchmark compressed sensing methods.

The work in this chapter is based on the paper "Compressed Sensing: A Discrete Optimization Approach" [24], authored with Dimitris Bertsimas.

Chapter 4 In this Chapter, we study the problem of learning a partially observed matrix under the low rank assumption in the presence of fully observed side information that depends linearly on the true underlying matrix. This problem consists of an important generalization of the Matrix Completion problem that arises in applications such as recommendation systems, signal processing, system identification and image denoising. We formalize this problem as an optimization problem with an objective that balances the strength of the fit of the reconstruction to the observed entries with the ability of the reconstruction to be predictive of the side information.

Formally, let $\Omega \subseteq [n] \times [m]$ denote a collection of revealed entries of a partially observed matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, let $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ denote a matrix of side information and let k denote a specified target rank. We consider the problem given by

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times m}, \boldsymbol{\alpha} \in \mathbb{R}^{m \times d}} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 + \lambda \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\alpha}\|_F^2 + \gamma \|\boldsymbol{X}\|_\star \quad \text{s.t.} \quad \text{rank}(\boldsymbol{X}) \leq k, \tag{1.5}$$

where $\lambda, \gamma > 0$ are hyperparameters that in practice can either take a default value or can be cross-validated by minimizing a validation metric [114] to obtain strong out-of-sample performance [34]. We assume that the ground truth matrix $\boldsymbol{A}$ has low rank and that the side information can be well approximated as $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\alpha} + \boldsymbol{N}$ for some weighting matrix $\boldsymbol{\alpha}$ and noise matrix $\boldsymbol{N}$.

In this chapter, we present three main contributions.

- First, we derive a mixed-projection reformulation of (1.5) and present a strong semidefinite cone relaxation.

- Second, we design an efficient, scalable alternating direction method of multipliers algorithm that produces high quality feasible solutions to (1.5).

- Third, we benchmark our approach extensively on synthetic and real world data. Our numerical results demonstrate that on synthetic data in the small rank regime ($k \leq 15$), our algorithm outputs solutions that achieve on average 79% lower objective value and 90.1% lower ℓ_2 reconstruction error than the solutions returned by the best performing benchmark method on a per experiment basis. The runtime of our algorithm is competitive with and often superior to that of the benchmark methods. Our algorithm is able to solve problems with $n = 10000$ rows and $m = 10000$ columns in less than a minute. On large scale real world data, our algorithm produces solutions that achieves 67% lower out of sample error than benchmark methods in 97% less execution time.

The work in this chapter is based on the preprint "Predictive Low Rank Matrix Learning under Partial Observations: Mixed-Projection ADMM" [25], authored with Dimitris Bertsimas.

1.2 Notation

We let nonbold face characters such as b denote scalars, lowercase bold faced characters such as $\boldsymbol{x}$ denote vectors, uppercase bold faced characters such as $\boldsymbol{X}$ denote matrices, and calligraphic uppercase characters such as $\mathcal{Z}$ denote sets. We let $[n]$ denote the set of running indices $\{1, ..., n\}$ and $\langle \cdot, \cdot \rangle$ denote the Euclidean (Frobenius) inner product between two vectors (matrices) of the same dimension. We let $\mathbf{0}_n$ denote an n-dimensional vector of all 0's, $\mathbf{0}_{n \times m}$ denote an $n \times m$-dimensional matrix of all 0's, and $\boldsymbol{I}_n$ denote the $n \times n$ identity matrix. We let $\mathcal{S}^n$ denote the cone of $n \times n$ symmetric matrices and $\mathcal{S}^n_+$ denote the cone of $n \times n$ positive semidefinite matrices.

Chapter 2

Sparse Plus Low Rank Matrix Decomposition: A Discrete Optimization Approach

The work in this chapter is based on [15] which is joint work with Dimitris Bertsimas and Ryan Cory-Wright.

2.1 Introduction

The *Sparse Plus Low Rank* (SLR) decomposition problem, or the problem of approximately decomposing a data matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ into a sparse matrix $\boldsymbol{Y}$ plus a low-rank matrix $\boldsymbol{X}$, arises throughout many fundamental applications in Operations Research, Machine Learning, and Statistics, including collaborative filtering [122], medical resonance imaging [51], and economic modeling [8] among others. Formally, given a target rank k_0 and a target sparsity k_1, we solve:

$$\min_{\boldsymbol{X},\boldsymbol{Y} \in \mathbb{R}^{n \times n}} \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda \|\boldsymbol{X}\|_F^2 + \mu \|\boldsymbol{Y}\|_F^2 \text{ s.t. } \text{Rank}(\boldsymbol{X}) \leq k_0, \|\boldsymbol{Y}\|_0 \leq k_1, \tag{2.1}$$

where $\lambda, \mu > 0$ are parameters that control sensitivity to noise and are to be cross-validated by minimizing a validation metric [see, e.g., 114] to obtain strong out-of-sample performance in theory and practice [34].

In SLR decomposition problems, the sparse matrix $\boldsymbol{Y}$ accounts for a small number of potentially large corruptions in $\boldsymbol{D}$, while $\boldsymbol{X}$ models the leading principal components of $\boldsymbol{D}$ after this corruption is removed. This is well justified, because SLR robustifies Principal Component Analysis (PCA), a leading technique for finding low-rank approximations of noiseless datasets [116], which performs poorly in high-dimensional settings and in the presence of noise [108]. In an opposite direction, SLR robustly accounts for noise via the sparse matrix $\boldsymbol{Y}$, while $\boldsymbol{X}$ recovers the uncorrupted principal component directions of $\boldsymbol{D}$. Correspondingly, SLR decomposition schemes, which are also called Robust PCA since at least the work of [44], are widely regarded as state-of-the-art approaches for high-dimensional matrix estimation problems [50, 108].

Our formulation (2.1) is also well-justified from an information-theoretic perspective. Indeed, several authors [2, 67] have demonstrated for special cases of Problem (2.1) that when the ground truth is sparse and/or low-rank, exact sparse and/or low-rank formulations

recover the ground truth at least as accurately as any polynomial time method, and indeed there is a gap between the amount of data required for an "exact" sparse plus low-rank formulation to recover the ground truth, and the amount of data required for a polynomial time approach [67].

A key characteristic of Problem (2.1) is that it directly employs a sparsity constraint on $\boldsymbol{Y}$ and a rank constraint on $\boldsymbol{X}$. These constraints are non-convex, which make (2.1) a difficult problem to solve exactly, both in practice—where the best-known exact algorithms cannot certify optimality beyond $n = 10$ [97]—and in theory, where the problem is NP-hard by reduction from low-rank matrix approximation [70].

In this chapter, we develop an alternating minimization heuristic and convex relaxation which collectively provide very small bound gaps for (2.1) and scale to high-dimensional settings. Our heuristic scales to $n = 10000$ in minutes and our convex relaxation scales to $n = 200$ in hours. A key feature of the approach is that it leverages the underlying discreteness of the problem to obtain tight yet computationally cheap lower bounds. We further demonstrate that the alternating minimization heuristic and convex relaxation can be embedded within a branch and bound tree to solve (2.1) to certifiable near-optimality for instances of size up to $n = 25$.

2.1.1 Contribution and Structure

The key contributions of this chapter are threefold:

- First, from a methodological perspective, we introduce a novel formulation (2.1) for the SLR decomposition problem that directly exploits the underlying discreteness of the problem. Our formulation is inspired by incorporating robustness against adversarial perturbations in the input data in SLR, which is useful in noisy settings.

- Second, from an algorithmic perspective, we develop a heuristic that obtains high quality feasible solutions to Problem (2.1) in Section 2.3 and derive a convex relaxation

of (2.1) that provides high quality bounds for the solutions returned by our heuristic in Section 2.4. We also interpret the convex relaxation as a novel reverse Huber penalty which penalizes the sparse and low-rank matrices in a convex manner. Further, we present a branch and bound framework that solves (2.1) to certifiable near-optimality for small problem instances in Section 2.5.

- Third, from a computational perspective, we extensively benchmark our proposed approach. Across a suite of numerical experiments, we demonstrate in Section 2.6 that our approach outperforms state-of-the-art non-convex methods like AccAltProj, GoDec and ScaledGD by obtaining sparser and lower rank matrices with a lower mean-squared error than via prior attempts, in a comparable amount of computational time. Moreover, our approach scales to successfully solve problem instances with 10000×10000 matrices.

2.1.2 Applications

We briefly overview several applications of the SLR matrix decomposition problem.

Medical Imaging Medical Resonance Imaging (MRI) is a medical imaging technique in which images of structures and organs in the human body are obtained using strong magnetic fields and radio waves. Dynamic MRI is a technique that allows for the imaging of functional changes of structures in the body over time, for instance the beating of a heart. A dynamic MRI is performed by repeatedly sampling spatial frequencies and reconstructing the final output through a Fourier Transform. It has been shown that SLR decomposition can facilitate the reconstruction process where output low rank matrix consists of the static background of the structure being imaged and the sparse matrix consists of time varying features [51] [112]. Having high quality, scalable algorithms for SLR is then helpful to reduce the number of spatial frequencies needed to be sampled, thereby reducing imaging time, and to improve image resolution.

Economic Modeling Many problems in economics and finance require modeling high dimensional time series for the purposes of forecasting and policy making. When vector autoregression is employed to this end and causal interactions are sought to be made between observed time series, it has been shown that SLR can effectively approximate the transition matrix of the model in the presence of a small number of unobserved latent factors [8].

Video Surveillance When performing video surveillance, the static background of a surveyed area must be separated from the fluid foreground. Given a series of frames, if we stack the flattened frames into a matrix and perform SLR, the returned low rank matrix will correspond to the background while the sparse matrix will correspond to objects that dynamically move along the background.

Facial Recognition It has been shown that the set of all images of a Lambertian surface under varying illuminations spans a low dimensional subspace [7]. In particular, this implies that the set of images of an individual's face under varying sources of lightning can be approximated by a low dimensional subspace. Accordingly, given a series of images of an individual's face, the images can be flattened and stacked into a matrix. Applying SLR, the output sparse matrix would consist of the image by image variations resulting from different sources of lightning while the output low rank matrix would extract the underlying features on the individual's face.

2.2 Literature Review and SLR Formulation Properties

In this section, we judiciously characterize Problem (2.1) and state-of-the-art approaches for addressing it. First, in Section 2.2.1, we cast a deliberate eye over existing attempts at solving Problem (2.1) that are currently considered to be state-of-the-art and establish that these approaches are either heuristics that do not provide performance guarantees or branch and bound methods that do not scale to even moderate problem sizes. Next, in Section 2.2.2,

we establish several key properties of Problem (2.1)'s objective function that we invoke throughout this chapter. Further, in Section 2.2.3, we justify the regularization terms in our formulation by interpreting our formulation through the lens of robust optimization. Finally, in Section 2.2.4, we characterize the conditions under which Problem (2.1) admits a reduction to matrix completion, a famous and frequently studied cousin of Problem (2.1) which is notoriously computationally challenging [45].

2.2.1 Literature Review

In this section, we selectively review several formulations from the literature that have been employed to solve the sparse plus low-rank decomposition problem and are currently considered to be state-of-the-art. Most of these approaches are heuristic in nature and do not provide valid lower bounds to certify the (sub) optimality of the output solution.

Stable Principal Component Pursuit

Optimizing over low-rank matrices is notoriously computationally challenging in both theory and practice [19, 122]. Accordingly, a popular approach is to replace the rank and sparsity terms with their nuclear norm and ℓ_1 norm surrogates, as advocated by [44, 50] among others. In the presence of noise, this substitution leads to the following formulation, which was originally proposed by [155] and is called Stable Principal Component Pursuit (S-PCP):

$$\min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad \|\boldsymbol{X}\|_* + \frac{1}{\sqrt{n}}\|\boldsymbol{Y}\|_1 + \frac{1}{2\mu}\|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2. \tag{2.2}$$

Problem (2.2) can either be reformulated as a semidefinite problem over a $2n \times 2n$ matrix as advocated by [44], solved in the original space using a nonsymmetric interior point method as proposed by [128] or solved in a semidefinite free fashion using an augmented Lagrangian approach as advocated by Yuan und Yang [151]. Unfortunately, all three approaches require repeatedly performing operations such as a singular value decomposition or a Newton step,

which has an $O(n^3)$ or higher time/memory cost. Correspondingly, all such semidefinite optimization approaches require too much memory to be successfully implemented in a standard computational environment when $n = 200$, at least with current technology [see 103, for a review of the state-of-the-art in semidefinite optimization]. Moreover, these methods are usually only guaranteed to recover a ground truth model under a mutual incoherence condition (or similar) on the ground truth [see 132, for a review], which implies that performance guarantees for such semidefinite methods are challenging to obtain indeed.

GoDec

Many existing formulations for SLR employ convex relaxations of the rank function and the ℓ_0 norm function rather than exploiting the inherent discreteness of the problem. An exception to this pattern is the work of [154], who leverage discreteness to obtain higher quality solutions to SLR. Their formulation is given by:

$$\min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 \text{ s.t. } \text{Rank}(\boldsymbol{X}) \leq k_0, \|\boldsymbol{Y}\|_0 \leq k_1. \tag{2.3}$$

Note that (2.3) differs from (2.1) by the absence of regularization terms on $\boldsymbol{X}$ and $\boldsymbol{Y}$. [154] obtain a feasible solution to (2.3) by performing alternating minimization on $\boldsymbol{X}$, $\boldsymbol{Y}$. Their algorithm, called GoDec, is similar in structure to the algorithm we develop in Section 2.3 to obtain high quality solutions to Problem (2.1). In a related direction, [149] adopt a similar approach to GoDec in the special case where their design matrix is taken to be the identity. [91] adopt a similar formulation as GoDec, however, they instead minimize the reconstruction error between an observation vector and a vector-valued linear map of the sum of the low-rank and sparse matrices. In a somewhat different vein, [152] consider an explicit rank constraint but not a sparsity constraint and proceed by leveraging manifold optimization techniques.

Low Rank Matrix Parameterization

An extensively studied family of methods parameterizes the low-rank matrix $\boldsymbol{X}$ as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{n \times k_0}$, and performs alternating minimization on $\boldsymbol{U}, \boldsymbol{V}$. Originally proposed in the context of low-rank semidefinite optimization by [39, 40] [see also 84], it has since evolved into an extensively used and practical approach for SLR problems [41, 56, 75, 110]. This approach eliminates the rank constraint and can substantially reduce the number of variables when $n \ll k_0$ at the expense of introducing non-convexity in the objective. Remarkably, in many circumstances, the induced non-convexity is benign and the resulting Burer-Monteiro reformulation can be solved efficiently from both a theoretical and a practical perspective. We refer readers to [57] for a detailed overview.

Two important parametrization-based approaches to SLR are Fast RPCA [150] and Scaled Gradient Descent [133]. In Fast RPCA, after parametrizing the low-rank matrix, [150] augment the objective with a regularization term on the norm of $(\boldsymbol{U}^T\boldsymbol{U} - \boldsymbol{V}^T\boldsymbol{V}) \in \mathbb{R}^{k_0 \times k_0}$ before performing alternating minimization on $\boldsymbol{U}$ and $\boldsymbol{V}$. In an alternate direction, [133] performs iterative gradient descent updates on $\boldsymbol{U}$ and $\boldsymbol{V}$ in Scaled Gradient Descent after designing an effective gradient preconditioner that results in desirable convergence behavior even for ill-conditioned problems. However, existing performance guarantees for these approaches rely on assumptions on the structure of the ground truth, such as mutual incoherence, that are difficult to verify without independent access to the ground truth or on being initialized within a "basin of attraction" which similarly is difficult to verify. We point out, however, that one could either use the dual bounds derived in this chapter, or side information such as scoring by humans (e.g., in video background separation applications) to provide performance guarantees when the ground truth is not known.

Branch and Bound

To our knowledge, the only existing work that provides guarantees on the quality of solutions to Problem (2.1) is [97], who propose a branch and bound algorithm for solving Problem

(2.1) to near-optimality. Specifically, they assume that the spectral norm of $\boldsymbol{X}$ is bounded from above by β, i.e., $\beta \geq \|\boldsymbol{X}\|_\sigma$, and invoke the following inequality to obtain valid lower bounds for each partially specified sparsity pattern [see also 66]:

$$\frac{\gamma}{\alpha}\|\boldsymbol{Y}\|_1 + \frac{1}{\beta}\|\boldsymbol{X}\|_* \leq \gamma\|\boldsymbol{Y}\|_0 + \text{Rank}(\boldsymbol{X}), \tag{2.4}$$

where $\alpha \geq \|\boldsymbol{Y}\|_\infty$ is a bound on the ℓ_∞ norm of $\boldsymbol{Y}$, which can either be taken to be equal to some large fixed constant M [71] or treated as a regularization parameter [17]. Unfortunately, while Lee und Zou [97]'s bound is often reasonable, it was not developed by taking the convex envelope of an appropriate substructure of Problem (2.1), and therefore is not strong enough to solve Problem (2.1) to optimality at even small problem sizes [see also 31, for a related discussion on the weakness of big-M bounds]. Indeed, the authors reported bound gaps but not optimal solutions for SLR problems when $n = 10$. Nonetheless, this lower bound is potentially interesting in its own right, since it demonstrates that the PCP formulation supplies a valid lower bound on Problem (2.1) if one is willing to either make a big-M assumption on the spectral norm of the low-rank matrix or compute a valid M [c.f. 19, Section 3.5].

2.2.2 Objective Function Properties

We now derive several key properties of Problem (2.1) that we leverage throughout this chapter and present a probabilistic interpretation of (2.1) which is motivated by Bayesian inference. Specifically, we establish that (2.1)'s objective is strongly convex, Lipschitz continuous, and the Maximum A Posteriori (MAP) estimator of a suitably defined probabilistic model under a Gaussian prior. Recall that a function $f(\boldsymbol{Z})$ is said to be strongly convex with parameter m (m-strongly convex) if the function $f(\boldsymbol{Z}) - \frac{m}{2}\|\boldsymbol{Z}\|_F^2$ is convex. Similarly, a function $f(\boldsymbol{Z})$ is said to be Lipschitz continuous with constant L (L-Lipschitz) if the function $\frac{L}{2}\|\boldsymbol{Z}\|_F^2 - f(\boldsymbol{Z})$ is convex. Formally, we have the following results (proofs deferred to Section

2.8):

Proposition 1 *The function* $f(\boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2$ *is jointly* m*-strongly convex in* $(\boldsymbol{X}, \boldsymbol{Y})$ *over* $\mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$*, i.e.,* $g(\boldsymbol{X}, \boldsymbol{Y}) = f(\boldsymbol{X}, \boldsymbol{Y}) - \frac{m}{2}(\|\boldsymbol{X}\|_F^2 + \|\boldsymbol{Y}\|_F^2)$ *is jointly convex in* $(\boldsymbol{X}, \boldsymbol{Y})$*, for* $m = 2 \cdot \min(\lambda, \mu)$*.*

Proposition 2 *The function* $f(\boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2$ *is* L*-Lipschitz continuous in* $(\boldsymbol{X}, \boldsymbol{Y})$ *over* $\mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$ *for* $L = 2 \cdot \max(\lambda, \mu) + 6$*.*

Note that Propositions 1–2 collectively imply that the condition number κ of $f(\boldsymbol{X}, \boldsymbol{Y})$ is

$$\kappa = \frac{L}{m} = \frac{2 \cdot \max(\lambda, \mu) + 6}{2 \cdot \min(\lambda, \mu)}. \tag{2.5}$$

We now provide a probabilistic interpretation of $f(\boldsymbol{X}, \boldsymbol{Y})$. Suppose the data $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ are sampled from

$$\boldsymbol{D} = \boldsymbol{X} + \boldsymbol{Y} + \boldsymbol{\epsilon}, \tag{2.6}$$

where $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times n}$ are unknown parameters to be estimated and $\boldsymbol{\epsilon} \in \mathbb{R}^{n\times n}$, $\epsilon_{ij} \sim N(0, \sigma^2)$ is i.i.d Gaussian noise with variance σ^2. If we adopt independent Gaussian prior beliefs $X_{ij} \sim N(0, \frac{\sigma^2}{\lambda})$ and $Y_{ij} \sim N(0, \frac{\sigma^2}{\mu})$ over the parameters $\boldsymbol{X}, \boldsymbol{Y}$, then the Maximum A Posteriori (MAP) estimate of $\boldsymbol{X}, \boldsymbol{Y}$ after observing $\boldsymbol{D}$ is given by $\arg\min_{\boldsymbol{X},\boldsymbol{Y}} f(\boldsymbol{X}, \boldsymbol{Y})$.

To see this, note that the posterior probability after observing $\boldsymbol{D}$ is given by

$$\mathbf{P}(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{D}) = \frac{\boldsymbol{P}(\boldsymbol{D}|\boldsymbol{X}, \boldsymbol{Y})\mathbf{P}(\boldsymbol{X})\mathbf{P}(\boldsymbol{Y})}{\mathbf{P}(\boldsymbol{D})} \propto \boldsymbol{P}(\boldsymbol{D}|\boldsymbol{X}, \boldsymbol{Y})\mathbf{P}(\boldsymbol{X})\mathbf{P}(\boldsymbol{Y}). \tag{2.7}$$

We can now obtain the MAP estimate by maximizing the posterior probability as follows

$$\begin{aligned}\arg\max_{\boldsymbol{X},\boldsymbol{Y}} \boldsymbol{P}(\boldsymbol{D}|\boldsymbol{X}, \boldsymbol{Y})\mathbf{P}(\boldsymbol{X})\mathbf{P}(\boldsymbol{Y}) &= \arg\max_{\boldsymbol{X},\boldsymbol{Y}} \prod_{1\le i,j\le n} \frac{e^{\frac{-(D_{ij}-X_{ij}-Y_{ij})^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \cdot \frac{\sqrt{\lambda}e^{\frac{-\lambda X_{ij}^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \cdot \frac{\sqrt{\mu}e^{\frac{-\mu Y_{ij}^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \\ &= \arg\min_{\boldsymbol{X},\boldsymbol{Y}} \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2 = \arg\min_{\boldsymbol{X},\boldsymbol{Y}} f(\boldsymbol{X}, \boldsymbol{Y})\end{aligned}$$

where the second equality follows by taking a log transformation and multiplying by $-2\sigma^2$.

2.2.3 Equivalence Between Regularization and Robustness

Real-world datasets are replete with inaccurate and missing data values, which prevents machine-learning models that do not account for these inconsistencies from generalizing well to unseen data. Accordingly, robustness is a highly desirable attribute for machine learning models, in both theory and practice [23, 145]. In this section, we demonstrate that our regularized problem (2.1) is equivalent to a robust optimization (RO) problem. This result motivates the inclusion of the Frobenius regularization terms within (2.1) and verifies that (assuming the hyperparameters in (2.1) are correctly cross-validated), regularization improves (2.1)'s out-of-sample performance.

We remark that our results should not be too surprising to readers familiar with the RO literature. Indeed, [13] have already derived a similar result for regularized linear regression problems. However, our main result is strictly more general. Indeed, [13] prove that augmenting an ℓ_2 loss function with an ℓ_2 regularization penalty is equivalent to solving a RO problem, and conjecture (but do not prove) that their result can be extended to ordinary least squares regression and ridge regularization (with ℓ_2^2 rather than ℓ_2 penalties). On the other hand, we prove a matrix analog of their result and generalize their result to the matrix analog of ℓ_2^2 regularization. Accordingly, this section may be of independent interest to the RO community.

We now connect our work with the work of [13] by deriving a conceptually simple analog of their characterization of the equivalence of regularization and robustness for sparse plus low-rank problems. This result sheds insight into the nature of regularization as a robustifying force in Problem (2.1). Subsequently, we derive an (admittedly more opaque) characterization of Problem (2.1) itself as a RO problem.

Formally, we have the following results (proofs deferred to Section 2.8):

Proposition 3 *Let* $\mathcal{U}_\lambda(\boldsymbol{X}) = \{\boldsymbol{\Delta} \in \mathbb{R}^{n\times n} : \|\boldsymbol{\Delta}\|_F \leq \lambda\|\boldsymbol{X}\|_F\}$ *for* $\boldsymbol{X} \in \mathbb{R}^{n\times n}, \lambda > 0$. *Consider the robust optimization problem:*

$$\min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \max_{\substack{\boldsymbol{\Delta}_1\in\mathcal{U}_\lambda(\boldsymbol{X})\\ \boldsymbol{\Delta}_2\in\mathcal{U}_\mu(\boldsymbol{Y})}} \|\boldsymbol{D}+\boldsymbol{\Delta_1}+\boldsymbol{\Delta_2}-\boldsymbol{X}-\boldsymbol{Y}\|_F \text{ s.t. } \boldsymbol{X}\in\mathcal{V}, \boldsymbol{Y}\in\mathcal{W}, \tag{2.8}$$

where $\mathcal{V}$ *and* $\mathcal{W}$ *are arbitrary subsets of* $\mathbb{R}^{n\times n}$. *Then,* (2.8) *is equivalent to* (2.9).

$$\min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F + \lambda\|\boldsymbol{X}\|_F + \mu\|\boldsymbol{Y}\|_F \text{ s.t. } \boldsymbol{X}\in\mathcal{V}, \boldsymbol{Y}\in\mathcal{W}. \tag{2.9}$$

Proposition 4 *Problem* (2.1) *is equivalent to the following robust optimization problem:*

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \max_{\boldsymbol{\Delta}_1,\boldsymbol{\Delta}_2} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \langle\boldsymbol{X},\boldsymbol{\Delta_1}\rangle + \langle\boldsymbol{Y},\boldsymbol{\Delta}_2\rangle - \frac{1}{4\lambda}\|\boldsymbol{\Delta}_1\|_F^2 - \frac{1}{4\mu}\|\boldsymbol{\Delta}_2\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{X}\in\mathcal{V}, \boldsymbol{Y}\in\mathcal{W}. \end{aligned} \tag{2.10}$$

Taking $\mathcal{V}$ to be the set of matrices with rank at most k_0 and $\mathcal{W}$ to be the set of matrices with ℓ_0 norm at most k_1, Proposition 3 implies that performing SLR decomposition with Frobenius regularization is equivalent to solving a RO problem that allows for adversarial errors in the input data matrix $\boldsymbol{D}$. Moreover, Proposition 4 implies that solving Problem (2.1) is equivalent to solving a RO problem with a soft robust penalty term in the objective, rather than a hard constraint on the size of the uncertainty set, as such robust equivalent problems usually consist of. This result is perhaps unsurprising in retrospect, since dual problems to quadratically constrained quadratic problems involve quadratic terms in the objective [see also 124, Section 6.3].

2.2.4 Connection to Matrix Completion

Low-rank matrix completion is a canonical problem in the Statistics and Machine Learning communities that has been employed in control theory [36], computer vision [45], and signal processing [85] among other applications. Given a partially observed matrix $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ where $\Omega \subset \{(i,j) : 1 \leq i,j \leq n\}$ denotes the set of indices of the revealed entries, the

low-rank matrix completion problem is to compute a low-rank matrix $\boldsymbol{X}$ that approximates $\boldsymbol{D}$. Low-rank matrix completion solves

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times n}} \quad \sum_{(i,j) \in \Omega} (D_{ij} - X_{ij})^2 \text{ s.t. } \text{Rank}(\boldsymbol{X}) \leq k_0, \tag{2.11}$$

where k_0 is a predefined target rank.

Although we require $\lambda, \mu > 0$ in our formulation of SLR given by (2.1), we now show that if we take $\mu = 0$ and also fix a sparsity pattern for the sparse matrix $\boldsymbol{Y}$, then (2.1) reduces to regularized matrix completion. Let $\boldsymbol{Z} \in \{0,1\}^{n \times n}$ be a matrix such that if $Z_{ij} = 0$, we must have $Y_{ij} = 0$. We refer to $\boldsymbol{Z}$ as a valid sparsity pattern for (2.1) if $\sum_{ij} Z_{ij} \leq k_1$. Formally, we have (proof deferred to Section 2.8):

Proposition 5 *Given a valid sparsity pattern $\boldsymbol{Z}$, if we take $\mu = 0$ then (2.1) reduces to regularized matrix completion with $\Omega = \{(i,j) : Z_{ij} = 0\}$.*

2.3 An Alternating Minimization Heuristic

In this section, we propose an alternating minimization algorithm that obtains high quality feasible solutions to (2.1) in Section 2.3.2, by iteratively fixing the sparse or low-rank matrix and optimizing the remaining matrix. This is a reasonable strategy, because alternating minimization (AM) strategies are known to obtain high quality solutions to low-rank problems [84] and, as we demonstrate in Section 2.3.1, when one matrix is fixed the other matrix can be optimized in closed form. Consequently, Problem (2.1) is amenable to AM techniques. Further, in Section 2.3.2, we bound the number of iterations required for AM to converge. Finally, in Section 2.3.3, we establish that for a fixed sparsity pattern and a sufficiently large amount of regularization, AM yields a globally optimal solution to (2.1). This result provides the basis for the branch and bound algorithm we develop in Section 2.5.

2.3.1 Two Natural Subproblems

In this subsection, we derive two subproblems of (2.1) by fixing either the sparse matrix $\boldsymbol{Y}$ (to obtain a low-rank subproblem) or the low-rank matrix $\boldsymbol{Y}$ (to obtain a sparse subproblem). Further, we establish that both subproblems admit closed-form solutions.

Low-Rank Subproblem: First, suppose that we fix a sparse matrix $\boldsymbol{Y}^*$ in Problem (2.1). Then, (2.1) becomes:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times n}} \quad \|\bar{\boldsymbol{D}} - \boldsymbol{X}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 \quad \text{s.t.} \quad \text{Rank}(\boldsymbol{X}) \leq k_0, \tag{2.12}$$

where $\bar{\boldsymbol{D}} = \boldsymbol{D} - \boldsymbol{Y}^*$ and we omit the regularization term on $\boldsymbol{Y}$ since it does not depend on $\boldsymbol{X}$. We refer to Problem (2.12) as the low-rank subproblem. We now demonstrate that this problem admits a closed-form solution, via the following result:

Proposition 6 *Let $\boldsymbol{X}^*$ be a matrix such that*

$$\boldsymbol{X}^* = \frac{1}{1+\lambda}\bar{\boldsymbol{D}}_{k_0},$$

where $\bar{\boldsymbol{D}}_{k_0}$ is a top-k_0 SVD approximation of $\bar{\boldsymbol{D}}$, i.e., $\bar{\boldsymbol{D}}_{k_0} = \boldsymbol{U}_{k_0}\boldsymbol{\Sigma}_{k_0}\boldsymbol{V}_{k_0}^T$ where $\bar{\boldsymbol{D}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a singular value decomposition of $\bar{\boldsymbol{D}}$. Then, $\boldsymbol{X}^\star$ is an optimal solution to Problem (2.12).

Proof It is well known that the solution of the problem

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times n}} \quad \|\boldsymbol{A} - \boldsymbol{X}\|_F^2 \quad \text{s.t.} \quad \text{Rank}(\boldsymbol{X}) \leq k_0$$

is given by $\boldsymbol{X}^* = \boldsymbol{A}_{k_0}$, a projection of $\boldsymbol{A}$ onto its first k_0 principal components [144]. Moreover, since

$$\|\bar{\boldsymbol{D}} - \boldsymbol{X}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 - \frac{\lambda}{1+\lambda}\|\bar{\boldsymbol{D}}\|_F^2 = (1+\lambda)\left\|\frac{1}{1+\lambda}\bar{\boldsymbol{D}} - \boldsymbol{X}\right\|_F^2,$$

it follows that Problem (2.12) is equivalent to (has the same optimal solution set as) solving

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times n}} \quad \left\| \frac{1}{1+\lambda}\bar{\boldsymbol{D}} - \boldsymbol{X} \right\|_F^2 \quad \text{s.t.} \quad \text{Rank}(\boldsymbol{X}) \leq k_0. \tag{2.13}$$

■

In Section 2.9, we provide an alternate proof of Proposition 6 via strong duality which reveals that (2.12) exhibits hidden convexity in the sense of [10].

Remark 7 *Observe that $\boldsymbol{X}^\star$ can be computed exactly in $O(n^2k)$ time, since we need not compute a full SVD of $\bar{\boldsymbol{D}}$. Alternatively, it can be computed approximately using randomized SVD in $O(n^2 \log k)$ time [80].*

Sparse Subproblem: Now, suppose we fix a low-rank matrix $\boldsymbol{X}^*$ in Problem (2.1). Then, (2.1) problem becomes:

$$\min_{\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad \|\tilde{\boldsymbol{D}} - \boldsymbol{Y}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{Y}\|_0 \leq k_1, \tag{2.14}$$

where $\tilde{\boldsymbol{D}} = \boldsymbol{D} - \boldsymbol{X}^*$ and we have omitted the regularization term on the low-rank matrix because it does not depend on $\boldsymbol{Y}$. We refer to Problem (2.14) as the sparse matrix subproblem. We now demonstrate that this problem also admits a closed-form solution:

Proposition 8 *Let $\boldsymbol{Y}^*$ be a matrix such that*

$$\boldsymbol{Y}^* = \boldsymbol{S}^* \circ \left(\frac{\tilde{\boldsymbol{D}}}{1+\mu} \right),$$

where $\boldsymbol{S}^*$ is a $n \times n$ binary matrix with k_1 entries $S^\star_{ij} = 1$ such that $S^\star_{i,j} \geq S^\star_{k,l}$ if $|\tilde{D}_{i,j}| \geq |\tilde{D}_{k,l}|$ and $\circ$ denotes the Hadamard product operation ($(\boldsymbol{A} \circ \boldsymbol{B})_{ij} = A_{ij} \times B_{ij}$). Then, $\boldsymbol{Y}^\star$ solves Problem (2.14).

Proof It is straightforward to show that the solution of:

$$\min_{\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad \|\boldsymbol{B}-\boldsymbol{Y}\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{Y}\|_0 \leq k_1$$

is given by $\boldsymbol{Y}^* = \boldsymbol{T}^* \circ \boldsymbol{B}$ where $\boldsymbol{T}^*$ is a $n \times n$ binary matrix with k_1 entries $T^\star_{ij} = 1$ such that $T^\star_{i,j} \geq T^\star_{k,l}$ if $|B_{i,j}| \geq |B_{k,l}|$. Moreover, since

$$\|\tilde{\boldsymbol{D}}-\boldsymbol{Y}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2 - \frac{\mu}{1+\mu}\|\tilde{\boldsymbol{D}}\|_F^2 = (1+\mu)\left\|\frac{1}{1+\mu}\tilde{\boldsymbol{D}}-\boldsymbol{Y}\right\|_F^2,$$

it follows that Problem (2.14) is equivalent to (i.e., has the same optimal solution set as):

$$\min_{\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad \left\|\frac{1}{1+\mu}\tilde{\boldsymbol{D}}-\boldsymbol{Y}\right\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{Y}\|_0 \leq k_1. \tag{2.15}$$

■

In Section 2.11, we provide an alternative proof of Proposition 8 via strong second-order cone duality which may be of independent interest as it reveals that Problem (2.15) is equivalent to a convex optimization problem.

Remark 9 *Observe that $\boldsymbol{Y}^\star$ can be computed in $O(n^2)$ time, by forming $\tilde{\boldsymbol{D}}$ and partitioning around its kth largest absolute element via quicksort. Correspondingly, this step is computationally cheaper than computing an optimal low-rank matrix. Moreover, since $\tilde{\boldsymbol{D}} \in \mathbb{R}^{n\times n}$, this operation is linear in the number of entries of $\tilde{\boldsymbol{D}}$.*

2.3.2 An Alternating Minimization Algorithm

By iteratively solving the sparse matrix subproblem and the low-rank matrix subproblem until we either converge to a stationary point or exceed a prespecified number of iterations, we arrive at a feasible solution to (2.1). We formalize this iterative procedure in Algorithm

1, and let

$$f(\boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2,$$

be our overall objective function and $\mathcal{V} = \{\boldsymbol{X} \in \mathbb{R}^{n\times n} : \text{Rank}(\boldsymbol{X}) \leq k_0\}$, $\mathcal{W} = \{\boldsymbol{Y} \in \mathbb{R}^{n\times n} : \sum_{ij} \mathbb{1}\{Y_{ij} \neq 0\} \leq k_1\}$ denote our respective feasible regions.

Algorithm 1: Alternating Minimization Heuristic

Data: $\boldsymbol{D} \in \mathbb{R}^{n\times n}, \lambda, \mu > 0, k_0, k_1 \in \mathbb{Z}^+$, tolerance parameter $\epsilon > 0$.
Result: $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ feasible and stationary for Problem (2.31)
$\boldsymbol{X}_0 \leftarrow \boldsymbol{0}$; $\boldsymbol{Y}_0 \leftarrow \boldsymbol{0}$;
$f_0 \leftarrow f(\boldsymbol{X}_0, \boldsymbol{Y}_0)$;
$t \leftarrow 0$;
do
  $t \leftarrow t + 1$;
  $\boldsymbol{Y}_t \leftarrow \arg\min_{\boldsymbol{Y}\in\mathcal{W}} f(\boldsymbol{X}_{t-1}, \boldsymbol{Y})$;
  $\boldsymbol{X}_t \leftarrow \arg\min_{\boldsymbol{X}\in\mathcal{V}} f(\boldsymbol{X}, \boldsymbol{Y}_t)$;
  $f_t \leftarrow f(\boldsymbol{X}_t, \boldsymbol{Y}_t)$;
while $f_t > 0$ *and* $\frac{f_{t-1}-f_t}{f_t} \geq \epsilon$;
return $\bar{\boldsymbol{X}} = \boldsymbol{X}_t$, $\bar{\boldsymbol{Y}} = \boldsymbol{Y}_t$

We note that the initialization strategy $\boldsymbol{X}_0 \leftarrow \boldsymbol{0}$ and $\boldsymbol{Y}_0 \leftarrow \boldsymbol{0}$ is arbitrary and any initialization strategy could equivalently be employed. For instance, one could employ a greedy rounding of the solution to the semidefinite relaxation we derive in Section 2.4 as an initialization [see also 19, Section 4.3]. Moreover, Algorithm 1 can be executed multiple times for different initializations of $\boldsymbol{X}_0$ and $\boldsymbol{Y}_0$ to obtain an even higher quality feasible solution to (2.31). This could be performed in parallel to avoid significantly increasing computational time.

It is well-documented in the optimization and machine learning literature that alternating minimization schemes such as Algorithm 1 produce a sequence of non-increasing iterates that converge to a local minimum; for Algorithm 1, this can be shown as a straightforward corollary of [154, Theorem 1]. Building upon this, we now demonstrate that, for a given relative improvement tolerance ϵ, Algorithm 1 terminates in a finite number of iterations. Indeed, Algorithm 1 terminates at iteration t if either $f_t = 0$ or $f_t > \left(\frac{1}{1+\epsilon}\right) f_{t-1}$. For any iter-

ation t, the update rules for $\boldsymbol{X}_{t+1}$ and $\boldsymbol{Y}_{t+1}$ imply that $f_{t+1} = f(\boldsymbol{X}_{t+1}, \boldsymbol{Y}_{t+1}) \leq f(\boldsymbol{X}_t, \boldsymbol{Y}_{t+1}) \leq f(\boldsymbol{X}_t, \boldsymbol{Y}_t) = f_t$. This implies that the sequence $\{f_t\}$ is strictly non-increasing.

Proposition 10 *Algorithm 1 terminates after at most* $\frac{\log \frac{\mu+\lambda+\mu\lambda}{\mu\lambda}}{\log 1+\epsilon}$ *iterations.*

Proof Assume that $\boldsymbol{D} \neq 0$. The case when $\boldsymbol{D} = 0$ is trivial as in this setting, Algorithm 1 terminates immediately because $f_0 = 0$. Suppose Algorithm 1 has yet to terminate after iteration t. This implies that

$$0 < f_t \leq \left(\frac{1}{1+\epsilon}\right) f_{t-1} \leq \left(\frac{1}{1+\epsilon}\right)^t f_0.$$

Recall that $f_0 = f(\boldsymbol{0}, \boldsymbol{0}) = \|\boldsymbol{D}\|_F^2$. Moreover, for all t we must have

$$f_t \geq \min_{\boldsymbol{X}\in\mathcal{V}, \boldsymbol{Y}\in\mathcal{W}} f(\boldsymbol{X}, \boldsymbol{Y}) \geq \min_{\boldsymbol{X}, \boldsymbol{Y}\in\mathbb{R}^{n\times n}} f(\boldsymbol{X}, \boldsymbol{Y}).$$

Simple unconstrained minimization gives $\min_{\boldsymbol{X}, \boldsymbol{Y}\in\mathbb{R}^{n\times n}} f(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\mu\lambda}{\mu+\lambda+\mu\lambda}\|\boldsymbol{D}\|_F^2$. Combining the above inequalities, we obtain

$$\frac{\mu\lambda}{\mu+\lambda+\mu\lambda}\|\boldsymbol{D}\|_F^2 \leq f_t \leq \left(\frac{1}{1+\epsilon}\right)^t \|\boldsymbol{D}\|_F^2.$$

The result follows by noting that the above inequality is violated if $t > \frac{\log \frac{\mu+\lambda+\mu\lambda}{\mu\lambda}}{\log 1+\epsilon}$. ■

In Section 2.4, we complement this result by introducing a lower bound that can be used to certify the quality of the solution returned by Algorithm 1. Moreover, in Section 2.6, we demonstrate numerically that Algorithm 1 produces high-quality solutions to (2.31).

2.3.3 Optimality of Algorithm 1 for a Fixed Sparsity Pattern

In this section, we establish the optimality of Algorithm 1 for a fixed sparsity pattern under certain easy-to-verify conditions that often hold in practice. Accordingly, here and

throughout this section, we assume we are given a collection of indices $\mathcal{I}_0 \subset \{(i,j) : 1 \leq i,j \leq n\}$, $\|\mathcal{I}_0\| = n^2 - k_1$ that correspond to entries of the sparse matrix $\boldsymbol{Y}$ that must take value 0, and that $\boldsymbol{S}^\star$ is a binary matrix that encodes this sparsity pattern. The collection $\mathcal{I}_0$ specifies a complete feasible sparsity pattern for the matrix $\boldsymbol{Y}$.

Given the sparsity pattern specified by $\mathcal{I}_0$, Problem (2.1) reduces to

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \lambda\cdot\|\boldsymbol{X}\|_F^2 + \mu\cdot\|\boldsymbol{Y}\|_F^2 \\ \text{s.t.} \quad & \text{Rank}(\boldsymbol{X}) \leq k_0,\ Y_{ij} = 0\ \forall (i,j)\in\mathcal{I}_0. \end{aligned} \tag{2.16}$$

Algorithm 1 can be easily adapted to produce a feasible solution to Problem (2.16). Indeed, by Proposition 8, an optimal binary matrix $\boldsymbol{Y}^\star$ in (2.16) is given by

$$\boldsymbol{Y}^* = \boldsymbol{S}^* \circ \left(\frac{\boldsymbol{D}-\boldsymbol{X}}{1+\mu}\right).$$

Moreover, applying Algorithm 1 with a fixed sparsity pattern and fixed low-rank matrix recovers this sparse matrix automatically. Thus, applying Algorithm 1 to Problem (2.16) is equivalent to solving the following non-convex optimization problem:

$$\begin{aligned} \min_{\boldsymbol{X}\in\mathbb{R}^{n\times n}} \quad & \left\|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{S}^*\circ\left(\frac{\boldsymbol{D}-\boldsymbol{X}}{1+\mu}\right)\right\|_F^2 + \lambda\cdot\|\boldsymbol{X}\|_F^2 + \mu\cdot\left\|\boldsymbol{S}^*\circ\left(\frac{\boldsymbol{D}-\boldsymbol{X}}{1+\mu}\right)\right\|_F^2 \\ \text{s.t.} \quad & \text{Rank}(\boldsymbol{X}) \leq k_0. \end{aligned} \tag{2.17}$$

Let us now define some additional notation: let $g(\boldsymbol{X})$ denote the objective value function of (2.17), $\Omega = \{\boldsymbol{X}\in\mathbb{R}^{n\times n} : \text{Rank}(\boldsymbol{X}) \leq k_0\}$ denote the set of n-by-n matrices with rank at most k_0, $\mathcal{P}_{\mathcal{X}}(\cdot)$ denote the projection operator onto a set $\mathcal{X}\subseteq\mathbb{R}^{n\times n}$, i.e., $\mathcal{P}_{\mathcal{X}}(\boldsymbol{Y}) = \arg\min_{\boldsymbol{X}\in\mathcal{X}}\|\boldsymbol{Y}-\boldsymbol{X}\|_F^2$, and let $\gamma_k(\boldsymbol{X}) = \frac{\sigma_{k+1}(\boldsymbol{X})}{\sigma_k(\boldsymbol{X})} \leq 1$ denote the ratio between the $(k+1)$th and the kth singular values of $\boldsymbol{X}$.

We have the following result (proof deferred to Section 2.8):

Proposition 11 *Given a full sparsity pattern* $\mathcal{I}_0 \subset \{(i,j) : 1 \leq i,j \leq n\}$, $\|\mathcal{I}_0\| = n^2 - k_1$, *if we constrain the binary matrix* $\boldsymbol{S}^*$ *in the solution of the sparse matrix subproblem* (2.14) *to satisfy* $S^*_{ij} = 0 \iff (i,j) \in \mathcal{I}_0$, *then Algorithm 1 is equivalent to performing Projected Gradient Descent on* (2.17) *given by* $\boldsymbol{X}_{t+1} = \mathcal{P}_\Omega(\boldsymbol{X}_t - \eta \nabla g(\boldsymbol{X}_t))$ *with step size* $\eta = \frac{1}{2(1+\lambda)}$. *By equivalent, we mean that the two algorithms produce the same sequence of feasible low-rank iterates* $\boldsymbol{X}_t$ *and that we have* $f(\boldsymbol{X}_t, \boldsymbol{Y}_t) = g(\boldsymbol{X}_t)$ *for all iterations* t *where* $\boldsymbol{Y}_t$ *denotes the sparse matrix iterates produced by Algorithm 1.*

We are now ready to establish the main result. We have:

Theorem 12 *Given a full sparsity pattern* $\mathcal{I}_0 \subset \{(i,j) : 1 \leq i,j \leq n\}$, $\|\mathcal{I}_0\| = n^2 - k_1$, *let* $\boldsymbol{S}^\star$ *be the binary matrix satisfying* $S^\star_{ij} = 0 \iff (i,j) \in \mathcal{I}_0$. *Let* $\boldsymbol{X}^\star$ *denote the optimal low-rank matrix for* (2.17) *and define* $\tilde{\boldsymbol{D}} = \left(\frac{1}{1+\lambda}\left[\boldsymbol{D} - \boldsymbol{S}^* \circ \left(\frac{\boldsymbol{D}-\boldsymbol{X}^\star}{1+\mu}\right)\right]\right)$.

Assume $\mathrm{Rank}(\boldsymbol{X}^\star) = k_0$ *and suppose that the following two conditions hold:*

1. $\lambda + \frac{2\mu}{1+\mu} - 1 > 0$;
2. $\gamma_{k_0}(\tilde{\boldsymbol{D}}) < \frac{1}{1+\lambda}\left(\lambda + \frac{2\mu}{1+\mu} - 1.\right)$.

Alternatively, assume $\mathrm{Rank}(\boldsymbol{X}^\star) < k_0$ *and suppose only the first condition listed above holds. In both of these two settings, Algorithm 1 converges linearly to the unique optimal solution of Problem* (2.16) *(where we constrain the binary matrix* $\boldsymbol{S}^*$ *in the solution of the sparse subproblem* (2.14) *to satisfy* $S^*_{ij} = 0 \iff (i,j) \in \mathcal{I}_0$*). Specifically, letting* $\{(\boldsymbol{X}_t, \boldsymbol{Y}_t)\}_{t=1}^\infty$ *denote the sequence of iterates generated by Algorithm 1 and* $(\boldsymbol{X}^*, \boldsymbol{Y}^*)$ *denote the optimal solution of* (2.16), *we have*

$$\frac{f(\boldsymbol{X}_{t+1}, \boldsymbol{Y}_{t+1}) - f(\boldsymbol{X}^*, \boldsymbol{Y}^*)}{f(\boldsymbol{X}_t, \boldsymbol{Y}_t) - f(\boldsymbol{X}^*, \boldsymbol{Y}^*)} \leq \frac{1}{(2\lambda+1)(1+\mu)+\mu} \quad \forall\ t.$$

Note that the first condition on the regularization parameters λ and μ in Theorem 12 is equivalent to requiring that the objective function of (2.17) has a small condition number.

The second condition is a more technical one that requires that the gradient of the objective function at the optimal solution of (2.17) is never too large.

Remark 13 *Theorem 12 implies that there is a phase transition in Problem (2.1)'s difficulty as the amount of regularization increases. Indeed, when $\mu = 0$ and the sparsity pattern is fixed, Problem (2.1) is equivalent to matrix completion (Proposition 5), which is a problem that may admit multiple local minima [19], and this may cause Algorithm 1 to converge to a non-global local optimum. On the other hand, our main result implies that, with a sufficiently large regularization term, Problem (2.1) can be solved to certifiable optimality by enumerating the sparsity patterns and running alternating minimization on each fixed sparsity pattern. Thus, regularization partially controls the complexity of (2.1).*

Proof We establish the result by invoking Theorem 3.3 from [79]. We prove the result for the more involved case where $\text{Rank}(\boldsymbol{X}^\star) = k_0$. The proof for the case where $\text{Rank}(\boldsymbol{X}^\star) < k_0$ follows similar reasoning by combining Proposition 11 with [79, Theorem 3.3]. We observe that the objective function $g(\boldsymbol{X})$ of (2.17) is m-strongly convex and L-Lipschitz continuous with $m = 2\lambda + \frac{2\mu}{1+\mu}$ and $L = 2\lambda + 2$. To see this, note that we have

$$g(\boldsymbol{X}) - \frac{m}{2}\|\boldsymbol{X}\|_F^2 = \Big(\lambda - \frac{m}{2}\Big)\|\boldsymbol{X}\|_F^2 + \sum_{(i,j)\in\mathcal{I}_0}(D_{ij} - X_{ij})^2 + \sum_{(i,j)\notin\mathcal{I}_0}(D_{ij} - X_{ij})^2 \cdot \frac{\mu}{1+\mu},$$

which is convex when $m = 2\lambda + \frac{2\mu}{1+\mu}$. Similarly, we have

$$\frac{L}{2}\|\boldsymbol{X}\|_F^2 - g(\boldsymbol{X}) = \Big(\frac{L}{2} - \lambda\Big)\|\boldsymbol{X}\|_F^2 - \sum_{(i,j)\in\mathcal{I}_0}(D_{ij} - X_{ij})^2 - \sum_{(i,j)\notin\mathcal{I}_0}(D_{ij} - X_{ij})^2 \cdot \frac{\mu}{1+\mu},$$

which is convex when $L = 2\lambda + 2$. Suppose that $\boldsymbol{X}^\star$ is a global minimizer of (2.17). We claim that gradient of $g(\boldsymbol{X})$ at $\boldsymbol{X}^\star$ satisfies:

$$\|\nabla g(\boldsymbol{X}^\star)\|_\sigma = 2(1+\lambda)\gamma_{k_0}(\tilde{\boldsymbol{D}})\sigma_{k_0}(\boldsymbol{X}^\star),$$

where $\|\boldsymbol{X}\|_\sigma = \sigma_1(\boldsymbol{X})$ denotes the spectral norm of $\boldsymbol{X}$. To see this, note that since $\boldsymbol{X}^\star$ is an optimal solution, it must be a fixed point of (2.37). Thus, we have

$$\begin{aligned}
\|\nabla g(\boldsymbol{X}^\star)\|_\sigma &= 2(1+\lambda)\left\|\boldsymbol{X}^\star - \frac{1}{1+\lambda}\left(\boldsymbol{D} - \boldsymbol{S}^* \circ \left(\frac{\boldsymbol{D}-\boldsymbol{X}^\star}{1+\mu}\right)\right)\right\|_\sigma \\
&= 2(1+\lambda)\|\boldsymbol{X}^\star - \tilde{\boldsymbol{D}}\|_\sigma \\
&= 2(1+\lambda)\sigma_{k_0+1}(\tilde{\boldsymbol{D}}) \\
&= 2(1+\lambda)\gamma_{k_0}(\tilde{\boldsymbol{D}})\sigma_{k_0}(\tilde{\boldsymbol{D}}) \\
&= 2(1+\lambda)\gamma_{k_0}(\tilde{\boldsymbol{D}})\sigma_{k_0}(\boldsymbol{X}^\star),
\end{aligned}$$

where the third and fifth equalities follow from $\boldsymbol{X}^\star$ being a fixed point of (2.37) and the fourth equality follows from the definition of $\gamma_{k_0}(\tilde{\boldsymbol{D}})$. It is easy to verify that when the first condition of Theorem 12 holds, the condition number $\kappa = \frac{L}{m}$ of $g(\boldsymbol{X})$ satisfies $\kappa < 2$. Moreover, when the second condition of Theorem 12 holds, it can similarly be verified that the gradient of $g(\boldsymbol{X})$ at $\boldsymbol{X}^\star$ satisfies $\|\nabla g(\boldsymbol{X}^\star)\|_\sigma < (2m - L)\sigma_{k_0}(\boldsymbol{X}^\star)$. Invoking the result of Theorem 3.3 from [79], $\boldsymbol{X}^\star$ is the unique fixed point of Projected Gradient Descent with step size $\eta = \frac{1}{2(1+\lambda)}$. Invoking Proposition 11, this immediately implies that Algorithm 1 converges to $\boldsymbol{X}^\star$.

Finally, it is known that Projected Gradient Descent converges linearly with rate $\frac{\kappa-1}{\kappa+1}$ for strongly convex functions [121]. Combining this with Proposition 11, we have

$$\frac{g(\boldsymbol{X}_{t+1}) - g(\boldsymbol{X}^*)}{g(\boldsymbol{X}_t) - g(\boldsymbol{X}^*)} = \frac{f(\boldsymbol{X}_{t+1}, \boldsymbol{Y}_{t+1}) - f(\boldsymbol{X}^*, \boldsymbol{Y}^*)}{f(\boldsymbol{X}_t, \boldsymbol{Y}_t) - f(\boldsymbol{X}^*, \boldsymbol{Y}^*)} \leq \frac{\kappa - 1}{\kappa + 1} = \frac{1}{(2\lambda+1)(1+\mu)+\mu},$$

which holds for all t. This completes the proof. ■

2.4 A Convex Relaxation

In this section, we reformulate (2.1) as a mixed-integer, mixed-projection optimization problem. We then employ the (matrix) perspective relaxation [19, 20, 77] to construct a convex relaxation of (2.1). We illustrate the power of our convex relaxation in Section 2.4.1, by demonstrating that it reflects the hidden convexity of the low-rank subproblem we derived in the previous section and allows this subproblem to be solved via convex optimization. Further, we compare our convex relaxation to the previously derived relaxation of [97] in Section 2.4.2 and demonstrate that when both relaxations make the same assumptions, our relaxation is at least as powerful, and sometimes strictly more powerful. Finally, in Section 2.4.3, we interpret (a slightly modified version of, where the sparsity and rank are penalized rather than constrained) our convex relaxation as a convex penalty.

To model the sparsity pattern of the sparse matrix $\boldsymbol{Y}$, we introduce binary variables $\boldsymbol{Z} \in \{0,1\}^{n\times n}$ and require that $Y_{ij} = 0$ if $Z_{i,j} = 0$ by imposing the nonlinear constraint $Y_{i,j} = Y_{i,j}Z_{i,j}$, and also require that $\sum_{ij} Z_{ij} \leq k_1$. To model the column space of $\boldsymbol{X}$, we introduce an orthogonal projection matrix $\boldsymbol{P} \in \mathcal{P}$ and require that $\mathrm{tr}(\boldsymbol{P}) \leq k_0$ and $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}$. Let $\mathcal{Z}_{k_1} = \{\boldsymbol{Z} \in \{0,1\}^{n\times n} : \sum_{ij} Z_{ij} \leq k_1\}$ and $\mathcal{P}_{k_0} = \{\boldsymbol{P} \in \mathcal{S}^n : \boldsymbol{P}^2 = \boldsymbol{P}, \mathrm{tr}(\boldsymbol{P}) \leq k_0\}$. This gives the following reformulation of (2.1):

$$\begin{aligned} \min_{\boldsymbol{Z}\in\mathcal{Z}_{k_1},\boldsymbol{P}\in\mathcal{P}_{k_0}} \min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda \cdot \|\boldsymbol{X}\|_F^2 + \mu \cdot \|\boldsymbol{Y}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}, \boldsymbol{Y} = \boldsymbol{Z} \circ \boldsymbol{Y}. \end{aligned} \tag{2.18}$$

We now have the following result (proof deferred to Section 2.8):

Proposition 14 *Problem* (2.18) *is a valid reformulation of Problem* (2.1).

The constraints $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}$ and $\boldsymbol{Y} = \boldsymbol{Z} \circ \boldsymbol{Y}$ in (2.18) are complicating because they are non-convex in the decision variables $(\boldsymbol{Z}, \boldsymbol{P}, \boldsymbol{X}, \boldsymbol{Y})$. Accordingly, to model these constraints in a convex manner, we invoke the (matrix) perspective reformulation [19, 20, 77]. Specifically,

to model the sparse matrix $\boldsymbol{Y}$, we introduce variables $\boldsymbol{\alpha} \in \mathbb{R}^{n\times n}$ where α_{ij} models Y_{ij}^2, and the constraint $\alpha_{ij} Z_{ij} \geq Y_{ij}^2$, which is second-order cone representable. To model the low-rank matrix $\boldsymbol{X}$, we introduce a variable $\boldsymbol{\Theta} \in \mathbb{R}^{n\times n}$ that models $\boldsymbol{X}^T\boldsymbol{X}$, and the constraint $\left(\begin{smallmatrix} \boldsymbol{\Theta} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{P} \end{smallmatrix}\right) \succeq 0$.

This yields the following reformulation of (2.18):

$$\begin{aligned} \min_{\boldsymbol{Z}\in\mathcal{Z},\boldsymbol{P}\in\mathcal{P}} \min_{\boldsymbol{X},\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \lambda\cdot\mathrm{tr}(\boldsymbol{\Theta}) + \mu\cdot\langle \boldsymbol{E},\boldsymbol{\alpha}\rangle \\ \text{s.t.} \quad & \boldsymbol{Y}\circ\boldsymbol{Y} \leq \boldsymbol{\alpha}\circ\boldsymbol{Z},\ \begin{pmatrix} \boldsymbol{\Theta} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{P} \end{pmatrix} \succeq 0, \end{aligned} \tag{2.19}$$

where $\boldsymbol{E}$ denotes a matrix of all ones of appropriate dimension.

Problem (2.19) is a reformulation of Problem (2.1) where the problem's non-convexity is entirely captured by the non-convex sets $\mathcal{Z}_{k_1}$ and $\mathcal{P}_{k_0}$. We now obtain a convex relaxation of (2.1) by solving (2.19) with $\boldsymbol{Z} \in \mathrm{conv}(\mathcal{Z}_{k_1})$ and $\boldsymbol{P} \in \mathrm{conv}(\mathcal{P}_{k_0})$ where $\mathrm{conv}(\mathcal{X})$ denotes the convex hull of the set $\mathcal{X}$. It is straightforward to see that $\mathrm{conv}(\mathcal{Z}_{k_1}) = \{\boldsymbol{Z} \in [0,1]^{n\times n} : \sum_{ij} Z_{ij} \leq k_1\}$. Moreover, we have $\mathrm{conv}(\mathcal{P}_{k_0}) = \{\boldsymbol{P} \in \mathcal{S}_+^n : \mathbb{I} - \boldsymbol{P} \succeq 0, \mathrm{tr}(\boldsymbol{P}) \leq k_0\}$[113]. This gives the following convex optimization problem:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\boldsymbol{P},\boldsymbol{\Theta},\boldsymbol{\alpha}\in\mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \lambda\cdot\mathrm{tr}(\boldsymbol{\Theta}) + \mu\cdot\langle \boldsymbol{E},\boldsymbol{\alpha}\rangle \\ \text{s.t.} \quad & \boldsymbol{Y}\circ\boldsymbol{Y} \leq \boldsymbol{\alpha}\circ\boldsymbol{Z},\ \langle \boldsymbol{E},\boldsymbol{Z}\rangle \leq k_1,\ \boldsymbol{0} \leq \boldsymbol{Z} \leq \boldsymbol{E}, \\ & \boldsymbol{P} \succeq 0,\ \mathbb{I}-\boldsymbol{P} \succeq 0,\ \mathrm{tr}(\boldsymbol{P}) \leq k_0,\ \begin{pmatrix} \boldsymbol{\Theta} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{P} \end{pmatrix} \succeq 0. \end{aligned} \tag{2.20}$$

We now have the following result (proof deferred to Section 2.8):

Theorem 15 *Problem (2.20) is a valid convex relaxation of (2.1).*

Note that Problem (2.20) only produces a nontrivial lower bound to (2.1) when the regularization parameters satisfy $\lambda, \mu > 0$. If either $\lambda = 0$ or $\mu = 0$, it can easily be shown

that the optimal value of (2.20) is 0. In Section 2.6, we employ this convex relaxation to produce bounds for feasible solutions returned by Algorithm 1. Moreover, we show that (2.20) can be embedded within a branch and bound framework.

2.4.1 Hidden Convexity in the Low Rank Subproblem

In this section, we demonstrate that the low-rank subproblem derived in the previous section exhibits hidden convexity in the sense of [10]. This result allows us to establish the strength of our overall convex relaxation in the next section. Formally, we have the following result (proof deferred to Section 2.10):

Theorem 16 *Consider the semidefinite optimization problem:*

$$\begin{aligned} \min_{\boldsymbol{P},\boldsymbol{\Theta}\in\mathcal{S}^n_+,\boldsymbol{X}\in\mathcal{S}^n} \quad & \|\bar{\boldsymbol{D}}\|_F^2 + (1+\lambda)\cdot \mathrm{tr}(\boldsymbol{\Theta}) - 2\cdot\langle \boldsymbol{X}, \bar{\boldsymbol{D}}\rangle \\ \text{s.t.} \quad & \mathrm{tr}(\boldsymbol{P}) \leq k_0,\ \mathbb{I} - \boldsymbol{P} \succeq 0,\ \begin{pmatrix} \boldsymbol{\Theta} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{P} \end{pmatrix} \succeq 0. \end{aligned} \tag{2.21}$$

Solving Problem (2.12) is equivalent to solving Problem (2.21) in that both problems have the same optimal objective value and given an optimal solution to either problem, an optimal solution to the other problem can be constructed efficiently.

2.4.2 Comparison With the Relaxation of Lee and Zou

To illustrate the power of our convex relaxation, we now present a formal comparison between (2.20) and the relaxation proposed by [97] and demonstrate that our relaxation is at least as powerful and sometimes strictly more powerful. Accordingly, here and throughout this subsection, we assume that the spectral norm of the low-rank matrix $\boldsymbol{X}$ and the infinity norm of the sparse matrix $\boldsymbol{Y}$ are bounded as otherwise the relaxation proposed by [97] yields a lower bound of zero. Explicitly, we assume that $\|\boldsymbol{X}\|_\sigma = \max_i \sigma_i(\boldsymbol{X}) \leq \beta$ and $\|\boldsymbol{Y}\|_\infty = \max_{ij} |Y_{ij}| \leq \gamma$ where $\sigma_i(\boldsymbol{X})$ denotes the i^{th} singular value of $\boldsymbol{X}$ for $\beta, \gamma \in \mathbb{R}_+$.

[97] obtain their relaxation by noting that under the spectral and infinity norm boundedness assumptions, convex lower bounds of the non-convex rank and ℓ_0 norm functions can be obtained as $\text{Rank}(\boldsymbol{X}) \geq \frac{1}{\beta}\|\boldsymbol{X}\|_\star$ and $\|\boldsymbol{Y}\|_0 \geq \frac{1}{\gamma}\|\boldsymbol{Y}\|_1$ respectively. Noting that the ℓ_1 norm can be trivially linearized and that the nuclear norm of a matrix $\boldsymbol{X}$ admits a well-known semidefinite characterization given by

$$\min_{\boldsymbol{W}_1,\boldsymbol{W}_2 \in \mathcal{S}^n} \quad \frac{1}{2}\text{tr}(\boldsymbol{W}_1 + \boldsymbol{W}_2) \text{ s.t. } \begin{pmatrix} \boldsymbol{W}_1 & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{W}_2 \end{pmatrix} \succeq 0,$$

we can express [97]'s relaxation of (2.1) as follows:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{V},\boldsymbol{W}_1,\boldsymbol{W}_2 \in \mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda \cdot \|\boldsymbol{X}\|_F^2 + \mu \cdot \|\boldsymbol{Y}\|_F^2 \\ \text{s.t.} \quad & -\boldsymbol{V} \leq \boldsymbol{Y} \leq \boldsymbol{V},\ \frac{1}{\gamma}\langle \boldsymbol{E}, \boldsymbol{V}\rangle \leq k_1, \\ & \frac{1}{2\beta}\text{tr}(\boldsymbol{W}_1) + \frac{1}{2\beta}\text{tr}(\boldsymbol{W}_2) \leq k_0,\ \begin{pmatrix} \boldsymbol{W}_1 & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{W}_2 \end{pmatrix} \succeq 0. \end{aligned} \tag{2.22}$$

To allow for a fair comparison between our relaxation and that given by (2.22), we note that under the assumptions $\|\boldsymbol{X}\|_\sigma \leq \beta$ and $\|\boldsymbol{Y}\|_\infty \leq \gamma$, we can strengthen (2.20) as follows:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\boldsymbol{P}_c,\boldsymbol{P}_r,\boldsymbol{\Theta},\boldsymbol{\alpha} \in \mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda \cdot \text{tr}(\boldsymbol{\Theta}) + \mu \cdot \langle \boldsymbol{E}, \boldsymbol{\alpha}\rangle \\ \text{s.t.} \quad & \boldsymbol{Y} \circ \boldsymbol{Y} \leq \boldsymbol{\alpha} \circ \boldsymbol{Z},\ \langle \boldsymbol{E}, \boldsymbol{Z}\rangle \leq k_1,\ \boldsymbol{0} \leq \boldsymbol{Z} \leq \boldsymbol{E},\ -\gamma \boldsymbol{Z} \leq \boldsymbol{Y} \leq \gamma \boldsymbol{Z}, \\ & \boldsymbol{P}_c \succeq 0,\ \mathbb{I} - \boldsymbol{P}_c \succeq 0,\ \text{tr}(\boldsymbol{P}_c) \leq k_0, \\ & \boldsymbol{P}_r \succeq 0,\ \mathbb{I} - \boldsymbol{P}_r \succeq 0,\ \text{tr}(\boldsymbol{P}_r) \leq k_0, \\ & \begin{pmatrix} \boldsymbol{\Theta} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{P}_c \end{pmatrix} \succeq 0,\ \begin{pmatrix} \beta \boldsymbol{P}_r & \boldsymbol{X} \\ \boldsymbol{X}^T & \beta \boldsymbol{P}_c \end{pmatrix} \succeq 0. \end{aligned} \tag{2.23}$$

The constraint $-\gamma Z_{ij} \leq Y_{ij} \leq \gamma Z_{ij}$ in (2.23) emerges immediately from the bound on

the infinity norm of the sparse matrix. The last four constraints in (2.23) follow from the bound on the spectral norm of the low-rank matrix. The variable $\boldsymbol{P}_c$ plays the role of $\boldsymbol{P}$ in (2.20) and models the k_0 dimensional column space of $\boldsymbol{X}$ as before while the variable $\boldsymbol{P}_r$ models the k_0 dimensional row space of $\boldsymbol{X}$. To see that these four constraints are valid, consider any matrix $\bar{\boldsymbol{X}}$ satisfying $\|\bar{\boldsymbol{X}}\|_\star \leq \beta$ and $\text{Rank}(\bar{\boldsymbol{X}}) \leq k_0$, and let $\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be its singular value decomposition. Define $\bar{\boldsymbol{P}}_c = \boldsymbol{U}\boldsymbol{U}^T$ and $\bar{\boldsymbol{P}}_r = \boldsymbol{V}\boldsymbol{V}^T$. We have $\beta^2\bar{\boldsymbol{P}}_r \succeq \bar{\boldsymbol{P}}_r\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}^T\bar{\boldsymbol{P}}_c\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}^T\bar{\boldsymbol{P}}_c^\dagger\bar{\boldsymbol{X}}$ so we have $\begin{pmatrix} \beta\bar{\boldsymbol{P}}_r & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \beta\bar{\boldsymbol{P}}_c \end{pmatrix} \succeq 0$. Feasibility of $\bar{\boldsymbol{P}}_c$ and $\bar{\boldsymbol{P}}_r$ for the remaining constraints follows the same reasoning employed in Theorem 15. Note that if we restrict $\boldsymbol{X}$ to be symmetric, we can take $\boldsymbol{P}_r = \boldsymbol{P}_c$ in (2.23) as the row space and the column space of $\boldsymbol{X}$ will be the same.

Proposition 17 *For any input data $\boldsymbol{D}, k_0, k_1$ and hyperparameters λ, μ, the optimal value of (2.23) is no less than the optimal value of (2.22).*

Proof To establish the proposition, we show that for any feasible solution to (2.23) we can construct a feasible solution to (2.22) that achieves the same or lower objective value.

Fix any input data $\boldsymbol{D} \in \mathbb{R}^{n\times n}, k_0, k_1 \in \mathbb{N}_+$ and any hyperparameters $\lambda, \mu > 0$. Consider an arbitrary feasible solution $\mathcal{S}_1 = (\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{P}}_c, \bar{\boldsymbol{P}}_r, \bar{\boldsymbol{\Theta}}, \bar{\boldsymbol{\alpha}})$ to (2.23). Let $\bar{\boldsymbol{V}} = \gamma\bar{\boldsymbol{Z}}, \bar{\boldsymbol{W}}_1 = \beta\bar{\boldsymbol{P}}_c$ and $\bar{\boldsymbol{W}}_2 = \beta\bar{\boldsymbol{P}}_r$. We will show that the solution $\mathcal{S}_2 = (\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2)$ is feasible to (2.22) and achieves an objective value that is no larger than the objective value achieves by $\mathcal{S}_2$ in (2.23). From feasibility of $\mathcal{S}_1$ in (2.23), we have $-\gamma\bar{Z}_{ij} \leq \bar{Y}_{ij} \leq \gamma\bar{Z}_{ij} \implies -\bar{V}_{ij} \leq \bar{Y}_{ij} \leq \bar{V}_{ij}$ and $\langle \boldsymbol{E}, \bar{\boldsymbol{Z}}\rangle \leq k_1 \implies \frac{1}{\gamma}\langle \boldsymbol{E}, \bar{V}_{ij}\rangle \leq k_1$. Moreover, we have

$$\frac{1}{2\beta}\text{tr}(\bar{\boldsymbol{W}}_1 + \bar{\boldsymbol{W}}_2) = \frac{1}{2\beta}\text{tr}(\beta\bar{\boldsymbol{P}}_c + \beta\bar{\boldsymbol{P}}_r) = \frac{1}{2}\text{tr}(\bar{\boldsymbol{P}}_c) + \frac{1}{2}\text{tr}(\bar{\boldsymbol{P}}_c) \leq \frac{k_0}{2} + \frac{k_0}{2} = k_0$$

We conclude that $\mathcal{S}_2$ is feasible to (2.22) by noting that the last constraint in (2.22) reduces to the fourth from last constraint in (2.23) after substituting the definitions of $\bar{\boldsymbol{W}}_1$ and $\bar{\boldsymbol{W}}_2$. We observe that $\mathcal{S}_2$ achieves an objective value in (2.22) no greater than that achieved by $\mathcal{S}_1$ in (2.23) by noting that feasibility of $\mathcal{S}_1$ implies that $\text{tr}(\bar{\boldsymbol{\Theta}}) \geq \|\bar{\boldsymbol{X}}\|_F^2$ and $\langle \boldsymbol{E}, \bar{\boldsymbol{\alpha}}\rangle \geq \|\bar{\boldsymbol{Y}}\|_F^2$.

Since this construction holds for every feasible solution to (2.23), it must hold for any optimal solution, which implies that the optimal value of (2.22) is no greater than the optimal value of (2.23). This completes the proof. ■

Proposition 17 establishes that our relaxation is at least as strong as (2.22), but does not in and of itself demonstrate its utility since it does not preclude the possibility of the optimal value of (2.23) always coinciding with the optimal value of (2.22). To address this, Proposition 18 which establishes the existence of problem instances for which the optimal value of (2.23) is strictly greater than the optimal value of (2.22). Taken together, Propositions 17 and 18 show that (2.23) is a (strictly) stronger convex relaxation to (2.1) than (2.22).

Proposition 18 *There exists input data* $\boldsymbol{D}, k_0, k_1$ *and hyperparameters* λ, μ *such that the optimal value of* (2.23) *is strictly greater than the optimal value of* (2.22).

Proof We establish the result constructively. Let $n = 2, \boldsymbol{D} = \mathbb{I}_2, k_0 = 1, k_1 = 0, \lambda = 1$ and $\mu = 1$. With these values, (2.1) reduces to

$$\min_{\boldsymbol{X} \in \mathbb{R}^{2\times 2}} \quad \|\mathbb{I}_2 - \boldsymbol{X}\|_F^2 + \|\boldsymbol{X}\|_F^2 \quad \text{s.t. } \text{Rank}(\boldsymbol{X}) \leq 1. \tag{2.24}$$

It follows immediately from Proposition 6 that the optimal solution to (2.24) is $\boldsymbol{X}^\star = \begin{pmatrix} 0.5 & 0 \\ 0 & 0 \end{pmatrix}$ and the optimal objective value is $\frac{3}{2}$. Let $\beta = 2$ and $\gamma = 1$. Note that γ can be chosen arbitrarily since the optimal sparse matrix is $\boldsymbol{Y}^\star = \boldsymbol{0}$. Consider solving (2.23) and (2.22) for this problem data. From Theorem 16, it follows that the optimal value of (2.23) coincides with the optimal value of (2.24). Next, note that if we ignore the rank constraint, it can easily be verified that the unconstrained minimum of (2.24) is given by $\tilde{\boldsymbol{X}} = \frac{1}{2}\mathbb{I}$ and achieves an objective value of 1. Finally, observe that taking $\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{V}} = \boldsymbol{0}, \tilde{\boldsymbol{W}}_1 = \tilde{\boldsymbol{W}}_2 = \mathbb{I}$, the solution $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}, \tilde{\boldsymbol{V}}, \tilde{\boldsymbol{W}}_1, \tilde{\boldsymbol{W}}_2)$ is feasible to (2.22) and achieves an objective value of 1. This completes the proof. ■

2.4.3 Penalty Interpretation of Relaxation

We now consider instances where the sparsity and rank of the matrices are penalized in the objective rather than constrained and interpret the resulting relaxation as a penalty function in the tradition of [19, 66, 118, 122] among others. Formally, we have the following result[1], which can be deduced by combining [118, Corollary 3] with [19, Lemma 6]:

Proposition 19 *The following two optimization problems are equivalent:*

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\boldsymbol{P},\boldsymbol{\Theta},\boldsymbol{\alpha}\in\mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \lambda\cdot\operatorname{tr}(\boldsymbol{\Theta}) + \mu\cdot\langle\boldsymbol{E},\boldsymbol{\alpha}\rangle + \rho_1\cdot\operatorname{tr}(\boldsymbol{P}) + \rho_2\cdot\langle\boldsymbol{E},\boldsymbol{Z}\rangle \\ \text{s.t.} \quad & \boldsymbol{Y}\circ\boldsymbol{Y}\leq\boldsymbol{\alpha}\circ\boldsymbol{Z},\ \boldsymbol{0}\leq\boldsymbol{Z}\leq\boldsymbol{E},\ \boldsymbol{P}\succeq 0,\ \mathbb{I}-\boldsymbol{P}\succeq 0,\ \begin{pmatrix}\boldsymbol{\Theta} & \boldsymbol{X}\\ \boldsymbol{X}^T & \boldsymbol{P}\end{pmatrix}\succeq 0. \end{aligned} \tag{2.25}$$

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \sum_{i\in[n]}\min\left(\sqrt{\rho_1\lambda}\sigma_i(\boldsymbol{X}),\rho_1+\lambda\sigma_i(\boldsymbol{X})^2\right) \\ & \qquad + \sum_{i,j\in[n]}\min\left(\sqrt{\mu\rho_2}Y_{i,j},\rho_2+\mu Y_{i,j}^2\right). \end{aligned} \tag{2.26}$$

The above result demonstrates that our regularized relaxation generalizes the reverse Huber penalty [c.f. 118] to sparse plus low-rank optimization problems. This is quite different from unregularized low-rank problems. Indeed, it follows directly from [19, Lemma 7] that under a standard big-M assumption on the ℓ_∞ norm of the sparse matrix and the spectral

[1] Note that the statement of our result is slightly different to the statement in [118], because, as noted by Dong u. a. [61], the original result contains some minor typos.

norm of the low-rank matrix, an unregularized relaxation of the form

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\boldsymbol{P}\in\mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2+\rho_1\mathrm{tr}(\boldsymbol{P})+\rho_2\langle \boldsymbol{E},\boldsymbol{Z}\rangle \\ \text{s.t.} \quad & |Y_{ij}|\leq mZ_{ij}\ \forall i,j\in[n],\ \boldsymbol{0}\leq\boldsymbol{Z}\leq\boldsymbol{E}, \\ & \boldsymbol{P}\succeq 0,\ \mathbb{I}-\boldsymbol{P}\succeq 0,\ \begin{pmatrix} MP & \boldsymbol{X} \\ \boldsymbol{X}^T & M\boldsymbol{P}\end{pmatrix}\succeq 0 \end{aligned} \tag{2.27}$$

is equivalent to the Lasso and nuclear norm regularized problem

$$\min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\boldsymbol{P}\in\mathbb{R}^{n\times n}} \quad \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2+\frac{\rho_1}{M}\|\boldsymbol{X}\|_*+\frac{\rho_2}{m}\|\boldsymbol{Y}\|_1. \tag{2.28}$$

Moreover, as demonstrated by [28, 118] among others, reverse Huber penalties outperform Lasso penalties for sparse regression problems both theoretically—by requiring fewer data to recover the ground truth under a restricted isometry model [118], and empirically—by providing a significantly lower false discovery rate and comparable accuracy rate after observing the same amount of data [29]. This is because Lasso-type penalties are robust estimators but not sparse estimators [13], while reverse Huber penalties are sparse estimators that recover the ground truth after observing slightly more data than via an exact approach [c.f. 4]. Since SLR decomposition is a generalization of sparse regression, this partially explains the superior numerical performance of our alternating minimization method compared to GoDec, as reflected in Section 2.6.

2.5 Branch and Bound

In this section, we propose a branch and bound algorithm in the sense of [92, 98] that computes certifiably (near) optimal solutions to Problem (2.1) in a practical amount of time. Specifically, we state explicitly our subproblem strategy in Section 2.5.1, before stating our overall algorithmic approach in Section 2.5.2. We also provide a sufficient condition for

branch and bound to obtain a globally optimal solution in Section 2.5.2. We remark that branch and bound strategies have previously been leveraged for matrix optimization problems [14, 97].

Let $h(\boldsymbol{Z}, \boldsymbol{P})$ denote the optimal value of the inner minimization problem in (2.18), i.e.:

$$\begin{aligned} h(\boldsymbol{Z}, \boldsymbol{P}) := \min_{\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times n}} \quad & \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda \cdot \|\boldsymbol{X}\|_F^2 + \mu \cdot \|\boldsymbol{Y}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}, \boldsymbol{Y} = \boldsymbol{Z} \circ \boldsymbol{Y}. \end{aligned}$$

Proposition 14 established that solving (2.1) is equivalent to solving $\min_{\boldsymbol{Z} \in \mathcal{Z}_{k_1}, \boldsymbol{P} \in \mathcal{P}_{k_0}} h(\boldsymbol{Z}, \boldsymbol{P})$. In Section 2.4, we illustrated how to obtain a lower bound for the optimal value of (2.1) by solving $\min_{\boldsymbol{Z} \in \text{conv}(\mathcal{Z}_{k_1}), \boldsymbol{P} \in \text{conv}(\mathcal{P}_{k_0})} h(\boldsymbol{Z}, \boldsymbol{P})$ which we formulated as a semidefinite program in (2.20). Suppose we wanted to compute a stronger lower bound for (2.1). Two natural Lagrangean relaxations to consider are:

$$\min_{\boldsymbol{Z} \in \text{conv}(\mathcal{Z}_{k_1}), \boldsymbol{P} \in \mathcal{P}_{k_0}} h(\boldsymbol{Z}, \boldsymbol{P}), \tag{2.29}$$

$$\min_{\boldsymbol{Z} \in \mathcal{Z}_{k_1}, \boldsymbol{P} \in \text{conv}(\mathcal{P}_{k_0})} h(\boldsymbol{Z}, \boldsymbol{P}). \tag{2.30}$$

It is not immediately clear which of these two problems produces a stronger lower bound for (2.1). However, as there does not yet exist an efficient method to branch over the set of $n \times n$ orthogonal projection matrices with trace at most k_0 [19], we focus on developing a branch and bound algorithm that can solve the second problem, (2.30). Moreover, Theorem 12 provides sufficient conditions under which we can exactly compute $\min_{\boldsymbol{P} \in \mathcal{P}_{k_0}} h(\boldsymbol{Z}_0, \boldsymbol{P})$ for any fixed $\boldsymbol{Z}_0 \in \mathcal{Z}_{k_1}$. Thus, provided these conditions hold, we can solve $\min_{\boldsymbol{Z} \in \mathcal{Z}_{k_1}, \boldsymbol{P} \in \mathcal{P}_{k_0}} h(\boldsymbol{Z}, \boldsymbol{P})$ to optimality by branching over the set $\mathcal{Z}_{k_1}$.

2.5.1 Subproblems

We construct an enumeration tree that branches on the entries of the binary matrix $\boldsymbol{Z}$, which models the sparsity pattern of the sparse matrix $\boldsymbol{Y}$. Each node in the tree is defined by a

(partial or complete) sparsity pattern, described by collections $\mathcal{I}_0, \mathcal{I}_1 \subset \{(i,j) : 1 \leq i, j \leq n\}$ where we have $|\mathcal{I}_0| \leq n^2 - k_1$, $|\mathcal{I}_1| \leq k_1$ and $\mathcal{I}_0 \cap \mathcal{I}_1 = \emptyset$, and has an accompanying subproblem. We note that [11] use a similar notion of partially-determined support when developing a custom branch and bound algorithm for the Sparse Principal Component Analysis problem. For indices $(i,j) \in \mathcal{I}_0$, we constrain $Z_{ij} = 0$ and for indices $(i,j) \in \mathcal{I}_1$, we constrain $Z_{ij} = 1$. We say that $\mathcal{I}_0$ and $\mathcal{I}_1$ define a complete sparsity pattern if either $|\mathcal{I}_0| = n^2 - k_1$ or $|\mathcal{I}_1| = k_1$, otherwise we say that $\mathcal{I}_0$ and $\mathcal{I}_1$ define a partial sparsity pattern. A terminal node is a node in the tree that can be described by a complete sparsity pattern.

At any given node in the enumeration defined by collections $\mathcal{I}_0$ and $\mathcal{I}_1$, we consider the subproblem given by:

$$\begin{aligned} \min_{\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times n}} \quad & \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda \cdot \|\boldsymbol{X}\|_F^2 + \mu \cdot \|\boldsymbol{Y}\|_F^2 \\ \text{s.t.} \quad & \text{Rank}(\boldsymbol{X}) \leq k_0, \sum_{(i,j) \notin \mathcal{I}_0 \cup \mathcal{I}_1} \mathbb{1}\{Y_{ij} \neq 0\} \leq k_1 - |\mathcal{I}_1|, \; Y_{ij} = 0 \; \forall (i,j) \in \mathcal{I}_0. \end{aligned} \tag{2.31}$$

This subproblem can equivalently be expressed as

$$\min_{\boldsymbol{Z} \in \mathcal{Z}_{k_1}, \boldsymbol{P} \in \mathcal{P}_{k_0}} \quad h(\boldsymbol{Z}, \boldsymbol{P}) \text{ s.t. } Z_{ij} = 0 \; \forall (i,j) \in \mathcal{I}_0, \; Z_{ij} = 1 \; \forall (i,j) \in \mathcal{I}_1. \tag{2.32}$$

Note that if $\mathcal{I}_0 = \mathcal{I}_1 = \emptyset$, (2.31) and (2.32) are equivalent to (2.1).

Subproblem Upper Bound

We adapt Algorithm 1 to compute feasible solutions to (2.31). Suppose that we fix a sparse matrix $\boldsymbol{Y}^*$ in Problem (2.31). Then, the problem exactly reduces to (2.12), which we know how to solve by Proposition 6. Suppose we fix a low-rank matrix $\boldsymbol{X}^*$ in Problem (2.31).

Then, the problem becomes:

$$\begin{aligned} \min_{\boldsymbol{Y}\in\mathbb{R}^{n\times n}} \quad & \|\tilde{\boldsymbol{D}}-\boldsymbol{Y}\|_F^2+\mu\cdot\|\boldsymbol{Y}\|_F^2 \\ \text{s.t.} \quad & \sum_{(i,j)\notin\mathcal{I}_0\cup\mathcal{I}_1} \mathbb{1}\{Y_{ij}\neq 0\}\leq k_1-|\mathcal{I}_1|,\ Y_{ij}=0\ \forall(i,j)\in\mathcal{I}_0. \end{aligned} \tag{2.33}$$

where $\tilde{\boldsymbol{D}}=\boldsymbol{D}-\boldsymbol{X}^*$ and we have omitted the regularization term on the low-rank matrix because it does not depend on $\boldsymbol{Y}$. Similarly to (2.14), (2.33) admits a closed-form solution:

Proposition 20 *Let $\boldsymbol{Y}^*$ be a matrix such that*

$$\boldsymbol{Y}^*=\boldsymbol{S}^*\circ\left(\frac{\tilde{\boldsymbol{D}}}{1+\mu}\right),$$

where $\boldsymbol{S}^*$ is a $n\times n$ binary matrix with k_1 entries $S^\star_{ij}=1$ such that $S^\star_{ij}=0\ \forall\,(i,j)\in\mathcal{I}_0, S^\star_{ij}=1\ \forall\,(i,j)\in\mathcal{I}_1$ and $S^\star_{i,j}\geq S^\star_{k,l}$ if $|\tilde{D}_{i,j}|\geq|\tilde{D}_{k,l}|\ \forall\,(i,j),(k,l)\notin\mathcal{I}_0\cup\mathcal{I}_1$. Then, $\boldsymbol{Y}^\star$ solves Problem (2.33).

Thus, by replacing the update $\boldsymbol{Y}_t\leftarrow\arg\min_{\boldsymbol{Y}\in\mathcal{W}} f(\boldsymbol{X}_{t-1},\boldsymbol{Y})$ in Algorithm 1 by the update $\boldsymbol{Y}_t\leftarrow\arg\min_{\boldsymbol{Y}\in\bar{\mathcal{W}}} f(\boldsymbol{X}_{t-1},\boldsymbol{Y})$ where $\bar{\mathcal{W}}=\{\boldsymbol{Y}\in\mathbb{R}^{n\times n}:\sum_{ij}\mathbb{1}\{Y_{ij}\neq 0\}\leq k_1-|\mathcal{I}_1|, Y_{ij}=0\ \forall\,(i,j)\in\mathcal{I}_0\}$ using the result of Proposition 20, Algorithm 1 can be readily adapted to obtain high quality feasible solutions to (2.31).

Subproblem Lower Bound

To obtain a lower bound for the objective value of a subproblem given by (2.32), we solve the relaxation given by

$$\min_{\boldsymbol{Z}\in\mathrm{Conv}(\mathcal{Z}_{k_1}),\ \boldsymbol{P}\in\mathrm{Conv}(\mathcal{P}_{k_0})} \quad h(\boldsymbol{Z},\boldsymbol{P})\ \text{s.t.}\ Z_{ij}=0\ \forall(i,j)\in\mathcal{I}_0,\ Z_{ij}=1\ \forall(i,j)\in\mathcal{I}_1. \tag{2.34}$$

From Section 2.4, it follows that (2.34) can be expressed as the following semidefinite problem:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\boldsymbol{P},\boldsymbol{\Theta},\boldsymbol{\alpha}\in\mathbb{R}^{n\times n}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2+\lambda\cdot\operatorname{tr}(\boldsymbol{\Theta})+\mu\cdot\operatorname{tr}\langle\boldsymbol{E},\boldsymbol{\alpha}\rangle \\ \text{s.t.} \quad & \boldsymbol{Y}\circ\boldsymbol{Y}\leq\boldsymbol{\alpha}\circ\boldsymbol{Z},\ \langle\boldsymbol{E},\boldsymbol{Z}\rangle\leq k_1,\ \boldsymbol{0}\leq\boldsymbol{Z}\leq\boldsymbol{E}, \\ & \boldsymbol{P}\succeq 0,\ \mathbb{I}-\boldsymbol{P}\succeq 0,\ \operatorname{tr}(\boldsymbol{P})\leq k_0,\ \begin{pmatrix}\boldsymbol{\Theta} & \boldsymbol{X}\\ \boldsymbol{X}^T & \boldsymbol{P}\end{pmatrix}\succeq 0, \\ & Z_{ij}=0\ \forall(i,j)\in\mathcal{I}_0,\ Z_{ij}=1\ \forall(i,j)\in\mathcal{I}_1. \end{aligned} \tag{2.35}$$

2.5.2 Branch and Bound Algorithm

Having specified the subproblem we consider at each node in the tree and how we compute upper bounds (feasible solutions) and lower bounds by leveraging Algorithm 1 and the convex relaxation given by (2.35), it remains to specify the branching rule and the node selection rule. Algorithm 2 describes our approach. Branching and node selection rules for branch and bound form a rich literature [106]. In our current implementation of Algorithm 2, we employ the most fractional branching rule. Specifically, for an arbitrary non-terminal node p, let $\boldsymbol{Z}^*$ be the optimal matrix $\boldsymbol{Z}$ of the node's convex relaxation given by (2.35). We branch on entry $(i^*,j^*)=\arg\min_{(i,j)\notin\mathcal{I}_0\cup\mathcal{I}_1}|Z_{ij}-0.5|$. When selecting which node to investigate in the tree, we choose a node having a lower bound equal to the current global lower bound. Let $\{(\bar{\boldsymbol{X}}_i,\bar{\boldsymbol{Y}}_i)\}_i$ denote the collection of feasible solutions produced by Algorithm 1 across all nodes that are visited during the execution of Algorithm 2 and let $g(\mathcal{I}_0,\mathcal{I}_1)$ denote the optimal value of Problem (2.35). The final upper bound returned by Algorithm 2 is given by $\min_i f(\bar{\boldsymbol{X}}_i,\bar{\boldsymbol{Y}}_i)$, the smallest objective value achieved by the feasible solution returned by Algorithm 1 for any subproblem explored during the execution of Algorithm 2. The final lower bound returned by Algorithm 2 is given by $\min_{(\mathcal{I}_0,\mathcal{I}_1)\in\mathcal{N}} g(\mathcal{I}_0,\mathcal{I}_1)$ where $\mathcal{N}$ denotes the set of nodes that have not been discarded upon the termination of Algorithm 2.

Theorem 21 *Algorithm 2 terminates in a finite number of iterations and either returns an ϵ globally optimal solution to (2.1) or returns the solution of (2.30).*

Proof To see that Algorithm 2 terminates in a finite number of iterations, it suffices to note that Algorithm 2 can never visit a node more than once and that there is a finite number of partial and complete sparsity patterns (each corresponding to a possible tree node) because the set $\mathcal{Z}_{k_1}$ is discrete.

Upon termination, we must have either $\frac{ub-lb}{ub} \leq \epsilon$ or $|\mathcal{N}| = 0$ (or both). Suppose that $\frac{ub-lb}{ub} \leq \epsilon$. Then, by definition, the output solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ is ϵ globally optimal to problem (2.1) since lb consists of a global lower bound and $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ is feasible to (2.1). Suppose instead that $|\mathcal{N}| = 0$. Algorithm 2 partitions the space of feasible solutions to (2.30) and only discards elements of the partition that are guaranteed not to contain the globally optimal solution. If $|\mathcal{N}| = 0$ upon termination, then Algorithm 2 has explored (or pruned) the entire space of feasible solutions so the output value lb is the optimal objective of (2.30). ■

Theorem 22 *Suppose $\lambda + \frac{2\mu}{1+\mu} - 1 > 0$ and for every full sparsity pattern $\mathcal{I}_0 \subset \{(i,j) : 1 \leq i, j \leq n\}$, $\|\mathcal{I}_0\| = n^2 - k_1$, we have*

$$\gamma_{k_0}(\tilde{\boldsymbol{D}}) < \frac{1}{1+\lambda}\Big(\lambda + \frac{2\mu}{1+\mu} - 1\Big),$$

where $\tilde{\boldsymbol{D}}$ is defined in Theorem 12. Then Algorithm 2 returns an ϵ-optimal solution to (2.1).

Proof Upon termination of Algorithm 2, we must have either $\frac{ub-lb}{ub} \leq \epsilon$ or $|\mathcal{N}| = 0$ (or both). Suppose that $\frac{ub-lb}{ub} \leq \epsilon$. Then, by definition, the output solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ is ϵ globally optimal to problem (2.1). Suppose instead that $|\mathcal{N}| = 0$. Then it must be the case that $ub = lb$. To see this, note that Algorithm 2 partitions the space of feasible solutions to (2.1) and only discards elements of the partition that are guaranteed not to contain the optimal solution. Moreover, at nodes that correspond to complete sparsity patterns, Theorem 12

guarantees that Algorithm 2 computes the exact solution of (2.16). Thus, if $|\mathcal{N}| = 0$ upon termination, Algorithm 2 has explored (or pruned) the entire space of feasible solutions so the output value lb is equal to ub and is the optimal objective of (2.1). ■

Algorithm 2: Near-Optimal SLR Decomposition

Data: $\boldsymbol{D} \in \mathbb{R}^{n\times n}, \lambda, \mu \in \mathbb{R}^+, k_0, k_1 \in \mathbb{Z}^+$. Tolerance parameter $\epsilon \geq 0$.
Result: $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ that solves (2.1) within the optimality tolerance ϵ.
$p_0 \leftarrow (\mathcal{I}_0, \mathcal{I}_1) = (\emptyset, \emptyset)$;
$\mathcal{N} \leftarrow \{p_0\}$;
$(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}) \leftarrow$ solution returned by Algorithm 1;
$ub \leftarrow f(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$;
$lb \leftarrow$ optimal value of (2.20);
while $\frac{ub-lb}{ub} > \epsilon$ *and* $|\mathcal{N}| > 0$ **do**
  select $(\mathcal{I}_0, \mathcal{I}_1) \in \mathcal{N}$;
  select some element $(i, j) \notin \mathcal{I}_0 \cup \mathcal{I}_1$;
  for $k = 0, 1$ **do**
    $l \leftarrow (k + 1) \bmod 2$;
    newnode $\leftarrow \Big(\big(\mathcal{I}_k \cup (i, j)\big), \mathcal{I}_l\Big)$;
    upper $\leftarrow$ upperBound(newnode) with feasible point $(\boldsymbol{X}^*, \boldsymbol{Y}^*)$;
    lower $\leftarrow$ lowerBound(newnode);
    if upper $< ub$ **then**
      $ub \leftarrow$ *upper*;
      $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}) \leftarrow (\boldsymbol{X}^*, \boldsymbol{Y}^*)$;
      remove any node in $\mathcal{N}$ with *lower* $\geq ub$;
    end
    if lower $< ub$ **then**
      add newnode to $\mathcal{N}$
    end
  end
  remove $(\mathcal{I}_0, \mathcal{I}_1)$ from $\mathcal{N}$;
  update lb to be the lowest value of *lower* over $\mathcal{N}$;
end
return $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$, lb

2.6 Computational Results

In this section, we evaluate the performance of our alternating minimization heuristic (Algorithm 1) and our branch and bound method (Algorithm 2) implemented in Julia 1.5.2 using the `JuMP.jl` package version 0.21.7 and solved using `Mosek` version 9.2 for the semidefinite subproblems (2.20). We compare our methods against GoDec given by (2.3), Stable Principal Component Pursuit (S-PCP) given by (2.2), Fast RPCA (fRPCA) [150] , Accelerated Alternating Projections (AccAltProj) [41] and Scaled Gradient Descent (ScaledGD) [133]. All experiments were performed using synthetic data, and run on MIT's Supercloud Cluster [123], which hosts Intel Xeon Platinum 8260 processors. The maximum RAM used across all trials was 192GB. To bridge the gap between theory and practice, we have made our code freely available on GitHub at https://github.com/NicholasJohnson2020/SparseLowRankSoftware. For experiments involving AccAltProj, we employ the MATLAB implementation of the method written by [41] which is available publicly at https://github.com/caesarcai/AccAltProj_for_RPCA/tree/master.

We aim to answer the following questions:

1. How does the performance of Algorithm 1 compare to state-of-the-art convex and non-convex methods such as GoDec, S-PCP, AccAltProj, fRPCA and ScaledGD?

2. How does the performance of the accelerated implementation of Algorithm 1 (described in Section 2.6.4) compare to its exact implementation?

3. How is the performance of Algorithm 1 affected by the dimension of the data matrix $\boldsymbol{D}$, the signal-to-noise level, the rank of the underlying low-rank matrix, and the sparsity of the underlying sparse matrix?

4. How does the performance of Algorithm 2 compare to Algorithm 1?

2.6.1 Synthetic Data Generation

All experiments were performed using synthetic data. To generate a synthetic data matrix $\boldsymbol{D}$, we first fix a problem dimension n, a desired rank for the low-rank matrix k_0, a desired sparsity for the sparse matrix k_1 and a value $\sigma > 0$ that controls the signal to noise ratio. Next, we generate a random rank k_0 matrix and k_1 sparse matrix. To generate the low-rank matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$, we set $\boldsymbol{L} = \boldsymbol{V}\boldsymbol{V}^T$ where $\boldsymbol{V} \in \mathbb{R}^{n \times k_0}$ and $V_{ij} \sim N(0, \frac{\sigma^2}{n})$. To generate the sparse matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, we randomly select a symmetric set of indices $\mathcal{S} \subset \{(i,j) : 1 \leq i, j \leq n\}$ with cardinality $|\mathcal{S}| = k_1$ and let $S_{ij} \sim U(-5, 5)$ if $(i,j) \in \mathcal{S}$ and $S_{ij} = 0$ otherwise. Finally, we set $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S} + \boldsymbol{N}$ where $N_{ij} = N_{ji} \sim N(0,1)$. Note that this data generation process is similar to that employed by [44].

2.6.2 Hyperparameter Tuning

We tune the hyperparameters of Algorithm 1, fRPCA, and ScaledGD using 30-fold cross-validation, as proposed by [114]. For each fold, we randomly sample l columns and rows from the input data matrix $\boldsymbol{D}$ and permute the columns and rows of $\boldsymbol{D}$ to obtain $\tilde{\boldsymbol{D}} = \begin{pmatrix} \boldsymbol{D}_{val} & \boldsymbol{D}_{UR} \\ \boldsymbol{D}_{LL} & \boldsymbol{D}_{train} \end{pmatrix}$ where $\boldsymbol{D}_{val} \in \mathbb{R}^{l \times l}$ is the submatrix corresponding to the randomly sampled rows and columns of $\boldsymbol{D}$, $\boldsymbol{D}_{train} \in \mathbb{R}^{(n-l)\times(n-l)}$, and $\boldsymbol{D}_{UR}, \boldsymbol{D}_{LL}^T \in \mathbb{R}^{l \times (n-l)}$. We set $l = \lfloor n \cdot (1 - \sqrt{0.7}) \rfloor$ so that the training set $\boldsymbol{D}_{train}$ contains at least 70% of the input data. For a given choice of hyperparameters, we perform a SLR decomposition on $\boldsymbol{D}_{train}$. Letting $\hat{\boldsymbol{X}}$ denote the estimated low-rank matrix, we compute the validation score for a single fold as $\frac{\|\boldsymbol{D}_{val} - \boldsymbol{D}_{UR}\hat{\boldsymbol{X}}^\dagger \boldsymbol{D}_{LL}\|_F{}^2}{\|\boldsymbol{D}_{val}\|_F{}^2}$. The final validation score for a given set of hyperparameters is the average over 30 folds.

For experiments reported in Section 2.6.3 and Section 2.6.4, we tune the hyperparameters (λ, μ) for Algorithm 1 from the collection $\left(\frac{10^{-2}}{\sqrt{n}}, \frac{10^{-1}}{\sqrt{n}}, \frac{10^{0}}{\sqrt{n}}, \frac{10^{1}}{\sqrt{n}}\right) \times \left(\frac{10^{-2}}{\sqrt{n}}, \frac{10^{-1}}{\sqrt{n}}, \frac{10^{0}}{\sqrt{n}}, \frac{10^{1}}{\sqrt{n}}\right)$ and we set the hyperparameter $\gamma = \alpha \frac{k_1}{n^2}$ for fRPCA and ScaledGD where α is tuned (in-

dependently for each method) from the collection $(0.01, 0.05, 0.1, 0.5, 1, 2, 4, 6, 8, 10)$. For subsequent experiments in Section 2.6.1 and beyond, the hyperparameters of Algorithm 1, fRPCA, and ScaledGD are fixed respectively to the best-performing hyperparameters selected via cross-validation in Section 2.6.3 and Section 2.6.4. For experiments employing Algorithm 2, we set $\lambda = \mu = \frac{1}{\sqrt{n}}$. We terminate Algorithm 1, GoDec, fRPCA, and ScaledGD when $\frac{f_{t-1}-f_t}{f_t} < 0.001$ where f_t denotes the objective value achieved by the estimate of the low-rank matrix $\boldsymbol{X}$ and the sparse matrix $\boldsymbol{Y}$ at iteration t.

2.6.3 A Comparison Between the Performance of Algorithm 1, GoDec, S-PCP, AccAltProj, fRPCA and ScaledGD

We present a comparison of Algorithm 1, GoDec, S-PCP, AccAltProj, fRPCA, and ScaledGD as we vary the dimension n of the input data matrix $\boldsymbol{D}$, the rank k_0 of the underlying low-rank matrix $\boldsymbol{L}$ and the sparsity level k_1 of the underlying sparse matrix $\boldsymbol{S}$. We report results for the exact implementations of Algorithm 1 ("Alg 1 Exact") and GoDec where the singular value decomposition is computed exactly at each step. We fix $\sigma = 10$ across all trials. For each value of (n, k_0, k_1), we perform 10 trials.

In Table 2.2, we report the low-rank matrix reconstruction error (L Error) of each method and the rank and sparsity of the solution returned by S-PCP. Let $\hat{\boldsymbol{L}}$ denote the low-rank matrix returned by one of the five methods. We define the low-rank matrix reconstruction error to be $\frac{\|\hat{\boldsymbol{L}}-\boldsymbol{L}\|_F^2}{\|\boldsymbol{L}\|_F^2}$. Let $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{S}}$ denote the low-rank and sparse matrices returned by S-PCP. We define the rank of a solution returned by S-PCP to be $\sum_{i=1}^{n} \mathbb{1}\{\sigma_i(\hat{\boldsymbol{L}}) > 10^{-2}\}$, the number of singular values of $\hat{\boldsymbol{L}}$ that are greater than 10^{-2}. Similarly, we define the sparsity of a solution returned by S-PCP to be $\sum_{ij} \mathbb{1}\{\hat{S}_{ij} > 10^{-2}\}$, the number of entries of $\hat{\boldsymbol{S}}$ that are greater than 10^{-2}.

For every parameter configuration explored, Algorithm 1 outperforms all benchmark methods by producing a solution that has a comparable although slightly lower low-rank matrix reconstruction error and a lower sparse matrix reconstruction error. Moreover, the

solutions returned by S-PCP always have an average rank that is far greater than the target rank k_0 and a sparsity level that is far greater than the target sparsity level k_1. Further, the numerical threshold used to compute the rank and sparsity of S-PCP solutions, 10^{-2}, is quite generous. Indeed, using a more common, more restrictive threshold for numerical tolerance would further amplify this discrepancy.

In Table 2.3, we report the low-rank matrix reconstruction error of each method, the bound gap between the solution returned by Algorithm 1 and the solution of (2.20), and the time required to solve (2.20). Letting $\hat{f}$ denote the objective value achieved by the solution returned by Algorithm 1 and letting f^* denote the optimal value of (2.20), we define the bound gap as $\frac{\hat{f}-f^*}{\hat{f}}$. Thus, not only does Algorithm 1 outperform S-PCP, GoDec, fRPCA and ScaledGD, but, by using the relaxation given by (2.20), we obtain a certificate of Algorithm 1's instance-wise quality.

2.6.4 An Accelerated Implementation of Algorithm 1 and its Performance

As noted in Section 2.3, the main bottleneck in our implementation of Algorithm 1 is the singular value decomposition step that must be performed at each iteration. One commonly proposed technique in the literature to circumvent this difficulty is to employ a randomized SVD [c.f. 80], which computes a low-rank matrix less accurately but in significantly less time than via an exact SVD. Accordingly, in this section, we investigate the use of a randomized SVD in Algorithm 1 ("Alg 1 Acc") against an exact SVD step ("Alg 1 Exact"). In the accelerated implementation of Algorithm 1, we compute a randomized SVD at every iteration except the final one, where we employ an exact SVD.

We now present a comparison of the exact and accelerated implementations of Algorithm 1 as we vary the dimension n of the input data matrix $\boldsymbol{D}$, the rank k_0 of the underlying low-rank matrix $\boldsymbol{L}$ and the sparsity level k_1 of the underlying sparse matrix $\boldsymbol{S}$. We fix $\sigma = 10$ across all trials. For each value of (n, k_0, k_1), we performed 10 trials.

In Table 2.4, we report the low-rank matrix reconstruction error and the execution time of the exact and accelerated implementations of Algorithm 1. The execution time reported is the average total runtime of each method which includes the time required to perform cross-validation for the hyperparameters λ and μ. The exact implementation of Algorithm 1 produces a lower reconstruction error than the accelerated implementation across all trials. This behavior is expected given that at each iteration, the exact implementation of Algorithm 1 solves the low-rank subproblem (2.12) to optimality, whereas the accelerated implementation only computes a high quality solution to this subproblem (except at the last step). Further, across all trials, the accelerated implementation of Algorithm 1 has a faster average execution time than the exact implementation, which is consistent with the $O(n^2 \log k)$ complexity of the low-rank update in the accelerated implementation compared to the $O(n^2 k)$ complexity in the exact implementation.

2.6.5 Scalability of Algorithm 1

We present a comparison of Algorithm 1 with GoDec, AccAltProj and ScaledGD as we vary the dimension of the input data matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$. We report results for the exact implementations of Algorithm 1 and GoDec. For the first experiment, we fixed $k_0 = 5$, $k_1 = 500$, $\sigma = 10$ across all trials, considered values of $n \in \{200, 250, 300, ..., 1000\}$, and performed 50 trials for each n. For the second experiment, we fixed $k_0 = 2$, $k_1 = 500$, $\sigma = 10$, considered values of $n \in \{2000, 4000, ..., 10000\}$, and performed 5 trials for each n. We fixed the hyperparameters $(\lambda, \mu) = \left(\frac{0.1}{\sqrt{n}}, \frac{10}{\sqrt{n}}\right)$ (resp. $\gamma = \frac{k_1}{2n^2}$) for Algorithm 1 (resp. ScaledGD) for these and all subsequent experiments.

We report the low-rank matrix reconstruction error, the sparse matrix reconstruction error, the sparse support discovery rate, and the execution time for each method in Figures 2.1–2.2. We additionally report the low-rank matrix reconstruction error, the sparse matrix reconstruction error and the execution time for Algorithm 1, GoDec and ScaledGD in Table 2.5 of Section 2.11. Let $\hat{\boldsymbol{S}}$ denote the sparse matrix returned by either Algorithm 1 or GoDec. We define the sparse matrix reconstruction error analogously to the low-rank matrix

reconstruction error as $\frac{\|\hat{\boldsymbol{S}}-\boldsymbol{S}\|_F^2}{\|\boldsymbol{S}\|_F^2}$. Let $\mathcal{I}(\boldsymbol{S}) = \{(i,j) : S_{ij} \neq 0\}$ denote the support of the sparse matrix $\boldsymbol{S}$, i.e., the set of indices for which the matrix $\boldsymbol{S}$ takes non zero values. Then, we define the sparse support discovery rate to be $\frac{1}{k_1}\sum_{(i,j)\in\mathcal{I}(\boldsymbol{S})} \mathbb{1}(\hat{S}_{ij} \neq 0)$. The execution time reported is the average runtime for a single trial of a given method. We note that if AccAltProj were implemented in Julia, it would very likely exhibit more favorable runtimes than its publicly available MATLAB implementation [30]. The performance metric of greatest interest is the low-rank matrix reconstruction error followed by the sparse matrix reconstruction error.

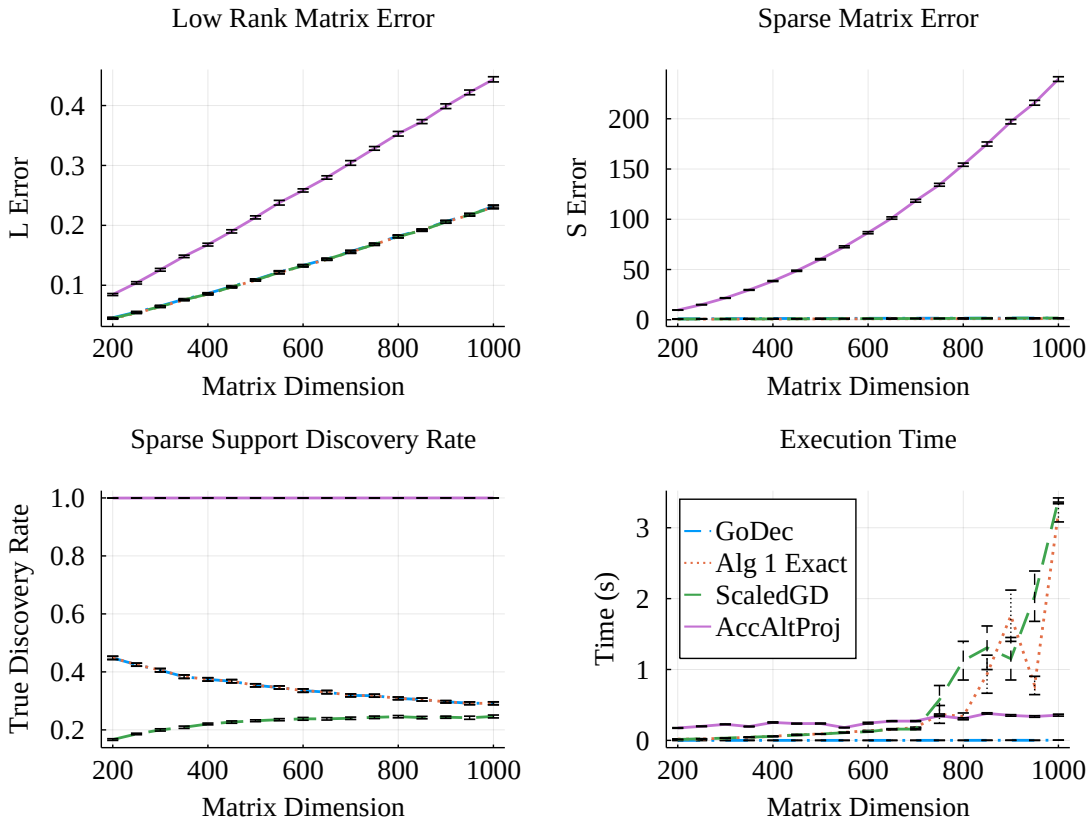

Figure 2.1: Low-rank matrix reconstruction error (top left), sparse matrix reconstruction error (top right), sparse support discovery rate (bottom left) and execution time (bottom right) versus n with $k_0 = 5$, $k_1 = 500$ and $\sigma = 10$. Averaged over 50 trials for each parameter configuration.

Our main findings from this set of experiments are:

1. Algorithm 1 outperforms GoDec, AccAltProj and ScaledGD across most trials by obtaining lower sparse and low-rank reconstruction errors, while having a comparable

execution time.

2. The low-rank matrix reconstruction error scales linearly with matrix dimension for Algorithm 1, AccAltProj, ScaledGD, and GoDec. It can be shown that for our data generation process, $\lim_{n\to\infty} \mathbb{E}[\|\boldsymbol{L}\|_F^2] = C(k_0, \sigma)$ where $C(k_0, \sigma)$ is a constant that depends only on the rank of $\boldsymbol{L}$ and the signal-to-noise level. This implies that for all methods, $\mathbb{E}[\|\hat{\boldsymbol{L}} - \boldsymbol{L}\|_F^2]$ is $\Theta(n)$.

3. The sparse matrix reconstruction error appears to scale linearly with matrix dimension for Algorithm 1, ScaledGD, and GoDec, while scaling superlinearly with the matrix dimension for AccAltProj. Note that AccAltProj does not allow the cardinality of the sparse matrix to be explicitly constrained. Accordingly, AccAltProj tends to return a sparse matrix that is considerably denser than the desired level. This produces a high sparse support discovery rate (true positive rate) at the expense of a high false discovery rate.. The sparse support discovery rate declines as the matrix dimension increases for GoDec and Algorithm 1in the regime investigated in Figure 2.1. ScaledGD underperforms GoDec and Algorithm 1 with respect to sparse support discovery rate in low-dimensional settings (Figure 2.1) but outperforms in high-dimensional settings (Figure 2.2). This is to be expected as with increasing matrix dimension while k_1 is held fixed, it becomes increasingly difficult to identify the underlying sparsity pattern.

2.6.6 Sensitivity to Noise

We present a comparison of Algorithm 1 with GoDec, AccAltProj and ScaledGD as we vary the signal to noise level σ of the input data matrix $\boldsymbol{D}$. Large values of σ correspond to a greater signal in the low-rank matrix $\boldsymbol{L}$ compared to the perturbation matrix $\boldsymbol{N}$. We report results for the exact implementations of Algorithm 1 and GoDec that exactly compute the singular value decomposition step. We fixed $n = 100$, $k_0 = 5$, $k_1 = 500$ across all trials and considered values of $\sigma \in \{1, 2, 3, ..., 30\}$. For each value of σ, we performed 50 trials.

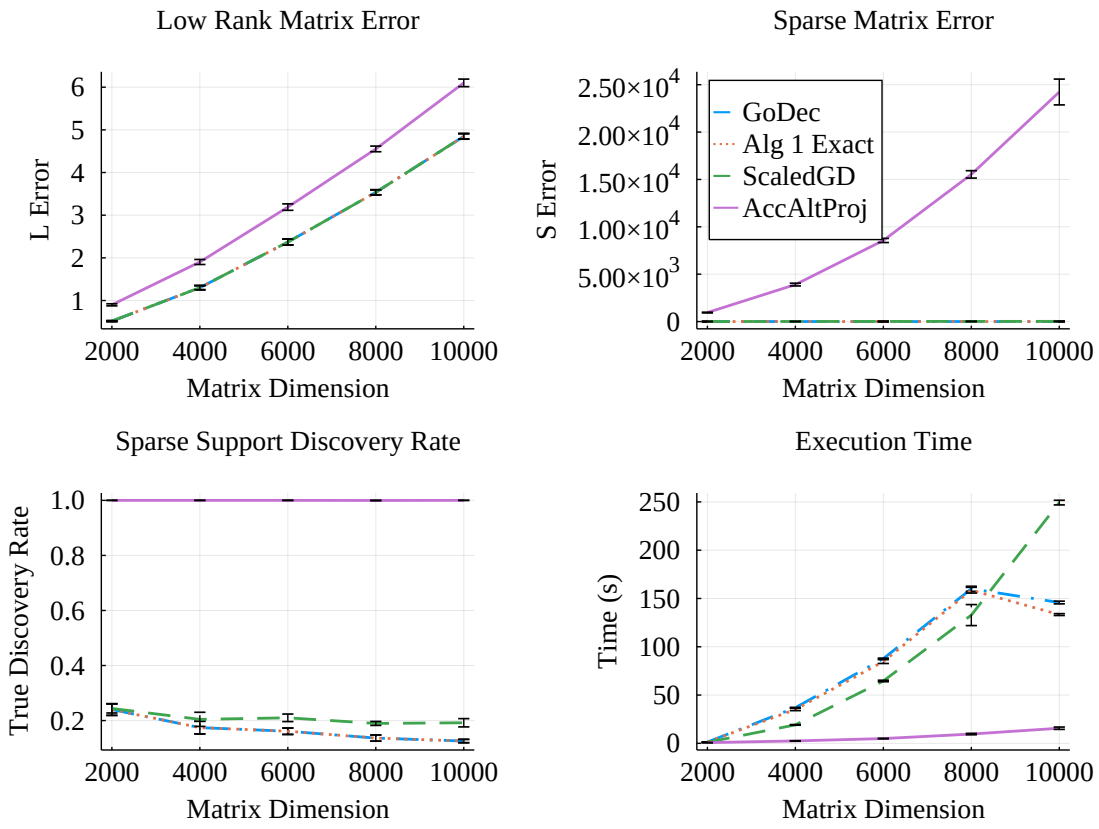

Figure 2.2: Low-rank matrix reconstruction error (top left), sparse matrix reconstruction error (top right), sparse support discovery rate (bottom left) and execution time (bottom right) versus n with $k_0 = 2$, $k_1 = 500$ and $\sigma = 10$. Averaged over 5 trials for each parameter configuration.

We report the low-rank matrix reconstruction error, the sparse matrix reconstruction error, the sparse support discovery rate, and the execution time for each method in Figure 2.3. Figure 2.3 includes only results for values of $\sigma \in [10, 30]$ to aid visualization due to significant differences in scale between these results and those for $\sigma \in [1, 10]$. We report the results for the full range $\sigma \in [1, 30]$ in Figure 2.8 of Section 2.11.

Our main findings from this set of experiments are:

1. Consistent with previous experiments, Algorithm 1 outperforms GoDec, AccAltProj and ScaledGD across most trials by obtaining a lower sparse and low-rank matrix reconstruction error while maintaining a comparable execution time and exhibiting superior sparse support discovery rates (compared to GoDec and ScaledGD). The su-

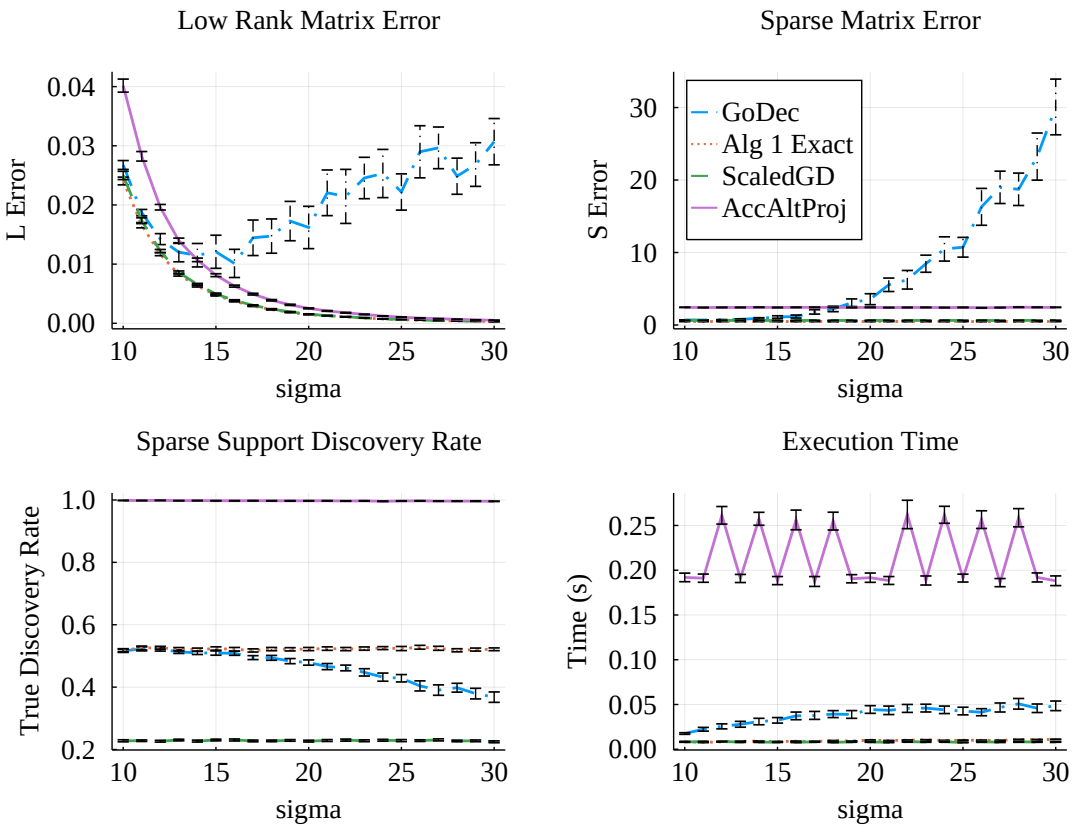

Figure 2.3: Low-rank matrix reconstruction error (top left), sparse matrix reconstruction error (top right), sparse support discovery rate (bottom left) and execution time (bottom right) versus σ with $n = 100$, $k_0 = 5$ and $k_1 = 500$. Averaged over 50 trials for each parameter configuration.

perior performance of Algorithm 1 relative to GoDec becomes more extreme as the signal-to-noise ratio increases.

2. The low-rank reconstruction error of Algorithm 1 decreases as σ increases. This is consistent with the intuition that larger values of σ correspond to easier problem instances, so it should be easier to recover the low-rank matrix. Further, the plotted trend suggests that should σ be further increased, Algorithm 1 would exactly recover $\boldsymbol{L}$. Somewhat surprisingly, the performance of GoDec appears to break down at higher levels of σ. The sparse matrix reconstruction error of Algorithm 1 also declines as σ increases, whereas that of GoDec again breaks down. ScaledGD exhibits a poor sparse recovery rate in these experiments.

3. The sparse support discovery rate of Algorithm 1 slightly declines as σ increases, whereas that of GoDec drops sharply. Though one might expect the sparse support discovery rate to increase with the signal-to-noise level, recall that σ controls the signal-to-noise level of the low-rank matrix compared to the noise matrix and not that of the sparse matrix. Consequently, as σ increases, it should become easier to recover the low-rank matrix but more difficult to recover the sparse matrix.

2.6.7 Sensitivity to Rank

We present a comparison of Algorithm 1 with GoDec, AccAltProj and ScaledGD as we vary the rank k_0 of the underlying low-rank matrix $\boldsymbol{L}$. We report results for the exact implementations of Algorithm 1 and GoDec that exactly compute the singular value decomposition step. We fixed $n = 100$, $k_1 = 500$, $\sigma = 10$ across all trials and considered values of $k_0 \in \{2, 4, 6, ..., 50\}$. For each value of k_0, we performed 50 trials.

We report the low-rank matrix reconstruction error, the sparse matrix reconstruction error, the sparse support discovery rate, and the runtime for each method in Figure 2.4.

Our main findings from this set of experiments are:

1. Consistent with previous experiments, Algorithm 1 outperforms GoDec, AccAltProj and ScaledGD across all trials by obtaining a lower low-rank matrix reconstruction error and sparse matrix reconstruction error while having a lesser (in the case of GoDec and AccAltProj) or comparable (in the case of ScaledGD) execution time and exhibiting superior sparse support discovery rates than GoDec and ScaledGD. The superior performance of Algorithm 1 becomes more extreme as the rank increases.

2. The low-rank reconstruction error of Algorithm 1 and that of AccAltProj decrease as k_0 increases whereas the low-rank reconstruction error of GoDec increases with increasing k_0 and that of ScaledGD remains roughly constant.

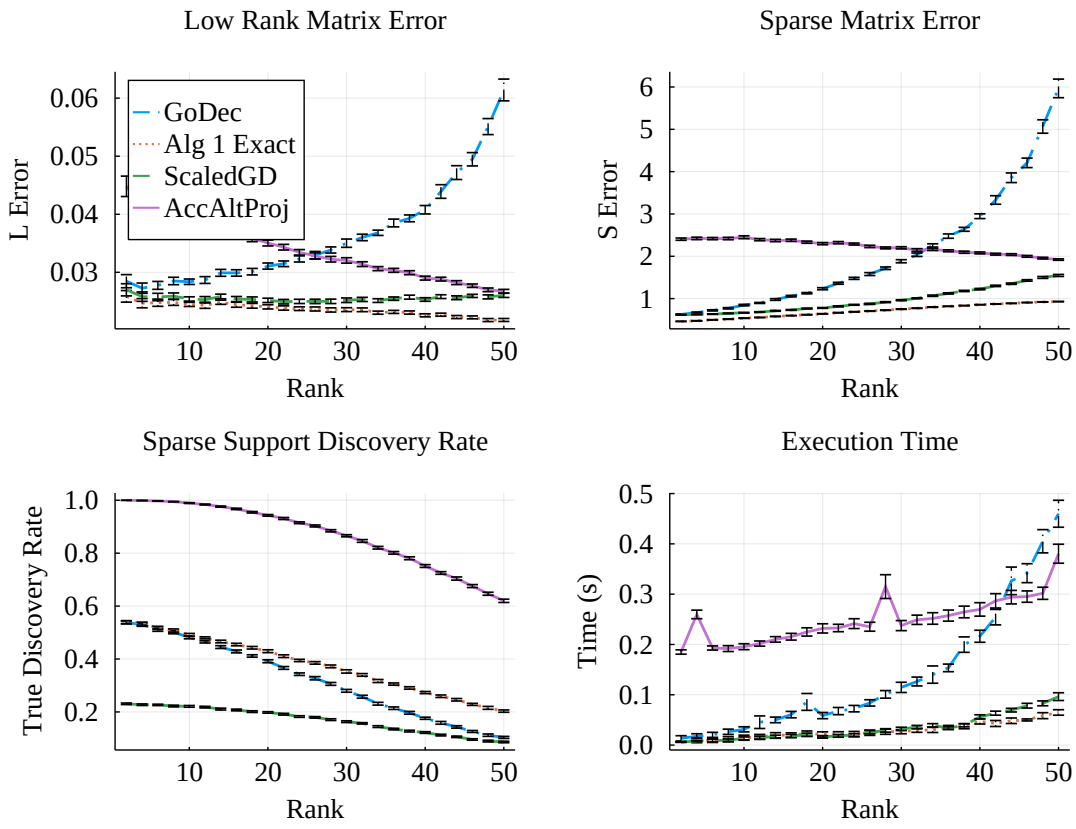

Figure 2.4: Low-rank matrix reconstruction error (top left), sparse matrix reconstruction error (top right), sparse support discovery rate (bottom left) and execution time (bottom right) versus k_0 with $n = 100$, $k_1 = 500$ and $\sigma = 10$. Averaged over 50 trials for each parameter configuration.

3. Algorithm 1's and ScaledGD's sparse matrix reconstruction error increases slightly, while GoDec's error increases significantly and AccAltProj's decreases slightly.

2.6.8 Sensitivity to Sparsity

We present a comparison of Algorithm 1 with GoDec, AccAltProj and ScaledGD as we vary the sparsity level k_1 of the underlying sparse matrix $\boldsymbol{S}$. We report results for the exact implementations of Algorithm 1 and GoDec that exactly compute the singular value decomposition step. We fixed $n = 100$, $k_0 = 5$, $\sigma = 10$ across all trials and considered values of $k_1 \in \{50, 100, 150, ..., 1000\}$. For each value of k_1, we performed 50 trials.

We report the low-rank matrix reconstruction error, the sparse matrix reconstruction

error, the sparse support discovery rate, and the runtime for each method in Figure 2.5.

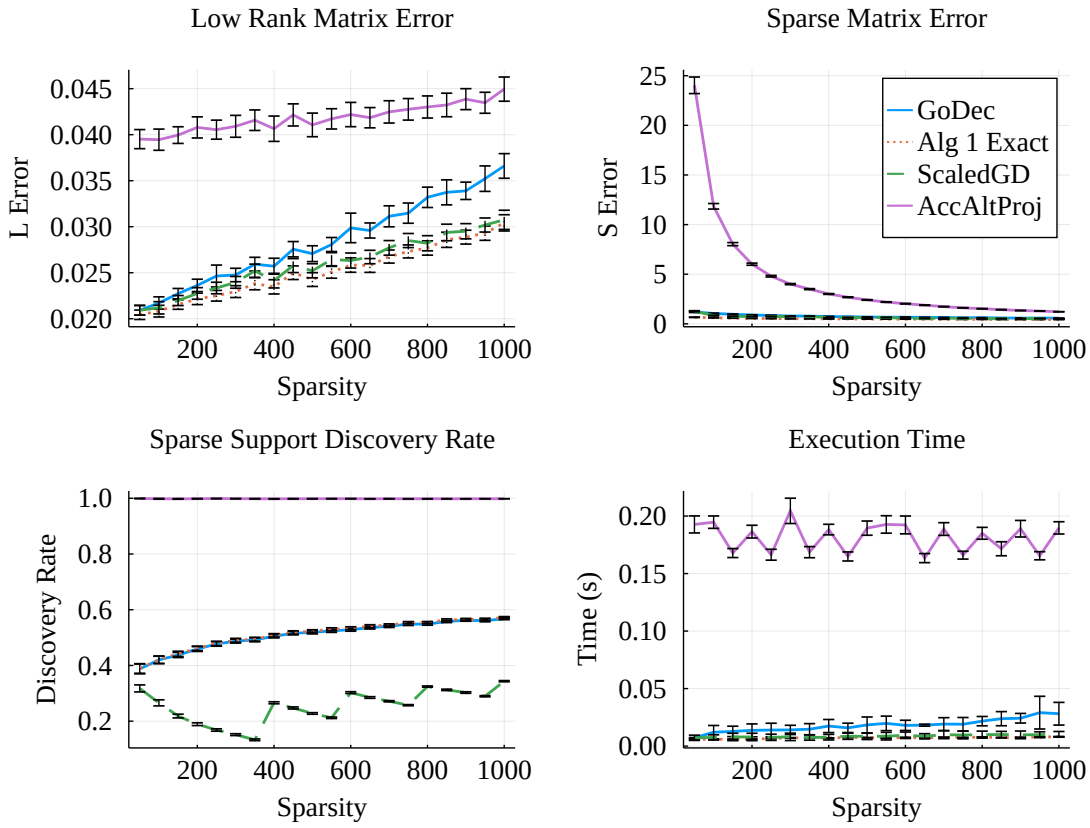

Figure 2.5: Low-rank matrix reconstruction error (top left), sparse matrix reconstruction error (top right), sparse support discovery rate (bottom left) and execution time (bottom right) versus k_1 with $n = 100$, $k_0 = 5$ and $\sigma = 10$. Averaged over 50 trials for each parameter configuration.

Our main findings from this set of experiments are:

1. Consistent with previous experiments, Algorithm 1 outperforms GoDec, AccAltProj and ScaledGD across all trials by obtaining a lower low-rank matrix reconstruction error and sparse matrix reconstruction error while having a lesser execution time. Algorithm 1 also exhibits a superior accuracy rate than GoDec and ScaledGD.

2. The low-rank reconstruction error of Algorithm 1, GoDec, AccAltProj and ScaledGD increase as k_1 increases. This is consistent with the intuition that as the sparsity of the underlying spare matrix increases, it becomes more difficult to identify the true low-rank matrix.

Table 2.1: Performance of Algorithm 2 for $\epsilon = 0.05$. Reported root node gap is a percentage.

N	k_0	k_1	Root Node Gap	Nodes Explored	Time (s)
10	1	10	5.66	3	41
10	1	15	2.94	1	43
10	2	20	2.37	1	43
15	1	22	7.34	33	58
15	2	33	5.08	3	47
15	3	45	3.26	1	40
20	1	20	5.48	5	44
20	2	40	6.44	123	126
20	3	60	4.33	1	40
20	4	80	4.15	1	41
25	1	31	7.43	205	479
25	2	62	8.30	14709	28977
25	3	93	6.60	1053	2485
25	5	125	7.50	653	1631

3. The sparse matrix reconstruction error of Algorithm 1, ScaledGD, AccAltProj and GoDec decline as k_1 increases.

2.6.9 Performance of Algorithm 2

We report the performance of Algorithm 2 on several problem instances. In these experiments, calls that Algorithm 2 make to Algorithm 1 employ the exact implementation of Algorithm 1. We fix $\sigma = 10$ and set $\epsilon = 0.05$, meaning that Algorithm 1 terminates when it has computed a solution to (2.1) that is certifiably within 5% of the globally optimal solution. We report the optimality gap between the root node upper bound and the root node lower bound, the total number of nodes explored, and the execution time of Algorithm 2 for 14 problem instances in Table 2.1.

As expected, when the root node optimality gap is less than ϵ, no additional nodes are explored. The total number of possible terminal nodes in any branch and bound instance is equal to the number of distinct sparsity patterns, given by $\binom{n^2}{k_1}$. This implies that the total number of possible nodes in any branch and bound instance is given by $2 \cdot \binom{n^2}{k_1} - 1$. In

the case of the last instance given in Table 2.1, this quantity is roughly equal to 5.3×10^{134}. Thus, the results of Table 2.1 indicate that Algorithm 2 is able to prune the vast majority of possible nodes in the branch and bound tree. We note that the execution time explodes as the number of nodes explored increases. One of the main limitations of the current implementation of Algorithm 2 is that it requires solving (2.35), a semidefinite optimization problem, at every node that is explored. This becomes a computational bottleneck as the most efficient interior point solvers for SDPs exhibit poor scaling.

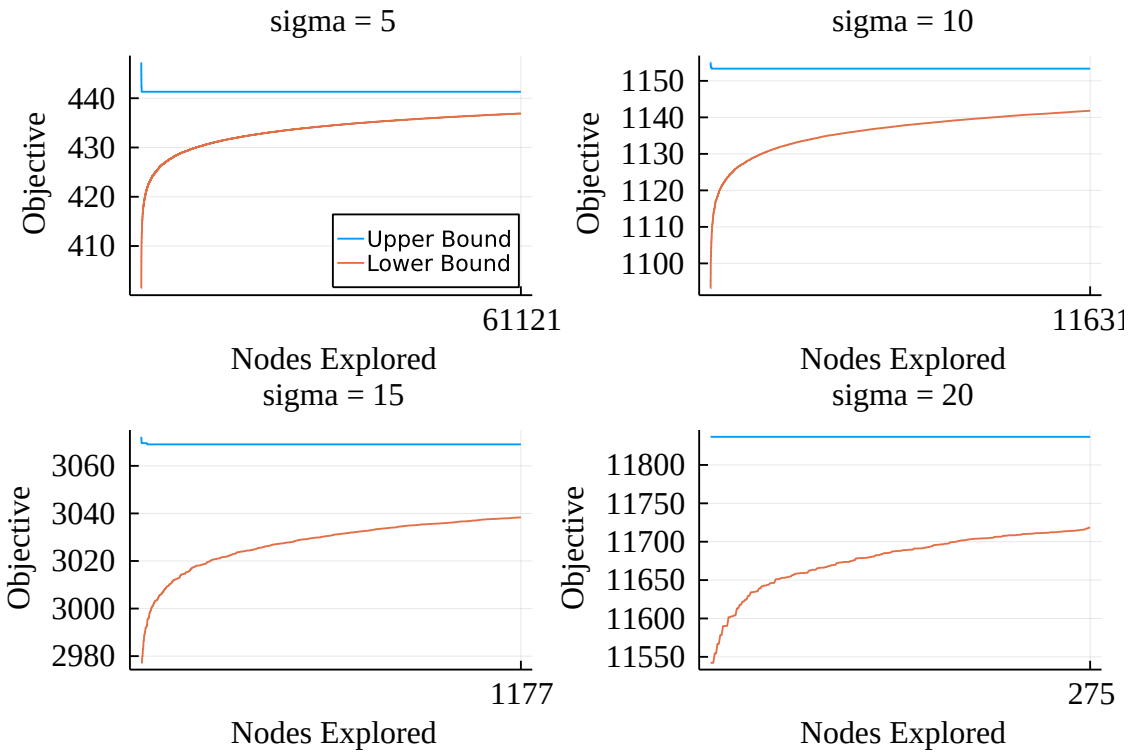

Figure 2.6: Algorithm 2 upper and lower bound evolution (for a single instance) for $\sigma = 5$ (top left), $\sigma = 10$ (top right), $\sigma = 15$ (bottom left) and $\sigma = 20$ (bottom right) with $n = 15$, $k_0 = 1$, $k_1 = 22$ and $\epsilon = 0.01$.

Figure 2.6 illustrates that Algorithm 2 only occasionally updates the global upper bound and that the vast majority of computational time is spent certifying optimality. This behavior is consistent across all problem instances in which the root node upper bound is not already ϵ optimal. Moreover, Figure 2.7 illustrates that Algorithm 2 successfully solves instances where $n = 15$ for all values of σ, and is fastest when there is the least amount of noise.

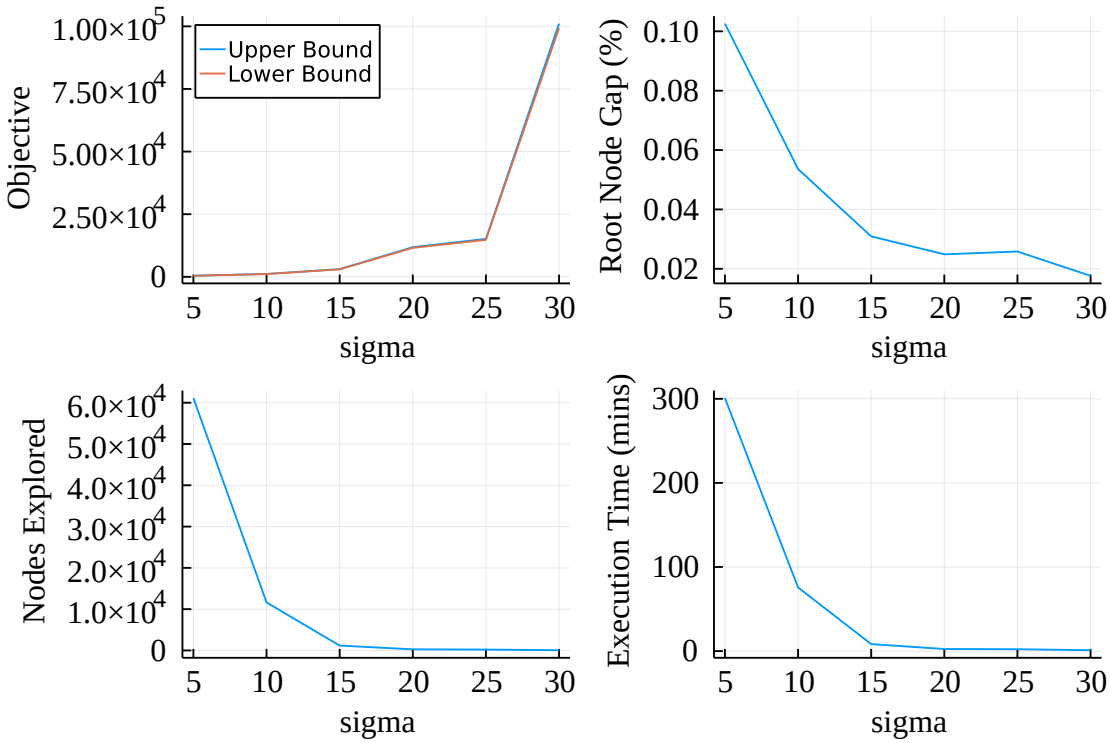

Figure 2.7: Algorithm 2 root node upper and lower bound (top left), root node optimality gap (top right), number of nodes explored (bottom left) and execution time (bottom right) versus σ with $n = 15$, $k_0 = 1$, $k_1 = 50$ and $\epsilon = 0.01$.

2.6.10 Summary of Findings From Numerical Experiments

We are now in a position to answer the four questions introduced at the start of this section. Our findings are as follows:

1. Algorithm 1 outperforms GoDec across all trials by obtaining a lower low-rank matrix reconstruction error and sparse matrix reconstruction error while having a lesser execution time and exhibiting superior sparse support discovery rates. The superior performance of Algorithm 1 is most extreme in regimes where the signal-to-noise level σ is high and separately when the rank k_0 of the underlying low-rank matrix is high. Further, Algorithm 1 outperforms S-PCP, AccAltProj and fRPCA across all trials by obtaining lower low-rank and sparse matrix reconstruction errors. With cross-validation, Algorithm 1 obtains low-rank matrices with a lower rank and a comparable

reconstruction error than ScaledGD, and with a rank constraint on both methods it obtains a lower low-rank error that ScaledGD on all but 3 trials. Moreover, it always achieves a lesser sparse matrix reconstruction error than ScaledGD.

2. The exact implementation of Algorithm 1 outperforms the accelerated implementation by achieving a lower reconstruction error across all trials. However, across all trials, the accelerated implementation of Algorithm 1 has a faster average execution time than the exact implementation.

3. (a) Increasing the matrix dimension n results in linear increases in the low-rank matrix reconstruction error and the sparse matrix reconstruction error for Algorithm 1, GoDec and ScaledGD. Increasing the matrix dimension n results in a linear increase in the low-rank matrix reconstruction error and a superlinear increase in the sparse matrix reconstruction error for AccAltProj. The sparse support discovery rate decreases with n for Algorithm 1 and GoDec while the execution time of each method scales superlinearly with n.

   (b) The low-rank matrix and sparse matrix reconstruction errors of Algorithm 1, AccAltProj and ScaledGD decrease with increasing values of σ and that of Algorithm 1 appears to converge towards 0. The sparse support discovery rate of Algorithm 1 decreases slightly with σ while its execution time remains roughly constant. Conversely, the low-rank matrix and sparse matrix reconstruction errors of GoDec explode for large values of σ. GoDec's sparse support discovery rate declines sharply in the high signal-to-noise level regime. ScaledGD generally has poor sparse support discovery.AccAltProj tends to exhibit high sparse support discovery rate because the sparse matrix selected by AccAltProj is in general substantially more dense than the ground truth sparse matrix.

   (c) Increasing the rank of the low-rank matrix results in a slight decrease in the low-rank matrix reconstruction error and a slight increase in the sparse matrix

reconstruction error for Algorithm 1 and ScaledGD. In contrast, the low-rank matrix and sparse matrix reconstruction errors grow superlinearly for GoDec with increasing rank. The sparse support discovery rate , of Algorithm 1, GoDec and ScaledGD, and the execution time of all methods grow with increasing rank.

(d) Algorithm 1, ScaledGD and GoDec exhibit similar behaviour as a function of sparsity k_1. As the sparsity level of the underlying sparse matrix increases, the low-rank matrix reconstruction error, sparse support discovery rate, and execution time of each of these methods increase while the sparse matrix reconstruction error decreases.

4. Algorithm 2 solves (2.1) to certifiable optimality for small problem instances (up to $n = 25$) in reasonable wall clock time. The majority of Algorithm 2's execution time is spent certifying optimality. This implies that the final solution returned by Algorithm 2 is, in general, only marginally better than the solution returned by Algorithm 1.

2.7 Concluding Remarks

In this chapter, we introduced a novel formulation (2.1) for SLR that exploits discreteness and leverages regularization. We presented Algorithm 1, an alternating minimization heuristic that can compute high quality feasible solutions to (2.1) and can scale to $n = 10000$ in minutes. We developed a strong semidefinite relaxation (2.20) that can certify the quality of the solutions returned by Algorithm 1. Finally, we presented Algorithm 2, a branch and bound method that solves (2.1) to certifiable near-optimality and scales to $n = 25$ in minutes. Moreover, we established sufficient conditions under which Algorithm 2 is optimal. Further work could focus on increasing the scalability of our branch and bound method. When executing Algorithm 2, a semidefinite optimization problem must be solved at every node in the branch and bound tree to compute a lower bound. This computation is quite costly. A possible extension would be to compute a second-order cone lower bound at each

node which would be more scalable at the expense of being less tight. Algorithm 2 can also potentially be further improved by adopting an alternate branching rule.

2.8 Appendix: SLR Formulation Properties Omitted Proofs

Recall that Proposition 1 states that $f(\boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2$ is jointly m-strongly convex in $(\boldsymbol{X}, \boldsymbol{Y})$. We prove this fact below:

Proof Consider any two points $(\boldsymbol{X}_1, \boldsymbol{Y}_1), (\boldsymbol{X}_2, \boldsymbol{Y}_2) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$ and any $t \in [0, 1]$. We have

$$
\begin{aligned}
g(t\boldsymbol{X}_1 + (1-t)\boldsymbol{X}_2,\, t\boldsymbol{Y}_1 + (1-t)\boldsymbol{Y}_2) &= \|\boldsymbol{D} - t\boldsymbol{X}_1 + (1-t)\boldsymbol{X}_2 - t\boldsymbol{Y}_1 + (1-t)\boldsymbol{Y}_2\|_F^2 + \\
&\quad (\lambda - \min(\lambda,\mu))\|t\boldsymbol{X}_1 + (1-t)\boldsymbol{X}_2\|_F^2 + (\mu - \min(\lambda,\mu))\|t\boldsymbol{Y}_1 + (1-t)\boldsymbol{Y}_2\|_F^2 \\
&\quad (\lambda - \min(\lambda,\mu))\|t\boldsymbol{X}_1 + (1-t)\boldsymbol{X}_2\|_F^2 + (\mu - \min(\lambda,\mu))\|t\boldsymbol{Y}_1 + (1-t)\boldsymbol{Y}_2\|_F^2 \\
&\leq t \cdot \left[\|\boldsymbol{D} - \boldsymbol{X}_1 - \boldsymbol{Y}_1\|_F^2 + (\lambda - \min(\lambda,\mu))\|\boldsymbol{X}_1\|_F^2 + (\mu - \min(\lambda,\mu))\|\boldsymbol{Y}_1\|_F^2\right] + \\
&\quad (1-t)\left[\|\boldsymbol{D} - \boldsymbol{X}_2 - \boldsymbol{Y}_2\|_F^2 + (\lambda - \min(\lambda,\mu))\|\boldsymbol{X}_2\|_F^2 + (\mu - \min(\lambda,\mu))\|\boldsymbol{Y}_2\|_F^2\right] \\
&= t \cdot g(\boldsymbol{X}_1,\, \boldsymbol{Y}_1) + (1-t) \cdot g(\boldsymbol{X}_2,\, \boldsymbol{Y}_2).
\end{aligned}
$$

■

Recall that Proposition 2 states that $f(\boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2$ is L-Lipschitz continuous in $(\boldsymbol{X}, \boldsymbol{Y})$. We prove this fact below:

Proof To establish Proposition 2, it suffices to show that $h(\boldsymbol{X}, \boldsymbol{Y}) = \frac{L}{2}(\|\boldsymbol{X}\|_F^2 + \|\boldsymbol{Y}\|_F^2) -$

$f(\boldsymbol{X}, \boldsymbol{Y})$ is convex for $L = 2 \cdot \max(\lambda, \mu) + 6$. We have

$$\begin{aligned}
h(\boldsymbol{X}, \boldsymbol{Y}) &= \frac{L}{2}(\|\boldsymbol{X}\|_F^2 + \|\boldsymbol{Y}\|_F^2) - \lambda\|\boldsymbol{X}\|_F^2 - \mu\|\boldsymbol{Y}\|_F^2 - \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F^2 \\
&= \left(\frac{L}{2} - \lambda - 1\right)\|\boldsymbol{X}\|_F^2 + \left(\frac{L}{2} - \mu - 1\right)\|\boldsymbol{Y}\|_F^2 + 2\Big(\langle \boldsymbol{D}, \boldsymbol{X}\rangle + \langle \boldsymbol{D}, \boldsymbol{Y}\rangle - \langle \boldsymbol{X}, \boldsymbol{Y}\rangle\Big) - \|\boldsymbol{D}\|_F^2 \\
&= \left(\frac{L}{2} - \lambda - 2\right)\|\boldsymbol{X}\|_F^2 + \left(\frac{L}{2} - \mu - 2\right)\|\boldsymbol{Y}\|_F^2 + \|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \\
&\qquad 2\Big(\langle \boldsymbol{D}, \boldsymbol{X}\rangle + \langle \boldsymbol{D}, \boldsymbol{Y}\rangle + \|\boldsymbol{D}\|_F^2\Big) - 3\|\boldsymbol{D}\|_F^2 \\
&= \left(\frac{L}{2} - \lambda - 3\right)\|\boldsymbol{X}\|_F^2 + \left(\frac{L}{2} - \mu - 3\right)\|\boldsymbol{Y}\|_F^2 + \|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \\
&\qquad \|\boldsymbol{X} - \boldsymbol{D}\|_F^2 + \|\boldsymbol{Y} - \boldsymbol{D}\|_F^2 - 3\|\boldsymbol{D}\|_F^2.
\end{aligned}$$

Taking $L = 2 \cdot \max(\lambda, \mu) + 6$, we have $\frac{L}{2} - \lambda - 3 = \max(\lambda, \mu) - \lambda \geq 0$ and $\frac{L}{2} - \mu - 3 = \max(\lambda, \mu) - \mu \geq 0$. Thus, we have written $h(\boldsymbol{X}, \boldsymbol{Y})$ as the sum of convex quadratic functions of $(\boldsymbol{X}, \boldsymbol{Y})$ which immediately implies $h(\boldsymbol{X}, \boldsymbol{Y})$'s joint convexity. ■

Recall that Proposition 3 states if we let $\mathcal{U}_\lambda(\boldsymbol{X}) = \{\boldsymbol{\Delta} \in \mathbb{R}^{n \times n} : \|\boldsymbol{\Delta}\|_F \leq \lambda\|\boldsymbol{X}\|_F\}$ for $\boldsymbol{X} \in \mathbb{R}^{n \times n}, \lambda > 0$, then (2.8) is equivalent to (2.9). We prove this result below:

Proof Consider the inner maximization problem in (2.8) and first note that by applying the triangle inequality for the Frobenius norm, we have

$$\max_{\substack{\boldsymbol{\Delta}_1 \in \mathcal{U}_\lambda(\boldsymbol{X}) \\ \boldsymbol{\Delta}_2 \in \mathcal{U}_\mu(\boldsymbol{Y})}} \|\boldsymbol{D} + \boldsymbol{\Delta_1} + \boldsymbol{\Delta_2} - \boldsymbol{X} - \boldsymbol{Y}\|_F \leq \|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F + \lambda\|\boldsymbol{X}\|_F + \mu\|\boldsymbol{Y}\|_F.$$

Next, note that by taking

$$\boldsymbol{\Delta_1^*} = \frac{\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}}{\|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F} \cdot \lambda\|\boldsymbol{X}\|_F, \text{ and}$$

$$\boldsymbol{\Delta_2^*} = \frac{\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}}{\|\boldsymbol{D} - \boldsymbol{X} - \boldsymbol{Y}\|_F} \cdot \mu\|\boldsymbol{Y}\|_F,$$

the upper bound on the maximization problem is attained:

$$\begin{aligned}\|\boldsymbol{D}+\boldsymbol{\Delta}_1^*+\boldsymbol{\Delta}_2^*-\boldsymbol{X}-\boldsymbol{Y}\|_F &= \left\|(\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y})\cdot\left(1+\frac{\lambda\|\boldsymbol{X}\|_F+\mu\|\boldsymbol{Y}\|_F}{\|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F}\right)\right\|_F \\ &= \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F+\lambda\|\boldsymbol{X}\|_F+\mu\|\boldsymbol{Y}\|_F.\end{aligned}$$

The proof is concluded by noting that we have $\boldsymbol{\Delta}_1^* \in \mathcal{U}_\lambda(\boldsymbol{X})$ and $\boldsymbol{\Delta}_2^* \in \mathcal{U}_\mu(\boldsymbol{Y})$. ■

We now provide a formal proof of Proposition 4:

Proof Let us rewrite Problem (2.1) as

$$\begin{aligned}\min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{U},\boldsymbol{V}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2+\lambda\|\boldsymbol{U}\|_F^2+\mu\|\boldsymbol{V}\|_F^2 \\ \text{s.t.} \quad & \text{Rank}(\boldsymbol{X})\leq k_0,\ \|\boldsymbol{Y}\|_0\leq k_1, \boldsymbol{X}=\boldsymbol{U}, \boldsymbol{Y}=\boldsymbol{V},\end{aligned}$$

and associate matrices of dual multipliers $\boldsymbol{\alpha}, \boldsymbol{\beta}$ with the linear constraints $\boldsymbol{X} = \boldsymbol{U}$ and $\boldsymbol{Y} = \boldsymbol{V}$ respectively. Then, this problem can be rewritten as

$$\begin{aligned}\min_{\boldsymbol{X},\boldsymbol{Y}}\min_{\boldsymbol{U},\boldsymbol{V}}\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \|\boldsymbol{D}-\boldsymbol{X}-\boldsymbol{Y}\|_F^2+\lambda\|\boldsymbol{U}\|_F^2+\mu\|\boldsymbol{V}\|_F^2+\langle\boldsymbol{\alpha},\boldsymbol{X}-\boldsymbol{U}\rangle+\langle\boldsymbol{\beta},\boldsymbol{Y}-\boldsymbol{V}\rangle \\ \text{s.t.} \quad & \text{Rank}(\boldsymbol{X})\leq k_0,\ \|\boldsymbol{Y}\|_0\leq k_1.\end{aligned}$$

Therefore, let us fix $\boldsymbol{X}, \boldsymbol{Y}$ and use a standard minimax theorem [see, e.g., 12, Chap. 6] to exchange the order of minimizing $\boldsymbol{U}, \boldsymbol{V}$ and maximizing $\boldsymbol{\alpha}, \boldsymbol{\beta}$. This gives the following subproblem in $\boldsymbol{U}, \boldsymbol{V}$ for a fixed $\boldsymbol{\alpha}, \boldsymbol{\beta}$:

$$\min_{\boldsymbol{U},\boldsymbol{V}} \quad \lambda\|\boldsymbol{U}\|_F^2+\mu\|\boldsymbol{V}\|_F^2+\langle\boldsymbol{\alpha},-\boldsymbol{U}\rangle+\langle\boldsymbol{\beta},-\boldsymbol{V}\rangle.$$

By differentiating and setting the gradient to zero, it is not too hard to see that this subproblem takes the value $\frac{-1}{4\lambda}\|\boldsymbol{\alpha}\|_F^2-\frac{1}{4\mu}\|\boldsymbol{\beta}\|_F^2$. This implies the result. ■

Recall that Proposition 5 establishes that (2.1) reduces to regularized matrix completion with $\Omega = \{(i,j) : Z_{ij} = 0\}$ where $\boldsymbol{Z}$ denotes a valid sparsity pattern and we take $\mu = 0$. We prove this result below:

Proof Given a valid sparsity pattern $\boldsymbol{Z}$ and letting $\Omega = \{(i,j) : Z_{ij} = 0\}$, Problem (2.1) can be expressed as

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Y} \in \mathbb{R}^{n \times n}} \quad & \lambda \|\boldsymbol{X}\|_F^2 + \sum_{(i,j) \in \Omega} (D_{ij} - X_{ij} - Y_{ij})^2 + \mu Y_{ij}^2 + \sum_{(i,j) \notin \Omega} (D_{ij} - X_{ij} - Y_{ij})^2 + \mu Y_{ij}^2 \\ \text{s.t.} \quad & \text{Rank}(\boldsymbol{X}) \leq k_0, \; Y_{ij} = 0 \; \forall \; (i,j) \in \Omega. \end{aligned}$$

Simple unconstrained minimization gives $Y_{ij} = \frac{D_{ij} - X_{ij}}{1+\mu}$ for $(i,j) \notin \Omega$. Using this relationship, Problem (2.1) can be further simplified to

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{R}^{n \times n}} \quad & \lambda \cdot \|\boldsymbol{X}\|_F^2 + \sum_{(i,j) \in \Omega} (D_{ij} - X_{ij})^2 + \frac{\mu}{1+\mu} \cdot \sum_{(i,j) \notin \Omega} (D_{ij} - X_{ij})^2. \\ \text{s.t.} \quad & \text{Rank}(\boldsymbol{X}) \leq k_0. \end{aligned} \tag{2.36}$$

The result then follows by observing that the last term in the objective function of (2.36) disappears when $\mu = 0$. Moreover, if we take $\lambda = 0$, then (2.36) exactly becomes (2.11). ■

We now provide a formal proof of Proposition 11:

Proof First, note that given the full sparsity pattern, the iterates $(\boldsymbol{X}_t^{AM}, \boldsymbol{Y}_t^{AM})$ produced by Algorithm 1 satisfy $\boldsymbol{Y}_{t+1}^{AM} = \boldsymbol{S}^* \circ \left(\frac{\boldsymbol{D} - \boldsymbol{X}_t^{AM}}{1+\mu} \right)$ and $\boldsymbol{X}_{t+1}^{AM} = \frac{1}{1+\lambda} \mathcal{P}_\Omega(\boldsymbol{D} - \boldsymbol{Y}_{t+1}^{AM})$. This implies that

$$\boldsymbol{X}_{t+1}^{AM} = \mathcal{P}_\Omega \left(\frac{1}{1+\lambda} \left[\boldsymbol{D} - \boldsymbol{S}^* \circ \left(\frac{\boldsymbol{D} - \boldsymbol{X}_t^{AM}}{1+\mu} \right) \right] \right). \tag{2.37}$$

Next, note that the gradient of $g(\boldsymbol{X}_t)$ is given by

$$\nabla g(\boldsymbol{X}_t) = 2 \left((1+\lambda) \boldsymbol{X}_t - \boldsymbol{D} + \boldsymbol{S}^* \circ \left(\frac{\boldsymbol{D} - \boldsymbol{X}_t}{1+\mu} \right) \right).$$

The result follows by noting that the Projected Gradient Descent update $\boldsymbol{X}_{t+1} = \mathcal{P}_{\Omega}(\boldsymbol{X}_t - \eta\nabla g(\boldsymbol{X}_t))$ is the same as the update given by (2.37) when $\eta = \frac{1}{2(1+\lambda)}$. ■

We now provide a formal proof of Proposition 14:

Proof We show that given a feasible solution to (2.18), we can construct a feasible solution to (2.1) that achieves the same objective value and vice versa.

Consider an arbitrary feasible solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{P}})$ to (2.18). Since $\bar{\boldsymbol{Z}} \in \mathcal{Z}_{k_1}$ and $\bar{\boldsymbol{Y}} = \bar{\boldsymbol{Z}} \circ \bar{\boldsymbol{Y}}$, we have $\|\bar{\boldsymbol{Y}}\|_0 \leq k_1$. Moreover, since $\bar{\boldsymbol{P}} \in \mathcal{P}_{k_0}$ and $\bar{\boldsymbol{X}} = \bar{\boldsymbol{P}}\bar{\boldsymbol{X}}$, we have $\text{Rank}(\bar{\boldsymbol{X}}) \leq k_0$. Thus, $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ is feasible to (2.1). Since both (2.18) and (2.1) have the same objective function, $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ achieves the same objective in (2.1) as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{P}})$ does in (2.18).

Consider an arbitrary feasible solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ to (2.1). Let $\bar{\boldsymbol{Z}} \in \{0,1\}^{n\times n}$ be the binary matrix such that $\bar{Z}_{ij} = 1$ if $\bar{Y}_{ij} \neq 0$ and $\bar{Z}_{ij} = 0$ otherwise. Further, let $\bar{\boldsymbol{P}} = \boldsymbol{U}\boldsymbol{U}^T$ where $\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a singular value decomposition of $\bar{\boldsymbol{X}}$. By construction, we have $\bar{\boldsymbol{Z}} \in \mathcal{Z}_{k_1}$ and $\bar{\boldsymbol{P}} \in \mathcal{P}_{k_0}$ since $\|\bar{\boldsymbol{Y}}\|_0 \leq k_1$ and $\text{Rank}(\bar{\boldsymbol{X}}) \leq k_0$. Thus, $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{P}})$ is feasible to (2.18) and achieves the same objective as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ does in (2.1). This completes the proof. ■

We now provide a formal proof of Theorem 15:

Proof Clearly Problem (2.20) is a convex optimization problem. We will show that given any feasible solution to Problem (2.1), we can construct a feasible solution to (2.20) that achieves the same objective value.

Consider an arbitrary feasible solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ to (2.1). Let $\bar{\boldsymbol{Z}} \in \{0,1\}^{n\times n}$ be the binary matrix such that $\bar{Z}_{ij} = 1$ if $\bar{Y}_{ij} \neq 0$ and $\bar{Z}_{ij} = 0$ otherwise and let $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^{n\times n}$ be the matrix such that $\bar{\alpha}_{ij} = \bar{Y}_{ij}^2$. Further, let $\bar{\boldsymbol{P}} = \boldsymbol{U}\boldsymbol{U}^T$ where $\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a singular value decomposition of $\bar{\boldsymbol{X}}$ and let $\bar{\boldsymbol{\Theta}} = \bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}$. By construction, we have $\bar{\boldsymbol{Z}} \in \mathcal{Z}_{k_1}$ and $\bar{\boldsymbol{P}} \in \mathcal{P}_{k_0}$ since $\|\bar{\boldsymbol{Y}}\|_0 \leq k_1$ and $\text{Rank}(\bar{\boldsymbol{X}}) \leq k_0$ which implies that $\text{tr}(\boldsymbol{E}\bar{\boldsymbol{Z}}) \leq k_1, 0 \leq \bar{\boldsymbol{Z}} \leq 1, \bar{\boldsymbol{P}} \succeq 0, \mathbb{I} - \bar{\boldsymbol{P}} \succeq 0$ and $\text{tr}(\bar{\boldsymbol{P}}) \leq k_0$. It is straightforward to see that we have $\bar{Y}_{ij}^2 \leq \bar{\alpha}_{ij}\bar{Z}_{ij}\ \forall\,(i,j)$. Finally, we have

$\bar{\boldsymbol{\Theta}} = \bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}^T\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}^T\bar{\boldsymbol{P}}^\dagger\bar{\boldsymbol{X}}$ so we have $\begin{pmatrix} \bar{\boldsymbol{\Theta}} & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \bar{\boldsymbol{P}} \end{pmatrix} \succeq 0$. Thus, we have shown that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{\Theta}}, \bar{\boldsymbol{\alpha}})$ is feasible to (2.20). This achieves an objective of

$$\begin{aligned}\|\boldsymbol{D} - \bar{\boldsymbol{X}} - \bar{\boldsymbol{Y}}\|_F^2 + \lambda\mathrm{tr}(\bar{\boldsymbol{\Theta}}) + \mu\mathrm{tr}(\boldsymbol{E}\bar{\boldsymbol{\alpha}}) &= \|\boldsymbol{D} - \bar{\boldsymbol{X}} - \bar{\boldsymbol{Y}}\|_F^2 + \lambda\mathrm{tr}(\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}) + \mu\sum_{ij}\bar{Y}_{ij}^2 \\ &= \|\boldsymbol{D} - \bar{\boldsymbol{X}} - \bar{\boldsymbol{Y}}\|_F^2 + \lambda\|\bar{\boldsymbol{X}}\|_F^2 + \mu\|\bar{\boldsymbol{Y}}\|_F^2.\end{aligned}$$

which is the same objective achieved by $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}})$ in (2.1). This completes the proof. ■

2.9 Appendix: Alternative Proof of Proposition 6

Proof Clearly, $\boldsymbol{X}^*$ is feasible for (2.12). Let $\boldsymbol{P}^* = \boldsymbol{U}_{k_0}\boldsymbol{U}_{k_0}^T$ and $\boldsymbol{\Theta}^* = \boldsymbol{X}^{*T}\boldsymbol{X}^*$. As established in the proof of Theorem 16, $(\boldsymbol{X}^*, \boldsymbol{P}^*, \boldsymbol{\Theta}^*)$ is feasible to (2.21) and achieves the same objective as $\boldsymbol{X}^*$ does in (2.12). We prove Proposition 6 by deriving the dual of (2.21) and constructing a dual feasible solution that achieves the same objective value as $(\boldsymbol{X}^*, \boldsymbol{P}^*, \boldsymbol{\Theta}^*)$ achieves in (2.21). By duality, this then implies that $(\boldsymbol{X}^*, \boldsymbol{P}^*, \boldsymbol{\Theta}^*)$ is optimal for (2.21) which in turn implies that $\boldsymbol{X}^*$ is optimal for (2.12).

The dual of (2.21) is given by

$$\begin{aligned}\max_{\boldsymbol{A},\boldsymbol{B}\in\mathcal{S}_+^n,\sigma\geq 0} \quad & \|\bar{\boldsymbol{D}}\|_F^2 + \sigma(n-k_0) - \mathrm{tr}(\boldsymbol{B}) \\ \text{s.t.} \quad & (1+\lambda)\mathbb{I} \succeq \boldsymbol{A},\ \boldsymbol{B} \succeq \sigma\mathbb{I},\ \begin{pmatrix} \boldsymbol{A} & \bar{\boldsymbol{D}} \\ \bar{\boldsymbol{D}}^T & \boldsymbol{B} \end{pmatrix} \succeq 0.\end{aligned} \tag{2.38}$$

Let $\{\phi_i\}_{i=1}^n$ denote the collection of singular values of $\bar{\boldsymbol{D}}$ in non-increasing order (so that $\phi_i \geq \phi_{i+1}\ \forall\, i$). Let $\sigma^* = \frac{1}{1+\lambda}\phi_{k_0}^2$. Let $\nu_i^* = \frac{1}{1+\lambda}\phi_i^2\ \forall\, i < k_0$ and let $\nu_i^* = \sigma^*\ \forall\, k_0 \leq i \leq n$. Let $\boldsymbol{A}^* = (1+\lambda)\mathbb{I}$ and $\boldsymbol{B}^* = \boldsymbol{U}Diag(\boldsymbol{\nu})\boldsymbol{U}^T$ where $\bar{\boldsymbol{D}} = \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^T$ is a spectral decomposition of

$\bar{\boldsymbol{D}}$ and $Diag(\boldsymbol{\nu})$ denotes the $n \times n$ diagonal matrix with diagonal entries given by the entries of $\boldsymbol{\nu}$. Note that the solution $(\boldsymbol{A}^*, \boldsymbol{B}^*, \sigma^*)$ is feasible to (2.38). To see this, observe that by construction, we have $\boldsymbol{A}^*, \boldsymbol{B}^* \in \mathcal{S}_+^n, \sigma^* \geq 0$, and $(1+\lambda)\mathbb{I} \succeq \boldsymbol{A}^*$. Moreover, since $\{\phi_i\}_{i=1}^n$ are in non-increasing order, we have $\min_i \nu_i \geq \sigma^*$ which implies $\boldsymbol{B}^* \succeq \sigma\mathbb{I}$. Finally, we have $\nu_i \geq \frac{1}{1+\lambda}\phi_i^2 \ \forall\, i$ which implies that $\boldsymbol{B}^* \succeq \bar{\boldsymbol{D}}^T \boldsymbol{A}^{*-1} \bar{\boldsymbol{D}}$ and $\begin{pmatrix} \boldsymbol{A}^* & \bar{\boldsymbol{D}} \\ \bar{\boldsymbol{D}}^T & \boldsymbol{B}^* \end{pmatrix} \succeq 0$. The feasible solution $(\boldsymbol{A}^*, \boldsymbol{B}^*, \sigma^*)$ achieves an objective of:

$$\begin{aligned} \|\bar{\boldsymbol{D}}\|_F^2 + \sigma^*(n-k_0) - \mathrm{tr}(\boldsymbol{B}^*) &= \sum_{i=1}^{n} \phi_i^2 + \frac{n-k_0}{1+\lambda}\phi_{k_0}^2 - \frac{1}{1+\lambda}\sum_{i=1}^{k_0-1}\phi_i^2 - \frac{1}{1+\lambda}\sum_{i=k_0}^{n}\phi_{k_0}^2 \\ &= \frac{\lambda}{1+\lambda}\sum_{i=1}^{k_0}\phi_i^2 + \sum_{i=k_0+1}^{n}\phi_i^2 \end{aligned}$$

in (2.38). Moreover, the solution $(\boldsymbol{X}^*, \boldsymbol{P}^*, \boldsymbol{\Theta}^*)$ achieves the same objective in (2.21):

$$\begin{aligned} \|\bar{\boldsymbol{D}}\|_F^2 + (1+\lambda)\mathrm{tr}(\bar{\boldsymbol{\Theta}}^*) - 2\cdot\mathrm{tr}(\bar{\boldsymbol{X}}^*\bar{\boldsymbol{D}}) &= \sum_{i=1}^{n}\phi_i^2 + \frac{1}{1+\lambda}\sum_{i+1}^{k_0}\phi_i^2 - \frac{2}{1+\lambda}\sum_{i=1}^{k_0}\phi_i^2 \\ &= \frac{\lambda}{1+\lambda}\sum_{i=1}^{k_0}\phi_i^2 + \sum_{i=k_0+1}^{n}\phi_i^2. \end{aligned}$$

By duality, the objective value of any feasible solution to (2.38) provides a lower bound on the objective of (2.21). Since $(\boldsymbol{X}^*, \boldsymbol{P}^*, \boldsymbol{\Theta}^*)$ is primal feasible and achieves the same objective as a feasible dual solution, it must be optimal for (2.21). This in turn implies that $\boldsymbol{X}^*$ is optimal to (2.12) by Theorem 16. This completes the proof. ■

2.10 Appendix: Proof of Convexity in the Low-Rank Subproblem

Proof We prove the equivalence in two steps. First, we show that given a feasible solution to (2.12), we can construct a feasible solution to (2.21) that achieves the same objective value. Second, we show that given a feasible solution to (2.21), we can construct a feasible solution to (2.12) that achieves the same or lower objective. Given an arbitrary feasible solution to (2.21), we construct a linear optimization problem in which feasible solutions correspond to feasible solutions to (2.21) and extreme points of the feasible set of the linear optimization problem correspond to feasible solutions to (2.12). The initial feasible solution to (2.21) is feasible to this linear optimization problem, so there is an extreme point corresponding to a feasible solution to (2.12) that achieves an equal or lower objective value.

Consider an arbitrary feasible solution $\bar{\boldsymbol{X}}$ to Problem (2.12). Since $\boldsymbol{D}$ is symmetric, we can restrict ourselves to considering symmetric feasible solutions. Since we have $\text{Rank}(\bar{\boldsymbol{X}}) \leq k$ and $\bar{\boldsymbol{X}}$ is symmetric, we can factor $\bar{\boldsymbol{X}}$ as $\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ where $\boldsymbol{U} \in \mathbb{R}^{n \times k_0}$, $\boldsymbol{U}^T\boldsymbol{U} = \mathbb{I}_{k_0}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{k_0 \times k_0}$ and $\boldsymbol{\Sigma}$ is diagonal. Let $\bar{\boldsymbol{P}} = \boldsymbol{U}\boldsymbol{U}^T$. $\bar{\boldsymbol{P}}$ is the orthogonal projection matrix onto the k_0 dimensional column space of $\boldsymbol{U}$. This implies that $\bar{\boldsymbol{P}} \succeq 0$, $\mathbb{I} - \bar{\boldsymbol{P}} \succeq 0$ and $\text{tr}(\bar{\boldsymbol{P}}) \leq k_0$. Let $\bar{\boldsymbol{\Theta}} = \bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}} \succeq 0$. Note that $\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{P}} = \bar{\boldsymbol{P}}^\dagger$, where $\bar{\boldsymbol{P}}^\dagger$ denotes the pseudo-inverse of $\bar{\boldsymbol{P}}$, since $\bar{\boldsymbol{P}}$ is an orthogonal projection matrix. Thus, we have $\bar{\boldsymbol{\Theta}} - \bar{\boldsymbol{X}}^T\bar{\boldsymbol{P}}^\dagger\bar{\boldsymbol{X}} = 0 \implies \begin{pmatrix} \bar{\boldsymbol{\Theta}} & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \bar{\boldsymbol{P}} \end{pmatrix} \succeq 0$. We have shown that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{\Theta}})$ is feasible to (2.21). To see that this solution achieves the same objective as $\bar{\boldsymbol{X}}$ achieves in (2.12), note that

$$\begin{aligned} \|\bar{\boldsymbol{D}} - \bar{\boldsymbol{X}}\|_F^2 + \lambda\|\bar{X}\|_F^2 &= \|\bar{\boldsymbol{D}}\|_F^2 + (1+\lambda)\|\bar{\boldsymbol{X}}\|_F^2 - 2 \cdot \text{tr}(\bar{\boldsymbol{X}}\bar{\boldsymbol{D}}) \\ &= \|\bar{\boldsymbol{D}}\|_F^2 + (1+\lambda)\text{tr}(\bar{\boldsymbol{\Theta}}) - 2 \cdot \text{tr}(\bar{\boldsymbol{X}}\bar{\boldsymbol{D}}). \end{aligned}$$

Now, consider an arbitrary feasible solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{\Theta}})$ to (2.21). Since the objective function of (2.21) includes the term $\text{tr}(\boldsymbol{\Theta})$ and feasibility requires $\boldsymbol{\Theta} \succeq \boldsymbol{X}^T\boldsymbol{P}^\dagger\boldsymbol{X}$, we can

take $\boldsymbol{\Theta}' = \bar{\boldsymbol{X}}^T \bar{\boldsymbol{P}}^\dagger \bar{\boldsymbol{X}}$ and the solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \boldsymbol{\Theta}')$ will be feasible to (2.21) with an objective value no greater than that of the original feasible solution. Since $\bar{\boldsymbol{P}}$ is PSD, it can be written as $\bar{\boldsymbol{P}} = \sum_{i=1}^n \phi_i u_i u_i^T$ where $u_i^T u_i = 1$ for all i, $u_i^T u_j = 0$ for all $i \neq j$ and the feasibility of $\bar{\boldsymbol{P}}$ implies $0 \leq \phi_i \leq 1$ for all i. Moreover, we have $\bar{\boldsymbol{P}}^\dagger = \sum_{i:\phi_i \neq 0} \frac{1}{\phi_i} u_i u_i^T$. Further, since the feasibility condition $\begin{pmatrix} \boldsymbol{\Theta}' & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \bar{\boldsymbol{P}} \end{pmatrix} \succeq 0$ implies that $\bar{\boldsymbol{X}} = \bar{\boldsymbol{P}}^\dagger \bar{\boldsymbol{P}} \bar{\boldsymbol{X}}$ by the generalized Schur complement lemma (see Boyd et al. 1994, Equation 2.41) and $\bar{\boldsymbol{X}}$ is symmetric, without loss of generality it can be written as $\bar{\boldsymbol{X}} = \sum_{i=1}^n \sigma_i u_i u_i^T$. The solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \boldsymbol{\Theta}')$ achieves an objective of

$$\begin{aligned} h(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \boldsymbol{\Theta}') &= \|\bar{\boldsymbol{D}}\|_F^2 + (1+\lambda)\mathrm{tr}(\boldsymbol{\Theta}') - 2 \cdot \mathrm{tr}(\bar{\boldsymbol{X}} \bar{\boldsymbol{D}}) \\ &= \|\bar{\boldsymbol{D}}\|_F^2 + \sum_{i:\phi_i \neq 0} \left[\frac{1+\lambda}{\phi_i} \sigma_i^2 - 2 \cdot \sigma_i \mathrm{tr}(u_i u_i^T \bar{\boldsymbol{D}}) \right]. \end{aligned}$$

Note that if we view the above as a function of σ_i and ϕ_i (denoted $f(\phi, \sigma)$), then this expression corresponds to the objective value achieved by some feasible solution to (2.21) provided we constrain $0 \leq \phi_i \leq 1$ and $\sum_i \phi_i \leq k_0$. $h(\phi, \sigma)$ is a convex quadratic in σ_i. It is minimized when $\nabla_{\sigma_i} h(\phi, \sigma) = \frac{2(1+\lambda)}{\phi_i} \sigma_i - 2\mathrm{tr}(u_i u_i^T \bar{\boldsymbol{D}}) = 0 \implies \sigma_i = \frac{\phi_i}{1+\lambda} \mathrm{tr}(u_i u_i^T \bar{\boldsymbol{D}})$. Substituting the optimal value of σ_i into $h(\phi, \sigma)$, we obtain

$$h(\phi) = \min_\sigma f(\phi, \sigma) = \|\bar{\boldsymbol{D}}\|_F^2 - \sum_{i:\phi_i \neq 0} \frac{\phi_i}{1+\lambda} [\mathrm{tr}(u_i u_i^T \bar{\boldsymbol{D}})]^2 = \|\bar{\boldsymbol{D}}\|_F^2 - \sum_{i=1}^n \frac{\phi_i}{1+\lambda} [\mathrm{tr}(u_i u_i^T \boldsymbol{D}^*)]^2.$$

$h(\phi)$ is a linear function of ϕ. Therefore, the minimum of $h(\phi)$ over the set $0 \leq \phi_i \leq 1$ for all i, $\sum_i \phi_i \leq k_0$ is achieved at some $\phi^* \in \{0,1\}^{n \times n}$. Let $\boldsymbol{P}^* = \sum_{i=1}^n \phi_i^* u_i u_i^T$, $\boldsymbol{X}^* = \sum_{i=1}^n \phi_i^* \mathrm{tr}(u_i u_i^T \bar{\boldsymbol{D}}) u_i u_i^T$ and $\boldsymbol{\Theta}^* = \boldsymbol{X}^{*T} \boldsymbol{P}^* \boldsymbol{X}$. Then $(\boldsymbol{X}^*, \boldsymbol{P}^*, \boldsymbol{\Theta}^*)$ is feasible to (2.21) and achieves objective $h(\phi^*)$. By construction, we have

$$h(\phi^*) \leq h(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \boldsymbol{\Theta}') \leq h(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{\Theta}}).$$

Further, since $\phi^* \in \{0,1\}^{n\times n}$ and $\sum_i \phi_i^* \leq k_0$, we have $\text{Rank}(\boldsymbol{X}^*) \leq k_0$ which means that $\boldsymbol{X}^*$ is feasible to (2.12) and achieves objective $h(\phi^*)$. This completes the proof. ■

2.11 Appendix: Alternative Proof of Proposition 8

Proof Let $f(\boldsymbol{Y}) = \|\tilde{\boldsymbol{D}} - \boldsymbol{Y}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2$, the objective function of Problem (2.14). We can rewrite $f(\boldsymbol{Y})$ as:

$$f(\boldsymbol{Y}) = \|\tilde{\boldsymbol{D}} - \boldsymbol{Y}\|_F^2 + \mu\|\boldsymbol{Y}\|_F^2 = \sum_{ij}(\tilde{d}_{ij} - y_{ij})^2 + \mu\sum_{ij} y_{ij}^2$$
$$= \sum_{ij}\left[(\tilde{d}_{ij} - y_{ij})^2 + y_{ij}^2\right] = \sum_{ij} f_{ij}(y),$$

where we define $f_{ij}(y) = (\tilde{d}_{ij} - y)^2 + y^2$. We have shown that the objective function is separable, so Problem (2.14) can be solved by minimizing each function $f_{ij}(y)$. $f_{ij}(y)$ is a convex quadratic function, and simple univariate calculus allows us to conclude that it achieves its minimum when $y^* = \frac{\tilde{d}_{ij}}{1+\mu}$. The minimum value of f_{ij} is therefore $f_{ij}(y^*) = \frac{\mu}{1+\mu}\tilde{d}_{ij}^2$. However, due to the sparsity constraint on $\boldsymbol{Y}$, at most k_1 entries of $\boldsymbol{Y}$ can be non-zero. By introducing binary variables s_{ij} and noting that $f_{ij}(0) = \tilde{d}_{ij}^2$, we can rewrite the objective of problem 2 as a function of the binary matrix $\boldsymbol{S}$:

$$f(\boldsymbol{S}) = \sum_{ij}\left[s_{ij}\cdot\frac{\mu}{1+\mu}\tilde{d}_{ij}^2 + (1 - s_{ij})\cdot\tilde{d}_{ij}^2\right].$$

Due to the sparsity constraint, at most k_1 of the variables s_{ij} can be 1 while all others must be 0. If $s_{ij} = 0$, the objective increases by $\tilde{d}_{ij}^2$ whereas if $s_{ij} = 1$, the objective only increases by $\frac{\mu}{1+\mu}\tilde{d}_{ij}^2$. It follows immediately that the objective will be minimized when $s_{ij} = 1$ if and only if $\tilde{d}_{ij}$ is one of the k_1 largest entries in absolute value of the matrix $\tilde{\boldsymbol{D}}$. Note that in the case that the k_1^{th} largest entry in absolute value and the $(k_1+1)^{th}$ largest entry in absolute

value are not distinct, the tie can be broken arbitrarily. Letting $\boldsymbol{S}^*$ represent the binary matrix formed by an optimal choice of the binary variables s_{ij}, the solution to Problem (2.14) is given by $\boldsymbol{Y}^* = \boldsymbol{S}^* \circ \left(\frac{\tilde{\boldsymbol{D}}}{1+\mu}\right)$. ■

2.12 Appendix: Supplemental Computational Results

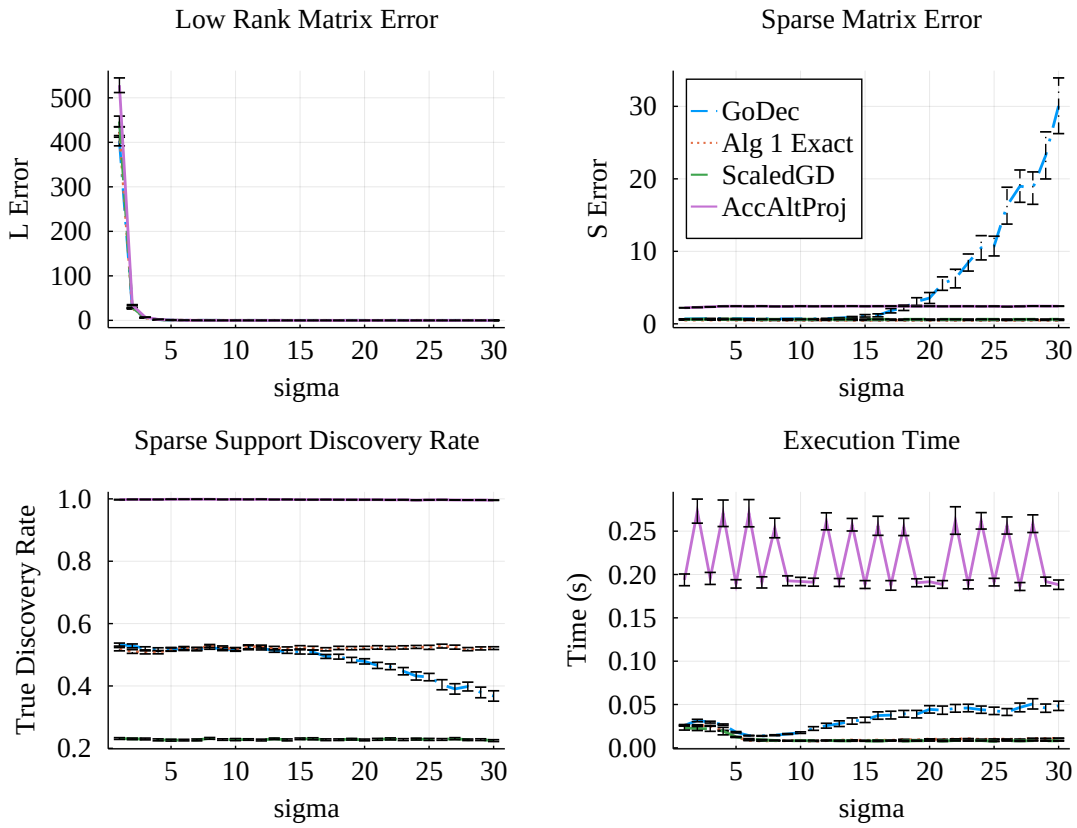

Figure 2.8: Low-rank matrix reconstruction error (top left), sparse matrix reconstruction error (top right), sparse support discovery rate (bottom left) and execution time (bottom right) versus σ with $n = 100$, $k_0 = 5$ and $k_1 = 500$. Averaged over 50 trials for each parameter configuration.

Table 2.2: Comparison of average low-rank matrix reconstruction error generated by S-PCP, GoDec, ScaledGD, AccAltProj, fRPCA, and Algorithm 1. Results are reported for the exact SVD implementation of GoDec. Averaged over 10 trials for each parameter configuration.

					L Error					
N	k_0	k_1	S-PCP Rank	S-PCP Sparsity	S-PCP	GoDec	ScaledGD	AccAltProj	fRPCA	Alg 1 Exact
20	1	20	5.7	95.6	0.0176	0.0101	0.0082	0.0111	0.0088	**0.0072**
20	2	40	12.0	197.4	0.0178	0.0430	0.0062	0.0074	0.0068	**0.0057**
20	3	60	15.3	275.3	0.1123	0.1136	0.0084	0.0083	0.0077	**0.0075**
20	4	80	17.5	341.1	0.1510	0.3247	0.0087	0.0092	0.0088	**0.0079**
40	2	80	5.4	286.6	0.0233	0.0121	0.0147	0.0168	0.0174	**0.0110**
40	4	160	16.4	417.2	0.0272	0.0189	0.0122	0.0143	0.0136	**0.0113**
40	6	240	27.3	731.3	0.0334	0.0996	0.0159	0.0171	0.0165	**0.0145**
40	8	320	36.7	1365.1	0.0453	0.3225	0.0170	0.0178	0.0157	**0.0149**
60	3	180	7.8	631.6	0.0311	0.0158	0.0182	0.0231	0.0197	**0.0149**
60	6	360	13.0	777.6	0.0328	0.0247	0.0171	0.0222	0.0177	**0.0150**
60	9	540	36.3	1181.1	0.0439	0.0520	0.0236	0.0251	0.0226	**0.0202**
60	12	720	55.9	2930.5	0.0577	0.2696	0.0236	0.0316	0.0242	**0.0209**
80	4	320	10.9	1128.5	0.0345	0.0176	0.0230	0.0272	0.0238	**0.0166**
80	8	640	15.4	1380.1	0.0448	0.0293	0.0240	0.0314	0.0248	**0.0223**
80	12	960	34.0	1634.6	0.0569	0.0537	0.0271	0.0307	0.0269	**0.0246**
80	16	1280	62.7	3316.8	0.0737	0.2989	0.0339	0.0378	0.0339	**0.0300**
100	5	500	13.8	1771.6	0.0443	0.0255	0.0288	0.0383	0.0267	**0.0239**
100	10	1000	19.2	2139.9	0.0531	0.0357	0.0318	0.0385	0.0345	**0.0271**
100	15	1500	36.4	2525.9	0.0640	0.0679	0.0356	0.0392	0.0330	**0.0304**
100	20	2000	63.4	3145.1	0.0840	0.3675	0.0399	0.0471	0.0395	**0.0381**
120	12	1440	21.3	3067.7	0.0644	0.0423	0.0368	0.0474	0.0400	**0.0333**
120	18	2160	38.8	3628.4	0.0789	0.0858	0.0440	0.0497	0.0424	**0.0388**
120	24	2880	72.0	4288.3	0.0968	0.3838	0.0512	0.0570	0.0498	**0.0464**
140	7	980	19.3	3436.0	0.0613	0.0365	0.0386	0.0553	0.0375	**0.0331**
140	21	2940	37.9	4911.8	0.0868	0.0910	0.0506	0.0573	0.0479	**0.0442**
140	28	3920	76.7	5790.9	0.1085	0.4156	0.0607	0.0695	0.0598	**0.0566**

Table 2.3: Bound gap of Algorithm 1 derived using (2.20). Averaged over 10 trials for each parameter configuration.

			L Error							
N	k_0	k_1	S-PCP	GoDec	ScaledGD	AccAltProj	fRPCA	Alg 1 Exact	Alg 1 Bound Gap	Bound Time (s)
20	1	20	.0176	.0101	0.0082	0.0111	0.0088	**0.0072**	0.7052	3.7200
60	6	360	.0328	.0247	0.0171	0.0222	0.0177	**0.0150**	0.8543	189.1900
60	9	540	.0439	.052	0.0236	0.0251	0.0226	**0.0202**	0.8601	184.9500
60	12	720	.0577	.2696	0.0236	0.0316	0.0242	**0.0209**	0.7709	155.2800
80	4	320	.0345	.0176	0.0230	0.0272	0.0238	**0.0166**	0.9180	577.8400
80	8	640	.0448	.0293	0.0240	0.0314	0.0248	**0.0223**	0.9267	765.9100
80	12	960	.0569	.0537	0.0271	0.0307	0.0269	**0.0246**	0.7944	691.5500
80	16	1280	.0737	.2989	0.0339	0.0378	0.0339	**0.0300**	0.7803	611.4700
100	5	500	.0443	.0255	0.0288	0.0383	0.0267	**0.0239**	0.9592	1936.2600
100	10	1000	.0531	.0357	0.0318	0.0385	0.0345	**0.0271**	0.9382	2987.0800
100	15	1500	.064	.0679	0.0356	0.0392	0.0330	**0.0304**	0.9062	2224.6100
20	2	40	.0178	.043	0.0062	0.0074	0.0068	**0.0057**	0.5935	3.8200
100	20	2000	.084	.3675	0.0399	0.0471	0.0395	**0.0381**	0.8145	2188.6600
120	12	1440	.0644	.0423	0.0368	0.0474	0.0400	**0.0333**	0.8951	6759.9200
120	18	2160	.0789	.0858	0.0440	0.0497	0.0424	**0.0388**	0.8968	6878.3600
120	24	2880	.0968	.3838	0.0512	0.0570	0.0498	**0.0464**	0.7877	5310.5800
140	7	980	.0613	.0365	0.0386	0.0553	0.0375	**0.0331**	0.9014	14731.2500
140	21	2940	.0868	.091	0.0506	0.0573	0.0479	**0.0442**	0.8854	11260.5200
140	28	3920	.1085	.4156	0.0607	0.0695	0.0598	**0.0566**	0.8116	11840.3000
20	3	60	.1123	.1136	0.0084	0.0083	0.0077	**0.0075**	0.5443	3.9600
20	4	80	.151	.3247	0.0087	0.0092	0.0088	**0.0079**	0.7146	4.0500
40	2	80	.0233	.0121	0.0147	0.0168	0.0174	**0.0110**	0.8214	30.6200
40	4	160	.0272	.0189	0.0122	0.0143	0.0136	**0.0113**	0.8804	27.9200
40	6	240	.0334	.0996	0.0159	0.0171	0.0165	**0.0145**	0.7937	28.4700
40	8	320	.0453	.3225	0.0170	0.0178	0.0157	**0.0149**	0.7051	23.8700
60	3	180	.0311	.0158	0.0182	0.0231	0.0197	**0.0149**	0.8075	154.9000

Table 2.4: Running time of the exact implementation of Algorithm 1 and the accelerated implementation of Algorithm 1. In the exact implementation, the SVD step is computed exactly, whereas in the accelerated implementation, a randomized SVD is employed in all but the final SVD step. Averaged over 10 trials for each parameter configuration.

			L Error		Time (s)		
N	k_0	k_1	Alg 1 Exact	Alg 1 Acc	Alg 1 Exact	Alg 1 Acc	Time Decrease (%)
20	1	20	**0.0072**	0.0094	0.1351	**0.0986**	27.06
20	2	40	**0.0057**	0.0084	0.2342	**0.1071**	54.27
20	3	60	**0.0075**	0.0084	0.5713	**0.1394**	75.59
20	4	80	**0.0079**	0.0085	0.8126	**0.1519**	81.31
40	2	80	**0.0110**	0.0123	0.4157	**0.1982**	52.31
40	4	160	**0.0113**	0.0139	0.9250	**0.2536**	72.59
40	6	240	**0.0145**	0.0183	2.0046	**0.3574**	82.17
40	8	320	**0.0149**	0.0192	2.8281	**0.4309**	84.76
60	3	180	**0.0149**	0.0178	0.7407	**0.3964**	46.47
60	6	360	**0.0150**	0.0198	2.2547	**0.5103**	77.37
60	9	540	**0.0202**	0.0286	4.4260	**0.6930**	84.34
60	12	720	**0.0209**	0.0300	7.2143	**0.8724**	87.91
80	4	320	**0.0166**	0.0199	1.2156	**0.6214**	48.88
80	8	640	**0.0223**	0.0331	4.1513	**0.8543**	79.42
80	12	960	**0.0246**	0.0399	8.0393	**1.1153**	86.13
80	16	1280	**0.0300**	0.0488	13.5348	**1.2970**	90.42
100	5	500	**0.0239**	0.0289	1.5669	**0.9722**	37.95
100	10	1000	**0.0271**	0.0439	6.4084	**1.2111**	81.10
100	15	1500	**0.0304**	0.0540	12.8520	**1.5614**	87.85
100	20	2000	**0.0381**	0.0671	13.5619	**1.4767**	89.11
120	12	1440	**0.0333**	0.0564	9.2897	**1.6930**	81.78
120	18	2160	**0.0388**	0.0752	18.0824	**2.1187**	88.28
120	24	2880	**0.0464**	0.0932	19.8079	**1.9967**	89.92
140	7	980	**0.0331**	0.0428	2.6152	**1.6039**	38.67
140	21	2940	**0.0442**	0.0922	18.1729	**2.1653**	88.08
140	28	3920	**0.0566**	0.1296	29.6370	**2.6352**	91.11

Table 2.5: Low-rank matrix reconstruction error, sparse matrix reconstruction error and execution time of Algorithm 1, GoDec and ScaledGD.

			L Error			S Error			Time (s)		
N	k_0	k_1	Alg 1 Exact	GoDec	ScaledGD	Alg 1 Exact	GoDec	ScaledGD	Alg 1 Exact	GoDec	ScaledGD
200	5	500	**0.0442**	0.0458	0.0449	**0.5677**	0.9246	0.7379	0.0185	0.0187	**0.0134**
250	5	500	**0.0538**	0.0553	0.0544	**0.6176**	1.0208	0.7417	**0.0191**	0.0250	0.0225
300	5	500	**0.0641**	0.0654	0.0644	**0.6725**	1.1036	0.7741	0.0314	**0.0290**	0.0321
350	5	500	**0.0755**	0.0766	0.0757	**0.7307**	1.1955	0.8259	0.0436	**0.0411**	0.0454
400	5	500	**0.0852**	0.0863	0.0854	**0.7716**	1.2483	0.8578	0.0574	**0.0517**	0.0562
450	5	500	**0.0970**	0.0980	0.0972	**0.8038**	1.2918	0.9134	0.0792	**0.0712**	0.0751
500	5	500	**0.1083**	0.1093	0.1085	**0.8530**	1.3585	0.9746	0.0906	0.0918	**0.0895**
550	5	500	**0.1213**	0.1222	0.1215	**0.8918**	1.4021	1.0518	0.1138	**0.1049**	0.1083
600	5	500	**0.1322**	0.1331	0.1324	**0.9377**	1.4593	1.1210	0.1357	0.1384	**0.1228**
650	5	500	**0.1430**	0.1438	0.1433	**0.9624**	1.4842	1.1881	**0.1538**	0.1693	0.1590
700	5	500	**0.1554**	0.1562	0.1556	**1.0126**	1.5524	1.2712	0.1810	0.2022	**0.1587**
750	5	500	**0.1681**	0.1689	0.1682	**1.0244**	1.5587	1.3332	**0.3668**	0.5669	0.5734
800	5	500	**0.1812**	0.1820	**0.1812**	**1.0676**	1.6062	1.4105	**0.3395**	0.5000	1.1244
850	5	500	0.1918	0.1925	**0.1917**	**1.0967**	1.6372	1.4958	**0.9337**	1.0395	1.3067
900	5	500	0.2057	0.2064	**0.2056**	**1.1348**	1.6852	1.5847	1.7587	1.5853	**1.1520**
950	5	500	**0.2174**	0.2181	0.2175	**1.1543**	1.6942	1.6608	0.7749	**0.7494**	2.0345
1000	5	500	0.2306	0.2313	**0.2305**	**1.1783**	1.7207	1.7417	3.2104	**3.1600**	3.3916
2000	2	500	**0.5171**	0.5177	0.5173	**1.5707**	2.1098	3.6181	1.3195	1.3155	**1.0648**
4000	2	500	**1.3013**	1.3019	1.3018	**2.1207**	2.6438	7.9775	35.1148	36.9397	**19.1202**
6000	2	500	**2.3694**	2.3700	2.3704	**2.3058**	2.7742	11.9133	84.7058	87.7330	**64.5782**
8000	2	500	**3.5365**	3.5373	**3.5365**	**2.5880**	3.0463	16.8837	158.5785	160.0202	**132.8005**
10000	2	500	**4.8465**	4.8472	4.8486	**2.7586**	3.1967	21.5332	**133.3238**	145.8102	249.2882

Chapter 3

Compressed Sensing: A Discrete Optimization Approach

The work in this chapter is based on [24] which is joint work with Dimitris Bertsimas.

3.1 Introduction

The *Compressed Sensing* (CS) problem seeks to find a most sparse vector $\boldsymbol{x} \in \mathbb{R}^n$ that is consistent with a set of m linear equalities. CS is a fundamental problem in Statistics, Operations Research and Machine Learning which arises in numerous applications such as multi-label learning [82], medical resonance imaging [101], holography [38], climate monitoring [81], natural resource mining [139] and electrocardiogram signal acquisition [52] among many others. Formally, given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^m$, CS is given by [62]:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \|\boldsymbol{x}\|_0 \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \tag{3.1}$$

In the presence of noisy measurements, it is necessary to relax the equality constraint in (3.1), leading to the following formulation for $\epsilon > 0$:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \|\boldsymbol{x}\|_0 \quad \text{s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon. \tag{3.2}$$

This problem is sometimes referred to as sparse approximation in the literature [135] and trivially reduces to (3.1) for $\epsilon = 0$. CS allows signals to be reconstructed surprisingly well after sampling at a rate far below the Nyquist sampling rate by leveraging the inherent sparsity of most signals, either in the signal's latent space or in an appropriately defined transform space. For example, natural images tend to have a sparse representation in the wavelet domain, speech can be represented using a small number of coefficients in the Fourier transform domain and medical images can be represented sparsely in the Radon transform domain [120].

In Section 3.2, we will see that the vast majority of existing approaches to CS either rely on ℓ_1 based convex approximations to (3.2) or are greedy heuristics whereas the use of integer optimization techniques has gone relatively unexplored. In this work, we formulate

CS as:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x}\|_0 + \frac{1}{\gamma}\|\boldsymbol{x}\|_2^2 \quad \text{s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon, \tag{3.3}$$

where $\gamma > 0$ is a regularization parameter that in practice can either take a default value (e.g. $\sqrt{n}$) or be cross-validated by minimizing a validation metric [see, e.g., 114] to obtain strong out-of-sample performance [34]. A defining characteristic of the approach we present in this work is that we leverage techniques from integer optimization to exploit the inherent discreteness of formulation (3.3) rather than relying on more commonly studied approximate methods. Note that Problem (3.3) is a special case of the formulation given by:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x}\|_0 + \frac{1}{\gamma}\|\boldsymbol{W}\boldsymbol{x}\|_2^2 \quad \text{s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon. \tag{3.4}$$

where $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with nonnegative diagonal entries that should be interpreted as coordinate weights on the vector $\boldsymbol{x}$. Indeed, (3.4) reduces to (3.3) when we take $\boldsymbol{W} = \boldsymbol{I}$.

3.1.1 Contributions and Structure

In this chapter, we approach CS using mixed-integer second order cone optimization. We derive a second order cone relaxation of this problem and show that under mild conditions on the regularization parameter, the resulting relaxation is equivalent to the well studied basis pursuit denoising problem. We present a semidefinite relaxation that strengthens the second order cone relaxation and develop a custom branch and bound algorithm that leverages our second order cone relaxation to solve instances of CS to certifiable optimality. Our numerical results show that our approach produces solutions that are on average 6.22% more sparse than solutions returned by three state of the art benchmark methods on synthetic data in minutes. If we restrict the comparison to the best performing benchmark method on each problem instance, our approach produces solutions that are on average 3.10% more sparse. On real world ECG data, for a given ℓ_2 reconstruction error our approach produces

solutions that are on average 9.95% more sparse than benchmark methods (3.88% more sparse if only compared against the best performing benchmark), while for a given sparsity level our approach produces solutions that have on average 10.77% lower reconstruction error than benchmark methods in minutes (1.42% lower error if only compared against the best performing benchmark). On a real world multi-label classification task, our approach outperforms existing approaches in terms of accuracy, precision and recall. This increase in accuracy, precision and recall comes at the expense of a significant increase in running time of several orders of magnitude. Thus, for applications where runtime is not of critical importance, leveraging integer optimization can yield sparser and lower error solutions to CS than existing benchmarks.

The rest of this chapter is structured as follows. In Section 3.2, we review existing formulations and solution methods of the CS problem. In Section 3.3, we study how our regularized formulation of CS (3.3) connects to the commonly used formulation (3.2). We reformulate (3.3) exactly as a mixed-integer second order cone problem in Section 3.4 and present a second order cone relaxation in Section 3.4.1 and a stronger but more computationally expensive semidefinite cone relaxation in Section 3.4.2. We show that our second order cone relaxation is equivalent to the Basis Pursuit Denoising problem under mild conditions offering a new interpretation of this well studied method as a convex relaxation of our mixed-integer second order cone reformulation of (3.3). We leverage our second order cone relaxation to develop a custom branch and bound algorithm in Section 3.5 that can solve instances of (3.3) to certifiable optimality. In Section 3.6, we investigate the performance of our branch and bound algorithm against state of the art benchmark methods on synthetic data, real world ECG signal acquisition and real world multi-label classification

3.2 Literature Review

In this section, we review several key approaches from the literature that have been employed to solve the CS problem. As an exhaustive literature review is outside of the scope of this chapter, we focus our review on a handful of well studied approaches which will be used as benchmarks in this work. For a more detailed CS literature review, we refer the reader to [135].

The majority of existing approaches to the CS problem are heuristic in nature and generally can be classified as either convex approximations or greedy methods as we will see in this section. For these methods, associated performance guarantees require making strong statistical assumptions on the underlying problem data. Integer optimization has been given little attention in the CS literature despite its powerful modelling capabilities. [89] and [33] explore formulating Problem (3.2) as a mixed-integer linear program for the case when $\epsilon = 0$. However this approach relies on using the big-M method which requires estimating reasonable values for M and cannot immediately generalize to the setting where $\epsilon > 0$.

3.2.1 Basis Pursuit Denoising

A common class of CS methods rely on solving convex approximations of (3.2) rather than attempting to solve (3.2) directly. A popular approach is to use the ℓ_1 norm as a convex surrogate for the ℓ_0 norm [47, 53, 54, 62, 69]. This approximation is typically motivated by the observation that the unit ℓ_1 ball given by $\mathcal{B}_{\ell_1} = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_1 \leq 1\}$ is the convex hull of the non-convex set $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_0 \leq 1, \|\boldsymbol{x}\|_\infty \leq 1\}$. Replacing the ℓ_0 norm by the ℓ_1 norm in (3.2), we obtain:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \|\boldsymbol{x}\|_1 \quad \text{s.t. } \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon. \tag{3.5}$$

Problem (3.5) is referred to as Basis Pursuit Denoising and is a quadratically constrained convex optimization problem which can be solved efficiently using one of several off the shelf optimization packages. Basis Pursuit Denoising produces an approximate solution to Problem (3.2) by either directly returning the solution of (3.5) or by post-processing the solution of (3.5) to further sparsify the result. One such post-processing technique is a greedy rounding mechanism where columns of the matrix $\boldsymbol{A}$ are iteratively selected in the order corresponding to decreasing magnitude of the entries of the optimal solution of (3.5) until the selected column set of $\boldsymbol{A}$ is sufficiently large to produce a feasible solution to (3.2). Basis Pursuit Denoising is very closely related to the Lasso problem which is given by:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\lambda\|\boldsymbol{x}\|_1, \tag{3.6}$$

where $\lambda > 0$ is a tunable hyperparameter. Lasso is a statistical estimator commonly used for sparse regression as empirically, the optimal solution of Problem (3.6) tends to be sparse [131]. More recently, strong connections between Lasso and robust optimization have been established [13]. Basis Pursuit Denoising and Lasso are equivalent in that Lasso is obtained by relaxing the hard constraint in (3.5) and instead introducing a penalty term in the objective function. It is straightforward to show that for given input data $\boldsymbol{A}, \boldsymbol{b}$ and ϵ in (3.5), there exists a value $\lambda^\star > 0$ such that there exists a solution $\boldsymbol{x}^\star$ that is both optimal for (3.5) and (3.6) when the tunable parameter takes value $\lambda = \lambda^\star$.

Note that by taking $\epsilon = 0$, Problem (3.5) reduces to the well studied Basis Pursuit problem where the equality constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is enforced. A large body of work studies conditions under which the optimal solution of the Basis Pursuit problem is also an optimal solution of (3.1). For example, see [65], [63], [74], and [134]. One of the most well studied conditions under which this equivalence holds is when the input matrix $\boldsymbol{A}$ satisfies the Restricted Isometry Property (RIP). Formally, a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is said to satisfy RIP of

order s and parameter $\delta_s \in (0,1)$ if for every vector $\boldsymbol{x} \in \mathbb{R}^n$ such that $\|\boldsymbol{x}\|_0 \leq s$, we have

$$(1-\delta_s)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta_s)\|\boldsymbol{x}\|_2^2.$$

It has been established that if $\boldsymbol{A}$ satisfies RIP of order $2s$ and parameter $\delta_{2s} < 1/3$, then the optimal solution of the Basis Pursuit problem is also an optimal solution of (3.1) where s denotes the cardinality of this optimal solution [46]. While it has been shown that certain random matrices satisfy this desired RIP property with high probability [6, 76], RIP in general is not tractable to verify on arbitrary real world data.

3.2.2 Iterative Reweighted L1

Iterative Reweighted ℓ_1 minimization is an iterative method that can generate an approximate solution to (3.2) by solving a sequence of convex optimization problems that are very closely related to the Basis Pursuit Denoising problem given by (3.5) [3, 49, 107]. This approach falls in the class of convex approximation based methods for solving CS. The approach considers the weighted ℓ_1 minimization problem given by:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{W}\boldsymbol{x}\|_1 \quad \text{s.t.} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 \leq \epsilon, \tag{3.7}$$

where $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with nonnegative diagonal entries. Each diagonal entry $W_{ii} = w_i$ of $\boldsymbol{W}$ can be interpreted as a weighting of the i^{th} coordinate of the vector $\boldsymbol{x}$. Interpreting the ℓ_1 norm as a convex surrogate for the ℓ_0 norm, Problem (3.7) can be viewed as a relaxation of the non-convex problem given by

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{W}\boldsymbol{x}\|_0 \quad \text{s.t.} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 \leq \epsilon. \tag{3.8}$$

It is trivial to verify that when $\boldsymbol{W} = \alpha\boldsymbol{I}$, where $\alpha > 0$ and $\boldsymbol{I}$ is the n-by-n identity matrix, (3.8) and (3.7) reduce exactly to (3.2) and (3.5) respectively. Assuming the weights never

vanish, the non-convex Problems (3.2) and (3.8) have the same optimal solution, yet their convex relaxations (3.5) and (3.7) will generally have very different solutions. In this regard, the weights can be regarded as parameters that if chosen correctly can produce a better solution than (3.5). Iterative Reweighted ℓ_1 minimization proceeds as follows [49]:

1. Initialize the iteration count $t \leftarrow 0$ and the weights $w_i^{(0)} \leftarrow 1$.
2. Solve (3.7) with $\boldsymbol{W} = \boldsymbol{W}^{(t)}$. Let $\boldsymbol{x}^{(t)}$ denote the optimal solution.
3. Update the weights as $w_i^{(t+1)} \leftarrow \frac{1}{|x_i^{(t)}|+\delta}$ where $\delta > 0$ is a fixed parameter for numerical stability.
4. Terminate if t reaches a maximum number of iterations or if the iterates $\boldsymbol{x}^{(t)}$ have converged. Otherwise, increment t and return to Step 2.

It has been shown empirically that in many settings the solution returned by Iterated Reweighted ℓ_1 minimization outperforms the solution returned by Basis Pursuit Denoising by recovering the true underlying signal while requiring fewer measurements to be taken [49]. We note that this approach is an instance of a broader class of sparsifying iterative reweighted methods [55, 140, 141].

3.2.3 Orthogonal Matching Pursuit

Orthogonal Matching Pursuit (OMP) is a canonical greedy algorithm for obtaining heuristic solutions to (3.2) [104, 115]. Solving Problem (3.2) can be interpreted as determining the minimum number of columns from the input matrix $\boldsymbol{A}$ that must be selected such that the residual of the projection of the input vector $\boldsymbol{b}$ onto the span of the selected columns has ℓ_2 norm equal to at most $\sqrt{\epsilon}$. The OMP algorithm proceeds by first selecting the column of $\boldsymbol{A}$ that is most collinear with $\boldsymbol{b}$ and subsequently iteratively adding the column of $\boldsymbol{A}$ that is most collinear with the residual of the projection of $\boldsymbol{b}$ onto the subspace spanned by the selected columns until the norm of this residual is at most $\sqrt{\epsilon}$. Concretely, OMP proceeds

as follows where for an arbitrary collection of indices $\mathcal{I}_t \subseteq [n]$, we let $\boldsymbol{A}(\mathcal{I}_t) \in \mathbb{R}^{m \times |\mathcal{I}_t|}$ denote the matrix obtained by stacking the $|\mathcal{I}_t|$ columns of $\boldsymbol{A}$ corresponding to the indices in the set $\mathcal{I}_t$:

1. Initialize the iteration count $t \leftarrow 0$, the residual $\boldsymbol{r}_0 \leftarrow \boldsymbol{b}$ and the index set $\mathcal{I}_0 \leftarrow \emptyset$.

2. Select the column that is most collinear with the residual $i_t \leftarrow \arg\max_{i \in [n] \setminus \mathcal{I}_t} |\boldsymbol{a}_i^T \boldsymbol{r}_t|$ and update the index set $\mathcal{I}_{t+1} \leftarrow \mathcal{I}_t \cup i_t$.

3. Compute the projection of $\boldsymbol{b}$ onto the current set of columns

$$\boldsymbol{x}_{t+1} \leftarrow \left[\boldsymbol{A}(\mathcal{I}_{t+1})^T \boldsymbol{A}(\mathcal{I}_{t+1})\right]^\dagger \boldsymbol{A}(\mathcal{I}_{t+1})^T \boldsymbol{b},$$

   and update the residual $\boldsymbol{r}_{t+1} \leftarrow \boldsymbol{b} - \boldsymbol{A}(\mathcal{I}_{t+1})\boldsymbol{x}_{t+1}$.

4. Terminate if $\|\boldsymbol{r}_{t+1}\|_2^2 \leq \epsilon$, otherwise increment t and return to Step 2.

Conditions under which the solution returned by OMP is the optimal solution of (3.2) (either with high probability or with certainty) have been studied extensively [42, 134, 142]. Unfortunately, these conditions suffer from the same limitation as RIP in that in general they are not tractable to verify on real world data. A closely related method to OMP is Subspace Pursuit (SP) which is another greedy algorithm for obtaining a heuristic solution to (3.2) in the $\epsilon = 0$ setting but has the additional requirement that a target sparsity value K must be specified in advance [59]. SP is initialized by selecting the K columns of $\boldsymbol{A}$ that are most collinear with the vector $\boldsymbol{b}$. At each iteration, SP first computes the residual of the projection of $\boldsymbol{b}$ onto the current column set and then greedily updates up to K elements of the column set, repeating this process until doing so no longer decreases the norm of the residual.

3.3 Formulation Properties

In this section, we rigorously investigate connections between formulations (3.3) and (3.2) for the CS problem in the noisy setting. The only difference between formulations (3.2) and (3.3) is the inclusion of a ℓ_2 regularization term in the objective function in (3.3). We will see in Section 3.4 that the presence of this regularization term facilitates useful reformulations. Moreover, in the case of regression, [13] show that augmenting the ordinary least squares objective function with a ℓ_2 regularization penalty produces regression vectors that are robust against data perturbations which suggests the presence of such a regularization term may result in a similar benefit in (3.3). A natural question to ask is: under what conditions do problems (3.2) and (3.3) have the same solution? We answer this question in Theorem 23.

Theorem 23 *There exists a finite value $\gamma_0 < \infty$ such that for all $\bar{\gamma} \geq \gamma_0$, there exists a vector $\boldsymbol{x}^\star$ such that $\boldsymbol{x}^\star$ is an optimal solution of (3.2) and also an optimal solution of (3.3) where we set $\gamma = \bar{\gamma}$. Letting $\tilde{\boldsymbol{x}}$ denote a minimum norm solution to (3.2), we can take $\gamma_0 = \|\tilde{\boldsymbol{x}}\|_2^2$ and $\boldsymbol{x}^\star = \tilde{\boldsymbol{x}}$.*

Phrased simply, Theorem 23 establishes that there exists a finite value γ_0 such that if the regularization parameter γ in problem (3.3) is at least as large as γ_0, then there is a vector $\boldsymbol{x}^\star$ that is optimal to both problems (3.2) and (3.3). We note that this finite value γ_0 depends on the input data $\boldsymbol{A}, \boldsymbol{b}$ and ϵ.

Proof Consider any matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, vector $\boldsymbol{b} \in \mathbb{R}^m$ and scalar $\epsilon > 0$. Let Ω denote the set of optimal solutions to (3.2) and let $\mathcal{X}$ denote the feasible set of (3.2) and (3.3). We have $\mathcal{X} = \{\boldsymbol{x} : \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon\}$ and $\Omega \subseteq \mathcal{X}$. Let $\tilde{\boldsymbol{x}} \in \arg\min_{\boldsymbol{x} \in \Omega} \|\boldsymbol{x}\|_2^2$ and let $\gamma_0 = \|\tilde{\boldsymbol{x}}\|_2^2$. Since $\tilde{\boldsymbol{x}} \in \Omega$, $\tilde{\boldsymbol{x}}$ is an optimal solution to (3.2). It remains to show that $\tilde{\boldsymbol{x}}$ is optimal to (3.3) for all $\gamma \geq \gamma_0$.

Fix any $\gamma \geq \gamma_0$. To show that $\tilde{\boldsymbol{x}}$ is an optimal solution of (3.3), we will show that for all

$\bar{\boldsymbol{x}} \in \mathcal{X}$, we have

$$\|\tilde{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\tilde{\boldsymbol{x}}\|_2^2 \leq \|\bar{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\bar{\boldsymbol{x}}\|_2^2.$$

Fix an arbitrary $\bar{\boldsymbol{x}} \in \mathcal{X}$. Either $\bar{\boldsymbol{x}} \in \mathcal{X} \setminus \Omega$ or $\bar{\boldsymbol{x}} \in \Omega$. Suppose $\bar{\boldsymbol{x}} \in \mathcal{X} \setminus \Omega$. The definition of Ω and the fact that $\tilde{\boldsymbol{x}} \in \Omega$ implies

$$\|\tilde{\boldsymbol{x}}\|_0 < \|\bar{\boldsymbol{x}}\|_0 \implies \|\tilde{\boldsymbol{x}}\|_0 + 1 \leq \|\bar{\boldsymbol{x}}\|_0.$$

Next, note that since $\gamma \geq \gamma_0 = \|\tilde{\boldsymbol{x}}\|_2^2$, we have

$$\|\tilde{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\tilde{\boldsymbol{x}}\|_2^2 \leq \|\tilde{\boldsymbol{x}}\|_0 + 1 \leq \|\bar{\boldsymbol{x}}\|_0 \leq \|\bar{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\bar{\boldsymbol{x}}\|_2^2.$$

Suppose instead that $\bar{\boldsymbol{x}} \in \Omega$. The definition of Ω and $\tilde{\boldsymbol{x}}$ imply $\|\tilde{\boldsymbol{x}}\|_0 = \|\bar{\boldsymbol{x}}\|_0$ and $\|\tilde{\boldsymbol{x}}\|_2^2 \leq \|\bar{\boldsymbol{x}}\|_2^2$. It then follows immediately that $\|\tilde{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\tilde{\boldsymbol{x}}\|_2^2 \leq \|\bar{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\bar{\boldsymbol{x}}\|_2^2$. Thus, $\tilde{\boldsymbol{x}}$ is optimal to (3.3). This completes the proof. ■

Though Theorem 23 is useful in establishing conditions for the equivalence of problems (3.2) and (3.3), it is important to note that computing the value of γ_0 specified in the Theorem requires solving (3.2) which is difficult in general. Suppose we are solving problem (3.3) with some regularization parameter γ in the regime where $0 < \gamma < \gamma_0$. A natural question to ask is: how well does the solution of (3.3) approximate the solution of (3.2). We answer this question in Theorem 24.

Theorem 24 *Let $\tilde{\boldsymbol{x}}$ and γ_0 be as defined in Theorem 23, and let $\mathcal{X}$ denote the feasible set of (3.2) and (3.3). Specifically, $\tilde{\boldsymbol{x}}$ denotes a minimum norm solution to (3.2), $\gamma_0 = \|\tilde{\boldsymbol{x}}\|_2^2$ and $\mathcal{X} = \{\boldsymbol{x} : \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon\}$. Let $\lambda_\epsilon > 0$ be a value such that*

$$\arg\min_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x}\|_2^2 = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda_\epsilon \|\boldsymbol{x}\|_2^2.$$

Fix any value γ with $0 < \gamma < \gamma_0$. Suppose $\bar{\boldsymbol{x}}$ is an optimal solution to (3.3). Then we have

$$\|\tilde{\boldsymbol{x}}\|_0 \leq \|\bar{\boldsymbol{x}}\|_0 \leq \|\tilde{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\left(\|\tilde{\boldsymbol{x}}\|_2^2 - \left\|\left(\frac{1}{\lambda_\epsilon}\boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{b}\right\|_2^2\right).$$

Proof Fix any value γ with $0 < \gamma < \gamma_0$ and consider any optimal solution $\bar{\boldsymbol{x}}$ to (3.3). The inequality $\|\tilde{\boldsymbol{x}}\|_0 \leq \|\bar{\boldsymbol{x}}\|_0$ follows immediately from the optimality of $\tilde{\boldsymbol{x}}$ in (3.2). By the optimality of $\bar{\boldsymbol{x}}$, we must have

$$\|\bar{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\bar{\boldsymbol{x}}\|_2^2 \leq \|\tilde{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\tilde{\boldsymbol{x}}\|_2^2 \implies \|\bar{\boldsymbol{x}}\|_0 \leq \|\tilde{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}(\|\tilde{\boldsymbol{x}}\|_2^2 - \|\bar{\boldsymbol{x}}\|_2^2).$$

Thus, to establish the result we need only derive an upper bound for the term $(\|\tilde{\boldsymbol{x}}\|_2^2 - \|\bar{\boldsymbol{x}}\|_2^2)$, or equivalently to derive a lower bound for the term $\|\bar{\boldsymbol{x}}\|_2^2$. Since $\bar{\boldsymbol{x}} \in \mathcal{X}$, such a lower bound can be obtained by solving the optimization problem given by

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \boldsymbol{x} \in \mathcal{X} = \{\boldsymbol{x} : \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon\}. \tag{3.9}$$

This optimization problem has the same optimal solution as the ridge regression problem given by

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda_\epsilon\|\boldsymbol{x}\|_2^2. \tag{3.10}$$

for some value $\lambda_\epsilon > 0$. To see this, we form the Lagrangian for (3.9) $L(\boldsymbol{x}, \mu) = \|\boldsymbol{x}\|_2^2 + \mu(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 - \epsilon)$ and observe that the KKT conditions for $(\boldsymbol{x}, \mu) \in \mathbb{R}^n \times \mathbb{R}$ are given by

1. $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon$;
2. $\mu \geq 0$;
3. $\mu(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 - \epsilon) = 0 \implies \mu = 0$ or $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \epsilon$;
4. $\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \mu) = \boldsymbol{0} \implies \boldsymbol{x} = (\frac{1}{\mu}\boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$ if $\mu \neq 0$ and $\boldsymbol{x} = 0$ if $\mu = 0$.

We note that if $\boldsymbol{0} \in \mathcal{X}$, then $\boldsymbol{0}$ is trivially an optimal solution to (3.9) with optimal value

given by 0. This corresponds to the degenerate case. In the nondegenerate case, we have $\mathbf{0} \notin \mathcal{X}$. This condition, coupled with the first and fourth KKT conditions implies that at optimality, we have $\mu \neq 0$ and $\boldsymbol{x} = (\frac{1}{\mu}\boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. Next, we note that the unconstrained quadratic optimization problem given by (3.10) has an optimal solution $\boldsymbol{x}^\star$ given by $\boldsymbol{x}^\star = (\lambda_\epsilon \boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. Finally, we observe that the two preceding expressions are the same when $\lambda_\epsilon = \frac{1}{\mu} > 0$. Thus, we have

$$\|\bar{\boldsymbol{x}}\|_2^2 \geq \min_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_2^2 = \left\| \left(\frac{1}{\lambda_\epsilon}\boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{b} \right\|_2^2,$$

which implies that

$$\|\bar{\boldsymbol{x}}\|_0 \leq \|\tilde{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\left(\|\tilde{\boldsymbol{x}}\|_2^2 - \left\| \left(\frac{1}{\lambda_\epsilon}\boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{b} \right\|_2^2 \right).$$

This completes the proof. ■

Remark 25 *Though the statement of Theorem 24 is made for any fixed γ satisfying $0 < \gamma < \gamma_0$ with γ_0 given by Theorem 23, we note that the proof of Theorem 24 in fact generalizes to any $\gamma > 0$. This implies that the result of Theorem 23 holds for any γ_0' satisfying $\gamma_0' > \left(\|\tilde{\boldsymbol{x}}\|_2^2 - \left\| \left(\frac{1}{\lambda_\epsilon}\boldsymbol{I} + \boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{b} \right\|_2^2 \right)$. This is a stronger condition than the one established by Theorem 23 but has the drawback of depending on the value λ_ϵ which in general cannot be computed easily.*

Theorem 24 provides a worst case guarantee on the sparsity of the solution of (3.3) when the regularization parameter γ satisfies $0 < \gamma < \gamma_0$.

3.4 An Exact Reformulation and Convex Relaxations

In this section, we reformulate (3.4) as a mixed-integer second order cone optimization problem. We then employ the perspective relaxation [77] to construct a second order cone relaxation for (3.4) and demonstrate that under certain conditions on the regularization parameter γ, the resulting relaxation is equivalent to the Weighted Basis Pursuit Denoising problem given by (3.7). As a special case, we obtain a convex relaxation for (3.3) and demonstrate that it is equivalent to (3.5) under the same conditions on γ. Finally, we present a family of semidefinite relaxations to (3.4) using techniques from polynomial optimization.

To model the sparsity of the vector $\boldsymbol{x}$ in (3.4), we introduce binary variables $\boldsymbol{z} \in \{0,1\}^n$ and require that $x_i = z_i x_i$. This gives the following reformulation of (3.4):

$$\min_{\boldsymbol{z},\boldsymbol{x}\in\mathbb{R}^n} \sum_{i=1}^n z_i + \frac{1}{\gamma}\sum_{i=1}^n w_i^2 x_i^2 \quad \text{s.t.} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon,\ x_i = z_i x_i\ \forall\ i,\ z_i \in \{0,1\}\ \forall\ i. \tag{3.11}$$

The constraints $x_i = z_i x_i$ in (3.11) are non-convex in the decision variables $(\boldsymbol{x}, \boldsymbol{z})$. To deal with these constraints, we make use of the perspective reformulation [77]. Specifically, we introduce non-negative variables $\boldsymbol{\theta} \in \mathbb{R}^n_+$ where θ_i models x_i^2 and introduce the constraints $\theta_i z_i \geq x_i^2$, which are second order cone representable. Thus, if $z_i = 0$, we will have $x_i = 0$. This results in the following reformulation of (3.11):

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{x},\boldsymbol{\theta}\in\mathbb{R}^n} \quad & \sum_{i=1}^n z_i + \frac{1}{\gamma}\sum_{i=1}^n w_i^2 \theta_i \\ \text{s.t.} \quad & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon,\ x_i^2 \leq z_i\theta_i\ \forall\ i, \\ & z_i \in \{0,1\}\ \forall\ i,\ \theta_i \geq 0\ \forall\ i. \end{aligned} \tag{3.12}$$

Theorem 26 *The mixed-integer second order cone problem given by* (3.12) *is an exact reformulation of* (3.4).

Proof We show that given a feasible solution to (3.4), we can construct a feasible solution

to (3.12) that achieves the same objective value and vice versa.

Consider an arbitrary solution $\bar{\boldsymbol{x}}$ to (3.4). Let $\bar{\boldsymbol{z}} \in \mathbb{R}^n$ be the binary vector obtained by setting $\bar{z}_i = \mathbb{1}\{\bar{x}_i \neq 0\}$ and let $\bar{\boldsymbol{\theta}} \in \mathbb{R}^n$ be the vector obtained by setting $\bar{\theta}_i = \bar{x}_i^2$. We have $\|\boldsymbol{A}\bar{\boldsymbol{x}} - \boldsymbol{b}\|_2^2 \leq \epsilon$, $\bar{z}_i\bar{\theta}_i = \mathbb{1}\{\bar{x}_i \neq 0\} \cdot \bar{x}_i^2 = \bar{x}_i^2$, $\bar{\boldsymbol{z}} \in \{0,1\}^n$ and $\bar{\theta}_i \geq 0$ so the solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$ is feasible to (3.12). Lastly, notice that we have

$$\sum_{i=1}^n \bar{z}_i + \frac{1}{\gamma}\sum_{i=1}^n w_i^2\bar{\theta}_i = \sum_{i=1}^n \mathbb{1}\{\bar{x}_i \neq 0\} + \frac{1}{\gamma}\sum_{i=1}^n w_i^2\bar{x}_i^2 = \|\bar{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\boldsymbol{W}\bar{\boldsymbol{x}}\|_2^2,$$

where $\boldsymbol{W} = \text{diag}(w_1, \ldots, w_n)$. Thus, the solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$ is a feasible solution to (3.12) that achieves the same objective value as $\bar{\boldsymbol{x}}$ does in (3.4).

Consider now an arbitrary solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$ to (3.12). Since we have $\|\boldsymbol{A}\bar{\boldsymbol{x}} - \boldsymbol{b}\|_2^2 \leq \epsilon$, $\bar{\boldsymbol{x}}$ is feasible to (3.4). Next, we note that the constraints $x_i^2 \leq z_i\theta_i$ and $z_i \in \{0,1\}$ imply that $\bar{z}_i \geq \mathbb{1}\{\bar{x}_i \neq 0\}$ and $\bar{\theta}_i \geq \bar{x}_i^2$. Finally, we observe that

$$\|\bar{\boldsymbol{x}}\|_0 + \frac{1}{\gamma}\|\boldsymbol{W}\bar{\boldsymbol{x}}\|_2^2 = \sum_{i=1}^n \mathbb{1}\{\bar{x}_i \neq 0\} + \frac{1}{\gamma}\sum_{i=1}^n w_i^2\bar{x}_i^2 \leq \sum_{i=1}^n \bar{z}_i + \frac{1}{\gamma}\sum_{i=1}^n w_i^2\bar{\theta}_i.$$

Thus, the solution $\bar{\boldsymbol{x}}$ is a feasible solution to (3.4) that achieves an objective value equal to or less than the objective value that $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}, \bar{\boldsymbol{\theta}})$ achieves in (3.12). This completes the proof. ■

3.4.1 A Second Order Cone Relaxation

Problem (3.12) is a reformulation of Problem (3.4) where the problem's non-convexity is entirely captured by the binary variables $\boldsymbol{z}$. We now obtain a convex relaxation of (3.4) by solving (3.12) with $\boldsymbol{z} \in \text{conv}(\{0,1\}^n) = [0,1]^n$. This gives the following convex optimization problem:

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{x},\boldsymbol{\theta}\in\mathbb{R}^n} \quad & \sum_{i=1}^n z_i + \frac{1}{\gamma}\sum_{i=1}^n w_i^2\theta_i \\ \text{s.t.} \quad & \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 \leq \epsilon,\ x_i^2 \leq z_i\theta_i\ \forall\ i, \\ & 0 \leq z_i \leq 1\ \forall\ i,\ \theta_i \geq 0\ \forall\ i. \end{aligned} \tag{3.13}$$

A natural question to ask is how problem (3.13) compares to the Weighted Basis Pursuit Denoising problem given by (3.7), a common convex approximation for CS in the noisy setting. Surprisingly, under mild conditions on the regularization parameter γ, it can be shown that solving (3.13) is exactly equivalent to solving (3.7). This implies that though Basis Pursuit Denoising is typically motivated as a convex approximation to CS in the presence of noise, it can alternatively be understood as the natural convex relaxation of the mixed-integer second order cone problem given by (3.12) for appropriately chosen values of γ. We formalize this statement in Theorem 27.

Theorem 27 *There exists a finite value $\gamma_0 < \infty$ such that for all $\bar{\gamma} \geq \gamma_0$, any vector $\boldsymbol{x}^\star$ that is an optimal solution of (3.7) is also an optimal solution of (3.13). Let $\mathcal{X} = \{\boldsymbol{x} : \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 \leq \epsilon\}$, the feasible set of (3.7). We can take $\gamma_0 = \max_{x\in\mathcal{X}} \|\boldsymbol{W}\boldsymbol{x}\|_\infty^2$, where $\boldsymbol{W} = diag(w_1,\ldots,w_n)$.*

Proof Rewrite (3.13) as the two stage optimization problem given by (3.14).

$$\min_{\boldsymbol{x}\in\mathcal{X}} \quad \min_{\boldsymbol{z},\boldsymbol{\theta}\in\mathbb{R}^n} \sum_{i=1}^n z_i + \frac{1}{\gamma}\sum_{i=1}^n w_i^2\theta_i \quad \text{s.t.} \quad x_i^2 \leq z_i\theta_i\ \forall\ i,\ 0 \leq z_i \leq 1\ \forall\ i,\ \theta_i \geq 0\ \forall\ i. \tag{3.14}$$

Let $\gamma_0 = \max_{x\in\mathcal{X}} \|x\|_\infty^2$. To establish the result, we will show that for any $\boldsymbol{x} \in \mathcal{X}$ the optimal value of the inner minimization problem in (3.14) is a scalar multiple of the ℓ_1 norm of $\boldsymbol{W}\boldsymbol{x}$ provided that $\gamma \geq \gamma_0$.

Fix $\gamma \geq \gamma_0$ and consider any $\bar{\boldsymbol{x}} \in \mathcal{X}$. We make three observations that allow us to reformulate the inner minimization problem in (3.14):

1. The objective function of the inner minimization problem is separable.

2. For any i such that $\bar{x}_i = 0$, it is optimal to set $z_i = \theta_i = 0$ which results in no contribution to the objective function.

3. For any i such that $\bar{x}_i \neq 0$, we must have $z_i > 0$ and it is optimal to take $\theta_i = \frac{\bar{x}_i^2}{z_i}$.

We can therefore equivalently express the inner minimization problem of (3.14) as:

$$\min_{\boldsymbol{z} \in \mathbb{R}^n} \sum_{i:x_i \neq 0} \left[z_i + \frac{w_i^2}{\gamma} \cdot \frac{\bar{x}_i^2}{z_i} \right] \quad \text{s.t.} \quad 0 < z_i \leq 1 \ \forall \ i. \tag{3.15}$$

Let $f_i(z) = z + \frac{w_i^2}{\gamma} \cdot \frac{\bar{x}_i^2}{z}$. We want to minimize the function $f_i(z)$ over the interval $(0, 1]$ for all i such that $\bar{x}_i \neq 0$. Fix an arbitrary i satisfying $\bar{x}_i \neq 0$. We have $\frac{d}{dz} f_i(z) = 1 - \frac{w_i^2}{\gamma} \cdot \frac{\bar{x}_i^2}{z_i^2}$ and $\frac{d}{dz} f_i(z^\star) = 0 \iff z^\star = \pm \frac{w_i}{\sqrt{\gamma}} |\bar{x}_i|$. The condition $\gamma \geq \gamma_0 = \max_{x \in \mathcal{X}} \|\boldsymbol{W}\boldsymbol{x}\|_\infty^2$ and the fact that $\bar{\boldsymbol{x}} \in \mathcal{X}$ implies that $1 \geq \frac{w_i^2 \bar{x}_i^2}{\gamma}$ for all i. Thus, we have $0 < \frac{w_i}{\sqrt{\gamma}} |\bar{x}_i| \leq 1$. Let $\bar{z} = \frac{w_i}{\sqrt{\gamma}} |\bar{x}_i|$. Noting that $\lim_{z \to 0^+} f_i(z) = \infty$, the minimum of $f_i(z)$ over the interval $(0, 1]$ must occur either at 1 or $\bar{z}$. We have

$$\left(\frac{w_i}{\sqrt{\gamma}} |\bar{x}_i| - 1 \right)^2 \geq 0 \implies f_i(1) = \frac{w_i^2 \bar{x}_i^2}{\gamma} + 1 \geq \frac{2 w_i}{\sqrt{\gamma}} |\bar{x}_i| = f_i(\bar{z}).$$

Therefore, the minimum of $f_i(z)$ on $(0, 1]$ occurs at $\bar{z} = \frac{w_i}{\sqrt{\gamma}} |\bar{x}_i|$ and is equal to $f_i(\bar{z}) = \frac{2}{\sqrt{\gamma}} |\bar{x}_i|$. This allows us to conclude that the optimal value of (3.14) is given by:

$$\sum_{i:x_i \neq 0} \frac{2 w_i}{\sqrt{\gamma}} |\bar{x}_i| = \sum_{i=1}^n \frac{2 w_i}{\sqrt{\gamma}} |\bar{x}_i| = \frac{2}{\sqrt{\gamma}} \|\boldsymbol{W}\bar{\boldsymbol{x}}\|_1.$$

We have shown that for fixed $\boldsymbol{x} \in \mathcal{X}$, the optimal value of the inner minimization problem of (3.14) is a scalar multiple of the ℓ_1 norm of $\boldsymbol{W}\boldsymbol{x}$. We can rewrite (3.14) as

$$\min_{\boldsymbol{x} \in \mathcal{X}} \frac{2}{\sqrt{\gamma}} \|\boldsymbol{W}\bar{\boldsymbol{x}}\|_1, \tag{3.16}$$

which has the same set of optimal solutions as (3.7) because this set is invariant under scaling

of the objective function. This completes the proof. ■

Remark 28 *Note that by taking* $\boldsymbol{W} = \boldsymbol{I}$*, it immediately follows from Theorem 27 that any vector* $\boldsymbol{x}^\star$ *that is an optimal solution of* (3.5) *is also an optimal solution of* (3.13) *when we set* $\gamma \geq \gamma_0 = \max_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x}\|_\infty^2$.

Convex relaxations of non-convex optimization problems are helpful for two reasons. Firstly, a convex relaxation provides a lower (upper) bound to a minimization (maximization) problem which given a feasible solution to the non-convex optimization problem provides a certificate of worst case suboptimality. Secondly, convex relaxations can often be used as building blocks in the construction of global optimization algorithms or heuristics for non-convex optimization problems. Strong convex relaxations are desirable because they produce tighter bounds on the optimal value of the problem of interest (stronger certificates of worst case suboptimality) and generally lead to more performant global optimization algorithms and heuristics. Let $\mathcal{X}_1 = \{(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n : \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon, x_i^2 \leq z_i \theta_i \; \forall \; i, \theta_i \geq 0 \; \forall \; i\}$ and $\mathcal{X}_1 = \{(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n : \boldsymbol{z} \in \{0,1\}^n\}$. We can equivalently write (3.12) as:

$$\min_{(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) \in \mathcal{X}_1 \cap \mathcal{X}_2} \sum_{i=1}^n z_i + \frac{1}{\gamma} \sum_{i=1}^n w_i^2 \theta_i.$$

The strongest possible convex relaxation to (3.12) would be obtained by minimizing the objective function in (3.12) subject to the constraint that $(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) \in \text{conv}(\mathcal{X}_1 \cap \mathcal{X}_2)$. Since the objective function is linear in the decision variables, solving over $\text{conv}(\mathcal{X}_1 \cap \mathcal{X}_2)$ would produce an optimal solution to (3.12) since the objective would be minimized at an extreme point of $\text{conv}(\mathcal{X}_1 \cap \mathcal{X}_2)$ which by definition must be an element of $\mathcal{X}_1 \cap \mathcal{X}_2$. Unfortunately, in general it is hard to represent $\text{conv}(\mathcal{X}_1 \cap \mathcal{X}_2)$ explicitly. The relaxation given by (3.13) consists of minimizing the objective function of (3.12) subject to the constraint that $(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) \in (\text{conv}(\mathcal{X}_1) \cap \text{conv}(\mathcal{X}_2)) = \mathcal{X}_1 \cap \text{conv}(\mathcal{X}_2) \supseteq \text{conv}(\mathcal{X}_1 \cap \mathcal{X}_2)$.

Stronger convex relaxations to (3.12) can be obtained by introducing additional valid inequalities to (3.12) and then relaxing the integrality constraint on $\boldsymbol{z}$. For example, suppose we know a value $M \geq \gamma_0 = \max_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{W}\boldsymbol{x}\|_\infty^2$. We can use this value to introduce Big-M constraints similar in flavour to the formulation proposed by [89]. Under this assumption, it follows immediately that any feasible solution to (3.12) satisfies $-Mz_i \leq w_i x_i \leq Mz_i \; \forall \; i$. Thus, we can obtain another convex relaxtion of (3.12) by minimizing its objective function subject to the constraint $(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) \in \bar{\mathcal{X}}_1 \cap \text{conv}(\mathcal{X}_2) \supseteq \text{conv}(\mathcal{X}_1 \cap \mathcal{X}_2)$ where we define $\bar{\mathcal{X}}_1 = \mathcal{X}_1 \cap \{(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n : -Mz_i \leq w_i x_i \leq Mz_i \; \forall \; i\}$. Explicitly, with knowledge of such a value M we can solve

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{x},\boldsymbol{\theta}\in\mathbb{R}^n} \quad & \sum_{i=1}^n z_i + \frac{1}{\gamma}\sum_{i=1}^n w_i^2\theta_i \\ \text{s.t.} \quad & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon, \; x_i^2 \leq z_i\theta_i \; \forall \; i, \\ & -Mz_i \leq w_i x_i \leq Mz_i \; \forall \; i, \; 0 \leq z_i \leq 1 \; \forall \; i, \; \theta_i \geq 0 \; \forall \; i. \end{aligned} \tag{3.17}$$

Remark 29 *Given any input data $\boldsymbol{A}, \boldsymbol{b}, \epsilon$, if M satisfies $M \geq \gamma_0 = \max_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{W}\boldsymbol{x}\|_\infty^2$, then the optimal value of (3.17) is no less than the optimal value of (3.13). This follows immediately by noting that under the condition on M, the feasible set of (3.17) is contained in the feasible set of (3.13).*

The mixed-integer second order cone reformulation and convex relaxation introduced in this section lead to two approaches for solving (3.4) to certifiable optimality. On the one hand, solvers like `Gurobi` contain direct support for solving mixed-integer second order cone problems so problem (3.4) can be solved directly. On the other hand, it is possible to develop a custom branch and bound routine that leverages a modification of (3.7) to compute lower bounds. We illustrate this in Section 3.5. This custom, problem specific approach outperforms `Gurobi` because (3.5) is a more tractable problem than (3.13) due in part to the presence of fewer second order cone constraints which decreases the computational time spent computing lower bounds.

3.4.2 A Positive Semidefinite Cone Relaxation

In this section, we formulate (3.4) as a polynomial optimization problem and present a semidefinite relaxation using the sum of squares (SOS) hierarchy [93]. We show that this semidefinite relaxation is tighter than the second order cone relaxation presented previously.

Let $f(\boldsymbol{z}, \boldsymbol{x}) = \sum_{i=1}^{n} z_i + \frac{1}{\gamma} \sum_{i=1}^{n} w_i^2 \theta_i$ denote the objective function of (3.12). Notice that the constraint $\boldsymbol{z} \in \{0,1\}^n$ in (3.12) is equivalent to the constraint $\boldsymbol{z} \circ \boldsymbol{z} = \boldsymbol{z}$ (where $\circ$ denotes the element wise product). With this observation, we can express the feasible set of (3.12) as the semialgebraic set given by:

$$\Omega = \{(\boldsymbol{z}, \boldsymbol{x}) \in \mathbb{R}^n \times \mathbb{R}^n : \epsilon - \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \geq 0, x_i z_i - x_i = 0 \ \forall \ i, z_i^2 - z_i = 0 \ \forall \ i\}.$$

Thus, we can equivalently write (3.12) as $\min_{(\boldsymbol{z},\boldsymbol{x}) \in \Omega} f(\boldsymbol{z}, \boldsymbol{x})$. It is not difficult to see that the preceding optimization problem has the same optimal value as the problem given by

$$\max_{\lambda \in \mathbb{R}} \quad \lambda \text{ s.t. } f(\boldsymbol{z}, \boldsymbol{x}) - \lambda \geq 0 \ \forall \ (\boldsymbol{z}, \boldsymbol{x}) \in \Omega. \tag{3.18}$$

Problem (3.18) is a polynomial optimization problem that has the same optimal value as (3.4).

We can obtain tractable lower bounds for (3.18) by leveraging techniques from sum of squares (SOS) optimization [94, 95]. A polynomial $g \in \mathbb{R}[x]$ is said to be sum of squares (SOS) if for some $K \in \mathbb{N}$ there exists polynomials $\{g_k\}_{k=1}^{K} \subset \mathbb{R}[x]$ such that $g = \sum_{k=1}^{K} g_k^2$. We denote the set of all SOS polynomials as $\Sigma^2[x]$. Moreover, we denote the set of polynomials of degree at most d as $\mathbb{R}_d[x] \subset \mathbb{R}[x]$ and we denote the set of SOS polynomials of degree at most $2d$ as $\Sigma_d^2[x] \subset \Sigma^2[x]$. It is trivial to see that any polynomial that is SOS is globally non-negative. More generally, SOS polynomials can be utilized to model polynomial non-negativity over arbitrary semialgebraic sets. The quadratic module associated with the

semialgebraic set Ω is defined as:

$$QM(\Omega) = \Big\{ s_0(\boldsymbol{z},\boldsymbol{x}) + s_1(\boldsymbol{z},\boldsymbol{x})(\epsilon - \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2) + \sum_{i=1}^{n} t_i(\boldsymbol{z},\boldsymbol{x})(x_i z_i - x_i) + \sum_{i=1}^{n} r_i(\boldsymbol{z},\boldsymbol{x})(z_i^2 - z_i) : s_0, s_1 \in \Sigma^2[\boldsymbol{z},\boldsymbol{x}], t_i, r_i \in \mathbb{R}[\boldsymbol{z},\boldsymbol{x}] \ \forall\ i \Big\}. \tag{3.19}$$

It is straightforward to see that if a function $h(\boldsymbol{z},\boldsymbol{x})$ is an element of $QM(\Omega)$, then $h(\boldsymbol{z},\boldsymbol{x})$ is non-negative on Ω (since for points in Ω, $h(\boldsymbol{z},\boldsymbol{x})$ takes the form of the sum of two SOS polynomials). Thus, membership in $QM(\Omega)$ is a sufficient condition for non-negativity on Ω. We further define the restriction of $QM(\Omega)$ to polynomials of degree at most $2d$ as:

$$QM_d(\Omega) = \Big\{ s_0(\boldsymbol{z},\boldsymbol{x}) + s_1(\boldsymbol{z},\boldsymbol{x})(\epsilon - \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2) + \sum_{i=1}^{n} t_i(\boldsymbol{z},\boldsymbol{x})(x_i z_i - x_i) + \sum_{i=1}^{n} r_i(\boldsymbol{z},\boldsymbol{x})(z_i^2 - z_i) : s_0 \in \Sigma_d^2[\boldsymbol{z},\boldsymbol{x}], s_1 \in \Sigma_{d-1}^2[\boldsymbol{z},\boldsymbol{x}], t_i, r_i \in \mathbb{R}_{2d-2}[\boldsymbol{z},\boldsymbol{x}] \ \forall\ i \Big\}. \tag{3.20}$$

It is immediate that $QM_d(\Omega) \subset QM(\Omega)$ and membership in $QM_d(\Omega)$ provides a certificate of non-negativity on Ω. Importantly, given an arbitrary polynomial $h(\boldsymbol{z},\boldsymbol{x})$ it is possible to verify membership in $QM_d(\Omega)$ by checking feasibility of a semidefinite program. Thus, for any $d \in \mathbb{N}$, we obtain a semidefinite relaxation of (3.4) by solving:

$$\max_{\lambda \in \mathbb{R}} \ \lambda \text{ s.t. } f(\boldsymbol{z},\boldsymbol{x}) - \lambda \in QM_d(\Omega). \tag{3.21}$$

Since $QM_d(\Omega) \subset QM_{d+1}(\Omega)$, (3.21) produces an increasingly strong lower bound with increasing values of d. A natural question to ask is how the relaxation given by (3.21) compares to that given by (3.13). We answer this question in Theorem 30.

Theorem 30 *For every $d \geq 1$, the optimal value of (3.21) is no less than the optimal value of (3.13).*

Proof Without loss of generality, we take $\boldsymbol{W} = \boldsymbol{I}$. We prove the result for $\gamma \geq \gamma_0 =$

$\max_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_\infty^2$ though the result extends naturally to the case of arbitrary γ. Fix any $\epsilon > 0$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. By Theorem 27, (3.13) has the same optimal value as (3.7). Consider the dual of (3.7) which for $\boldsymbol{W} = \boldsymbol{I}$ is given by

$$\max_{\boldsymbol{\nu}\in\mathbb{R}^m} \quad \boldsymbol{b}^T\boldsymbol{\nu} - \sqrt{\epsilon}\|\boldsymbol{\nu}\|_2 \text{ s.t. } |\boldsymbol{\nu}^T A_i| \leq \frac{2}{\sqrt{\gamma}} \quad \forall\ i, \tag{3.22}$$

where A_i denotes the i^{th} column of $\boldsymbol{A}$. Strong duality holds between (3.22) and (3.7) since $\boldsymbol{\nu} = 0$ is always a strictly feasible point in (3.22) [35]. Fix $d = 1$. We will show that for any feasible solution to (3.22), we can construct a feasible solution to (3.21) that achieves the same objective value. Let $\bar{\boldsymbol{\nu}} \in \mathbb{R}^m$ denote an arbitrary feasible solution to (3.22). Define $\bar{r}_i(\boldsymbol{z},\boldsymbol{x}) = -1, \bar{t}_i(\boldsymbol{z},\boldsymbol{x}) = A_i^T\bar{\boldsymbol{\nu}}$ for all i, $\bar{s}_1(\boldsymbol{z},\boldsymbol{x}) = \frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}$ and define $\bar{s}_0(\boldsymbol{z},\boldsymbol{x}) = \text{monomial}(\boldsymbol{z},\boldsymbol{x},1)^T\bar{\boldsymbol{S}}\text{monomial}(\boldsymbol{z},\boldsymbol{x},1)$ where $\text{monomial}(\boldsymbol{x},\boldsymbol{z},1) \in \mathbb{R}[\boldsymbol{z},\boldsymbol{x}]^{2n+1}$ is the vector of monomials in $\mathbb{R}[\boldsymbol{z},\boldsymbol{x}]$ of degree at most 1 and $\bar{\boldsymbol{S}} \in \mathbb{R}^{2n+1\times 2n+1}$ is given by

$$\bar{\boldsymbol{S}} = \left[\begin{array}{c|c|c} \frac{1}{\gamma}\boldsymbol{I}_n + \frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}\boldsymbol{A}^T\boldsymbol{A} & \text{diag}\left(\frac{-\boldsymbol{A}^T\bar{\boldsymbol{\nu}}}{2}\right) & \frac{1}{2}\boldsymbol{A}^T\left(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\right) \\ \hline \text{diag}\left(\frac{-\boldsymbol{A}^T\bar{\boldsymbol{\nu}}}{2}\right) & \boldsymbol{I}_n & \boldsymbol{0}_n \\ \hline \frac{1}{2}\left(\bar{\boldsymbol{\nu}}^T - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}^T\right)\boldsymbol{A} & \boldsymbol{0}_n^T & \left(\frac{\boldsymbol{b}^T\boldsymbol{b}}{2\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2}\right)\|\bar{\boldsymbol{\nu}}\|_2 - \bar{\boldsymbol{\nu}}^T\boldsymbol{b} \end{array}\right].$$

Clearly, we have $\bar{t}_i, \bar{r}_i \in \mathbb{R}_0[\boldsymbol{z},\boldsymbol{x}]$ for all i and $\bar{s}_1 \in \Sigma_0^2[\boldsymbol{z},\boldsymbol{x}]$ because $\frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}} \geq 0$. We claim that $\bar{s}_0 \in \Sigma_1^2[\boldsymbol{z},\boldsymbol{x}]$. To see this, note that by the generalized Schur complement lemma (see Boyd et al. 1994, Equation 2.41), $\bar{\boldsymbol{S}} \succeq 0$ if and only if $\begin{pmatrix} \boldsymbol{I}_n & \boldsymbol{0}_n \\ \boldsymbol{0}_n^T & \sigma \end{pmatrix} \succeq 0$ and $\frac{1}{\gamma}\boldsymbol{I}_n + \frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}\boldsymbol{A}^T\boldsymbol{A} - \text{diag}\left(\frac{-\boldsymbol{A}^T\bar{\boldsymbol{\nu}}}{2}\right)^2 - \frac{1}{4\sigma}\boldsymbol{A}^T\left(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\right)\left(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\right)^T\boldsymbol{A} \succeq 0$ where we let $\sigma = \left(\frac{\boldsymbol{b}^T\boldsymbol{b}}{2\sqrt{\epsilon}} + \frac{\sqrt{\epsilon}}{2}\right)\|\bar{\boldsymbol{\nu}}\|_2 - \bar{\boldsymbol{\nu}}^T\boldsymbol{b}$. The first condition is satisfied if $\sigma \geq 0$. To see that this is always the case, notice that we can equivalently express σ as

$$\sigma = \left(\frac{\boldsymbol{b}^T\boldsymbol{b} + \epsilon}{2\sqrt{\epsilon}\|\boldsymbol{b}\|_2}\right)\|\boldsymbol{b}\|_2\|\bar{\boldsymbol{\nu}}\|_2 - \bar{\boldsymbol{\nu}}^T\boldsymbol{b}.$$

By Cauchy-Schwarz, we have $|\bar{\boldsymbol{\nu}}^T\boldsymbol{b}| \leq \|\boldsymbol{b}\|_2\|\bar{\boldsymbol{\nu}}\|_2$. Moreover, we have $0 \leq (\|\boldsymbol{b}\|_2 - \sqrt{\epsilon})^2 \implies \frac{\boldsymbol{b}^T\boldsymbol{b}+\epsilon}{2\sqrt{\epsilon}\|\boldsymbol{b}\|_2} \geq 1$. It follows immediately that $\sigma \geq 0$.

To establish the second condition, we first rewrite the Schur complement of $\bar{\boldsymbol{S}}$ as the sum of two matrices:

$$\left[\frac{1}{\gamma}\boldsymbol{I}_n - \operatorname{diag}\Big(\frac{-\boldsymbol{A}^T\bar{\boldsymbol{\nu}}}{2}\Big)^2\right] + \left[\frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}\boldsymbol{A}^T\boldsymbol{A} - \frac{1}{4\sigma}\boldsymbol{A}^T\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)^T\boldsymbol{A}\right],$$

where we let $\boldsymbol{\Phi}$ denote the first matrix in the sum and we let $\boldsymbol{\Psi}$ denote the second matrix. It suffices to show that $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are both positive semidefinite. The eigenvalues of $\boldsymbol{\Phi}$ are given by $\{\frac{1}{\gamma} - \frac{(\bar{\boldsymbol{\nu}}^T A_i)^2}{4}\}_{i=1}^n$. Thus, $\boldsymbol{\Phi}$ is positive semidefinite as long as $|\bar{\boldsymbol{\nu}}^T A_i| \leq \frac{2}{\sqrt{\gamma}}$ for all i which is guaranteed by the feasibility of $\bar{\boldsymbol{\nu}}$ in (3.22). To see that $\boldsymbol{\Psi} \succeq 0$, note that we can write $\boldsymbol{\Psi}$ as

$$\boldsymbol{\Psi} = \boldsymbol{A}^T\left(\frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}\boldsymbol{I}_m - \frac{1}{4\sigma}\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)^T\right)\boldsymbol{A}.$$

Since any matrix of the form $\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{X}$ is positive semidefinite provided that $\boldsymbol{Y}$ is positive semidefinite, it suffices to show that $\frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}\boldsymbol{I}_m \succeq \frac{1}{4\sigma}\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)^T$. Notice that the left hand side term of the matrix inequality has m eigenvalues of the form $\frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}$ while the right hand side term of the inequality is a rank 1 matrix with $\frac{1}{4\sigma}\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)^T\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)$ as its only non-zero eigenvalue. Recalling the definition of σ, it can be easily verified that $\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)^T\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big) = \frac{2\sigma\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}$ by simple algebraic manipulation. Accordingly, we have $\frac{\|\bar{\boldsymbol{\nu}}\|_2}{2\sqrt{\epsilon}}\boldsymbol{I}_m \succeq \frac{1}{4\sigma}\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)\Big(\bar{\boldsymbol{\nu}} - \frac{\|\bar{\boldsymbol{\nu}}\|_2}{\sqrt{\epsilon}}\boldsymbol{b}\Big)^T \implies \boldsymbol{\Psi} \succeq 0$. Thus, we have established that $\bar{\boldsymbol{S}} \succeq 0 \implies \bar{s}_0 \in \Sigma_1^2[\boldsymbol{z}, \boldsymbol{x}]$. Finally, we note that

$$\bar{s}_0 + \bar{s}_1(\epsilon - \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2) + \sum_{i=1}^n \bar{t}_i(x_i z_i - x_i) + \sum_{i=1}^n \bar{r}_i(z_i^2 - z_i) = f(\boldsymbol{z}, \boldsymbol{x}) - \boldsymbol{b}^T\bar{\boldsymbol{\nu}} + \sqrt{\epsilon}\|\bar{\boldsymbol{\nu}}\|_2.$$

We have shown that given an arbitrary feasible solution to (3.22), we can construct a solution that is feasible to (3.21) that achieves the same objective value. Note that this construction holds for any $d \geq 1$. Thus, for any $d \in \mathbb{N}$ the optimal value of (3.21) is at least as high as

the optimal value of (3.13). ■

We have shown that for any value of d, (3.21) produces a lower bound on the optimal value of (3.4) at least as strong as the bound given by (3.13). Unfortunately, (3.21) suffers from scalability challenges as it requires solving a positive semidefinite program with PSD constraints on matrices with dimension $\binom{2n+d}{d} \times \binom{2n+d}{d}$. We further discuss the scalability of (3.21) in Section 3.6. Note that since (3.21) is a maximization problem, any feasible solution (in particular a nearly optimal one) still consists of a valid lower bound on the optimal value of (3.4).

3.5 Branch and Bound

In this section, we propose a branch and bound algorithm in the sense of [92, 98] that computes certifiably optimal solutions to Problem (3.3) by solving the mixed-integer second order cone reformulation given by (3.12). We state explicitly our subproblem strategy in Section 3.5.1, before stating our overall algorithmic approach in Section 3.5.2.

3.5.1 Subproblems

Henceforth, for simplicity we will assume the weights w_i take value 1 for all i. What follows generalizes immediately to the setting where this assumption does not hold. Notice that (3.12) can be equivalently written as the two stage optimization problem given by $\min_{\boldsymbol{z}\in\{0,1\}^n} h(\boldsymbol{z})$ where we define $h(\boldsymbol{z})$ as:

$$\begin{aligned} h(\boldsymbol{z}) = \min_{\boldsymbol{x},\boldsymbol{\theta}\in\mathbb{R}^n} \quad & \sum_{i=1}^n z_i + \frac{1}{\gamma}\sum_{i=1}^n \theta_i \\ \text{s.t.} \quad & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon,\ x_i^2 \leq z_i\theta_i\ \forall\ i,\ \theta_i \geq 0\ \forall i. \end{aligned} \tag{3.23}$$

Note that in general, there exist binary vectors $\bar{\boldsymbol{z}} \in \{0,1\}^n$ such that the optimization problem in (3.23) is infeasible. For any such $\bar{\boldsymbol{z}}$, we define $h(\bar{\boldsymbol{z}}) = \infty$. We construct an enumeration tree that branches on the entries of the binary vector $\boldsymbol{z}$ which models the support of $\boldsymbol{x}$. A (partial or complete) sparsity pattern is associated with each node in the tree and is defined by disjoint collections $\mathcal{I}_0, \mathcal{I}_1 \subseteq [n]$. For indices $i \in \mathcal{I}_0$, we constrain $z_i = 0$ and for indices $j \in \mathcal{I}_1$, we constrain $z_j = 1$. We say that $\mathcal{I}_0$ and $\mathcal{I}_1$ define a complete sparsity pattern if $|\mathcal{I}_0| + |\mathcal{I}_1| = n$, otherwise we say that $\mathcal{I}_0$ and $\mathcal{I}_1$ define a partial sparsity pattern. A node in the tree is said to be terminal if its associated sparsity pattern is complete.

Each node in the enumeration tree has an associated subproblem, defined by the collections $\mathcal{I}_0$ and $\mathcal{I}_1$, which is given by:

$$\min_{\boldsymbol{z} \in \{0,1\}^n} \quad h(\boldsymbol{z}), \quad \text{s.t.} \quad z_i = 0 \; \forall \, i \in \mathcal{I}_0, z_j = 1 \; \forall \, j \in \mathcal{I}_1. \tag{3.24}$$

Note that if $\mathcal{I}_0 = \mathcal{I}_1 = \emptyset$, (3.24) is equivalent to (3.12) (under the assumetion that $w_i = 1$ for all i).

Subproblem Lower Bound

Let $\mathcal{I} = \mathcal{I}_0 \cup \mathcal{I}_1$. We obtain a lower bound for (3.24) by relaxing the binary variables that are not fixed (z_i such that $i \notin \mathcal{I}$) to take values within the interval $[0, 1]$. The resulting lower bound is given by

$$\begin{aligned} \min_{\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^n} \quad & \sum_{i=1}^n z_i + \frac{1}{\gamma} \sum_{i=1}^n \theta_i \\ \text{s.t.} \quad & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon, \; x_i^2 \leq z_i \theta_i \; \forall \; i, \; 0 \leq z_i \leq 1 \; \forall \; i \notin \mathcal{I}, \\ & z_i = 0 \; \forall \; i \in \mathcal{I}_0, \; z_i = 1 \; \forall \; i \in \mathcal{I}_1, \; \theta_i \geq 0 \; \forall \; i. \end{aligned} \tag{3.25}$$

Notice that for an arbitrary set $\bar{\mathcal{I}}_0 \subseteq [n]$, problems (3.24) and (3.25) are infeasible if and only if the set $\{\boldsymbol{x} : \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon, x_i = 0 \; \forall \, i \in \bar{\mathcal{I}}_0\}$ is empty. Moreover, observe it immediately follows that if (3.24) and (3.25) are infeasible for $\bar{\mathcal{I}}_0$, then they are also infeasible

for any set $\hat{\mathcal{I}}_0 \subseteq [n]$ satisfying $\bar{\mathcal{I}}_0 \subseteq \hat{\mathcal{I}}_0$. We use this observation in section 3.5.2 to generate feasibility cuts whenever an infeasible subproblem is encountered in the branch and bound tree. Using a similar argument as in the proof of Theorem 27, it can be shown that when $\gamma \geq \gamma_0 = \max_{x \in \mathcal{X}} \|\boldsymbol{x}\|_\infty^2$, (3.25) is equivalent to the convex optimization problem given by (3.26):

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} |\mathcal{I}_1| + \frac{1}{\gamma} \sum_{i \in \mathcal{I}_1} x_i^2 + \frac{2}{\sqrt{\gamma}} \sum_{i \notin \mathcal{I}} |x_i| \quad \text{s.t.} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \epsilon, \; x_i = 0 \; \forall \; i \in \mathcal{I}_0, \tag{3.26}$$

where if $\boldsymbol{x}^\star$ is optimal to (3.26), then $(\boldsymbol{z}^\star, \boldsymbol{x}^\star, \boldsymbol{\theta}^\star)$ is optimal to (3.25) taking $z_i^\star = \frac{|x_i|^\star}{\sqrt{\gamma}}$ and $\theta_i^\star = x_i^{\star 2}$. Problem (3.26) is a second order cone problem that emits the following dual:

$$\max_{\boldsymbol{\nu} \in \mathbb{R}^m} \; |\mathcal{I}_1| + \boldsymbol{b}^T \boldsymbol{\nu} - \sqrt{\epsilon} \|\boldsymbol{\nu}\|_2 - \frac{\gamma}{4} \boldsymbol{\nu}^T \sum_{i \in \mathcal{I}_1} (A_i A_i^T) \boldsymbol{\nu} \text{ s.t. } |\boldsymbol{\nu}^T A_i| \leq \frac{2}{\sqrt{\gamma}} \; \forall \; i \notin \mathcal{I}. \tag{3.27}$$

Strong duality holds between (3.26) and (3.27) since $\boldsymbol{\nu} = 0$ is always a strictly feasible point in (3.27) for any collections $\mathcal{I}_0, \mathcal{I}_1$ [35]. In our branch and bound implementation described in 3.5.2, we compute lower bounds by solving (3.26) using `Gurobi`. We note that depending on the solver employed, it may be beneficial to compute lower bounds using (3.27) in place of (3.26).

Subproblem Upper Bound

Recall that solving Problem (3.2) can be interpreted as determining the minimum number of columns from the input matrix $\boldsymbol{A}$ that must be selected such that the residual of the projection of the input vector $\boldsymbol{b}$ onto the span of the selected columns has ℓ_2 norm equal to at most $\sqrt{\epsilon}$. The same interpretation holds for Problem (3.3) under the assumption that the ℓ_2 regularization term in the objective is negligible.

Consider an arbitrary node in the branch and bound algorithm and let $\boldsymbol{x}^\star$ denote an optimal solution to (3.26). To obtain an upper bound to (3.24), we define an ordering on the columns of $\boldsymbol{A}$ and iteratively select columns from this ordering from largest to smallest

until the ℓ_2 norm of the residual of the projection of $\boldsymbol{b}$ onto the selected columns is less than $\sqrt{\epsilon}$. The ordering of the columns of $\boldsymbol{A}$ corresponds to sorting the entries of $\boldsymbol{x}^\star$ in decreasing absolute value. Specifically, we have $A_i \succeq A_j \iff |x_i^\star| \geq |x_j^\star|$. Algorithm 3 outlines this approach. For an arbitrary collection of indices $\mathcal{I}_t \subseteq [n]$, we let $\boldsymbol{A}(\mathcal{I}_t) \in \mathbb{R}^{m \times |\mathcal{I}_t|}$ denote the matrix obtained by stacking the $|\mathcal{I}_t|$ columns of $\boldsymbol{A}$ corresponding to the indices in the set $\mathcal{I}_t$. Specifically, if i_k denotes the k^{th} entry of $\mathcal{I}_t$, then the k^{th} column of $\boldsymbol{A}(\mathcal{I}_t)$ is A_i. Let $\boldsymbol{x}^{ub}$ denote the output of Algorithm 3. The objective value achieved by $\boldsymbol{x}^{ub}$ in (3.3) is the upper bound.

Algorithm 3: branch and bound Upper Bound

Data: $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m, \epsilon > 0$. An optimal solution $\boldsymbol{x}^\star$ of (3.26).
Result: $\bar{\boldsymbol{x}}$ is feasible to (3.3).
$\mathcal{I}_0 \leftarrow \emptyset$;
$\boldsymbol{r}_0 \leftarrow \boldsymbol{b}$;
$t \leftarrow 0$;
$\delta_0 \leftarrow \|\boldsymbol{r}_0\|_2^2$;
while $\delta_t > \epsilon$ **do**
  $i_t \leftarrow \arg\max_{i \in [n] \setminus \mathcal{I}_t} |x_i^\star|$;
  $\mathcal{I}_{t+1} \leftarrow \mathcal{I}_t \cup i_t$;
  $\boldsymbol{x}_{t+1} \leftarrow \left[\boldsymbol{A}(\mathcal{I}_{t+1})^T \boldsymbol{A}(\mathcal{I}_{t+1})\right]^\dagger \boldsymbol{A}(\mathcal{I}_{t+1})^T \boldsymbol{b}$;
  $\boldsymbol{r}_{t+1} \leftarrow \boldsymbol{b} - \boldsymbol{A}(\mathcal{I}_{t+1})\boldsymbol{x}_{t+1}$;
  $\delta_{t+1} \leftarrow \|\boldsymbol{r}_{t+1}\|_2^2$;
  $t \leftarrow t + 1$;
end
Define $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ as $\bar{x}(i_k) = x_t(k)$ for $i_k \in \mathcal{I}_t$ and $\bar{x}(i_k) = 0$ otherwise;
return $\bar{\boldsymbol{x}}$.

The computational bottleneck of Algorithm 3 is computing the matrix inverse of $\boldsymbol{A}(\mathcal{I}_t)^T \boldsymbol{A}(\mathcal{I}_t) \in \mathbb{R}^{|\mathcal{I}_t| \times |\mathcal{I}_t|}$ at each iteration. Doing so explicitly at each iteration t would require $O(|\mathcal{I}_t|^3)$ operations. Letting $k^\star = \|\boldsymbol{x}^{ub}\|_0$ where $\boldsymbol{x}^{ub}$ is the output of Algorithm 3, the total cost of executing these matrix inversions is

$$\sum_{t=1}^{k^\star} |\mathcal{I}_t|^3 = \sum_{t=1}^{k^\star} t^3 = \left[\frac{k^\star(k^\star+1)}{2}\right]^2 = O(k^{\star 4})$$

However, it is possible to accelerate the computation of these matrix inverses by leveraging

the fact that $\boldsymbol{A}(\mathcal{I}_t)$ and $\boldsymbol{A}(\mathcal{I}_{t+1})$ differ only by the addition of one column and leveraging block matrix inversion which states that for matrices $\boldsymbol{C} \in \mathbb{R}^{n_1 \times n_1}, \boldsymbol{D} \in \mathbb{R}^{n_2 \times n_2}$ and $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{n_1 \times n_2}$, we have:

$$\left[\begin{array}{c|c} \boldsymbol{C} & \boldsymbol{U} \\ \hline \boldsymbol{V}^T & \boldsymbol{D} \end{array}\right]^\dagger = \left[\begin{array}{c|c} \boldsymbol{C}^\dagger + \boldsymbol{C}^\dagger \boldsymbol{U}(\boldsymbol{D} - \boldsymbol{V}^T\boldsymbol{C}^\dagger\boldsymbol{U})^{-1}\boldsymbol{V}^T\boldsymbol{C}^\dagger & -\boldsymbol{C}^\dagger\boldsymbol{U}(\boldsymbol{D} - \boldsymbol{V}^T\boldsymbol{C}^\dagger\boldsymbol{U})^{-1} \\ \hline -(\boldsymbol{D} - \boldsymbol{V}^T\boldsymbol{C}^\dagger\boldsymbol{U})^{-1}\boldsymbol{V}^T\boldsymbol{C}^\dagger & (\boldsymbol{D} - \boldsymbol{V}^T\boldsymbol{C}^\dagger\boldsymbol{U})^{-1} \end{array}\right]$$

where it is assumed that the matrix $(\boldsymbol{D} - \boldsymbol{V}^T\boldsymbol{C}^\dagger\boldsymbol{U})$ is invertible [117]. Letting $n_1 = |\mathcal{I}_t|, n_2 = 1, \boldsymbol{C} = \boldsymbol{A}(\mathcal{I}_t)^T\boldsymbol{A}(\mathcal{I}_t), \boldsymbol{U} = \boldsymbol{V} = \boldsymbol{A}(\mathcal{I}_t)^T a_{i_t}$, and $\boldsymbol{D} = a_{i_t}^T a_{i_t}$, we can compute the matrix inverse of $\boldsymbol{A}(\mathcal{I}_{t+1})^T\boldsymbol{A}(\mathcal{I}_{t+1})$ using $O(|\mathcal{I}_t|^2 + m|\mathcal{I}_t|)$ operations. With this implementation, the total cost of executing matrix inversions in Algorithm 3 becomes

$$\sum_{t=1}^{k^\star} |\mathcal{I}_t|^2 + m|\mathcal{I}_t| = \sum_{t=1}^{k^\star} t^2 + m\sum_{t=1}^{k^\star} t = O(k^{\star 3} + mk^{\star 2})$$

which is a significant improvement over the naive $O(k^{\star 4})$ approach.

3.5.2 Branch and Bound Algorithm

Having stated how we can compute upper and lower bounds to (3.23) at each node in the enumeration tree, we are now ready to present the branch and bound algorithm in its entirety. Algorithm 4 describes our approach which is based on the implementation by [15]. Though branching rules and node selection rules for branch and bound algorithms form a rich literature [106], we follow the design of [15] and employ the most fractional branching rule and least lower bound node selection rule.

Explicitly, for an arbitrary non-terminal node p, let $\boldsymbol{z}^*$ be the optimal vector $\boldsymbol{z}$ of the node relaxation given by (3.25). We branch on entry $i^\star = \arg\min_{i \notin \mathcal{I}_0 \cup \mathcal{I}_1} |z_i - 0.5|$. When selecting a node to investigate, we select the node whose lower bound is equal to the global lower bound. If multiple such nodes exist, we choose arbitrarily from the collection of nodes satisfying this

condition. Suppose that a given node produces a subproblem (3.26) that is infeasible where we let $\bar{\mathcal{I}}_0$ correspond to the zero index set of this node. Note that this implies that all child nodes of this node will also produce infeasible subproblems. Accordingly, to prune this region of the parameter space entirely, we introduce the feasibility cut $\sum_{i\in\bar{\mathcal{I}}_0} z_i \geq 1$. Let $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_0 + \frac{1}{\gamma}\|\boldsymbol{x}\|_2^2$, the objective function of (3.3) and let $g(\mathcal{I}_0, \mathcal{I}_1)$ denote the optimal value of (3.26) for any collections $\mathcal{I}_0, \mathcal{I}_1 \subseteq [n], \mathcal{I}_0 \cap \mathcal{I}_1 = \emptyset$. The final objective value returned by Algorithm 4 is given by $\min_i f(\boldsymbol{x}_i)$ where $\{\boldsymbol{x}_i\}_i$ denotes the collection of feasible solutions produced by Algorithm 3 at any point during the execution of Algorithm 4. The output lower bound of Algorithm 4 is given by $\min_{(\mathcal{I}_0,\mathcal{I}_1)\in\mathcal{N}} g(\mathcal{I}_0, \mathcal{I}_1)$ where $\mathcal{N}$ denotes the set of non-discarded nodes upon the termination of Algorithm 4.

Let lb denote a lower bound to a given arbitrary minimization problem and let ub denote the objective value achieved by a feasible solution $\bar{\boldsymbol{x}}$ to the minimization problem. We call the solution $\bar{\boldsymbol{x}}$ a δ *globally optimal solution* to the given minimization problem if we have $lb \leq ub \leq (1+\delta)lb$.

Theorem 31 *Algorithm 4 terminates in a finite number of iterations and returns a δ globally optimal solution to (3.2).*

Proof The proof follows the proof of Theorem 21 in [15]. Note that Algorithm 4 can never visit a node more than once and that there is a finite number of partial and complete sparsity patterns (each corresponding to a possible node in the tree) because the set $\{0,1\}^n$ is discrete. Thus, Algorithm 4 terminates in a finite number of iterations. Moreover, upon termination we must have $\frac{ub-lb}{ub} \leq \delta$, therefore the output solution $\bar{\boldsymbol{x}}$ is δ globally optimal to problem (3.3) by definition since lb consists of a global lower bound and $\bar{\boldsymbol{x}}$ is feasible to (3.3). ■

We conclude the discussion of Algorithm 4 by describing two modifications that accelerate its execution time (or equivalently, improve its scalability) at the expense of sacrificing the

Algorithm 4: Optimal Compressed Sensing

Data: $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^m, \epsilon, \gamma \in \mathbb{R}^+$. Tolerance parameter $\delta \geq 0$
Result: $\bar{\boldsymbol{x}}$ solves (3.3) within the optimality tolerance δ.
if $\|(\boldsymbol{I} - \boldsymbol{A}\big[\boldsymbol{A}^T\boldsymbol{A}\big]^\dagger \boldsymbol{A}^T)\boldsymbol{b}\|_2^2 > \epsilon$ **then**
  return $\emptyset$
end
if $\|\boldsymbol{b}\|_2^2 \leq \epsilon$ **then**
  return 0
end
$p_0 \leftarrow (\mathcal{I}_0, \mathcal{I}_1) = (\emptyset, \emptyset)$;
$\mathcal{N} \leftarrow \{p_0\}$;
$lb \leftarrow$ optimal value of (3.26);
$\bar{\boldsymbol{x}} \leftarrow$ solution returned by Algorithm 3;
$ub \leftarrow f(\bar{\boldsymbol{x}})$;
while $\frac{ub-lb}{ub} > \epsilon$ **do**
  Select $(\mathcal{I}_0, \mathcal{I}_1) \in \mathcal{N}$ according to the node selection rule;
  Select an index $i \notin \mathcal{I}_0 \cup \mathcal{I}_1$ according to the branching rule;
  for $k = 0, 1$ **do**
    $l \leftarrow (k+1) \bmod 2$;
    newnode $\leftarrow \Big((\mathcal{I}_k \cup i), \mathcal{I}_l\Big)$;
    if *newnode violates an existing feasibility cut* **then**
      Continue;
    end
    if *newnode is infeasible* **then**
      Add the feasibility cut $\sum_{i\in\mathcal{I}_0} z_i \geq 1$;
    end
    lower $\leftarrow$ lowerBound(newnode);
    upper $\leftarrow$ upperBound(newnode) with feasible point $\boldsymbol{x}^\star$;
    if upper $<$ *ub* **then**
      $ub \leftarrow upper$;
      $\bar{\boldsymbol{x}} \leftarrow \boldsymbol{x}^\star$;
      Remove any node in $\mathcal{N}$ with $lower \geq ub$;
    end
    if lower $<$ *ub* **then**
      Add newnode to $\mathcal{N}$;
    end
  end
  Remove $(\mathcal{I}_0, \mathcal{I}_1)$ from $\mathcal{N}$;
  Update *lb* to be the lowest value of *lower* over $\mathcal{N}$;
end
return $\bar{\boldsymbol{x}}$, *lb*.

universal optimality guarantee by drawing on techniques from the high dimensional sparse machine learning literature [21] and the deep learning literature [73].

Backbone Optimization

Note that the total number of terminal nodes in the branch and bound tree is at most $\sum_{k=1}^{n} \binom{n}{k} = 2^n$ in the worst case so the total number of nodes can be upper bounded by $2^{n+1} - 1$. Since the runtime of Algorithm 4 (and the feasible space) is proportional to the number of nodes explored which grows exponentially in n, reducing n leads to reduced run time. Observe that if we knew in advance that the support of the optimal solution to (3.3) was contained within a set of cardinality less than n, then we could run Algorithm (4) on the corresponding reduced feature set which would result in improving the runtime of (4) while preserving its optimality guarantee. Formally, let $\boldsymbol{x}^\star$ denote an optimal solution to (3.3). If we know a priori that $\text{support}(\boldsymbol{x}^\star) \subseteq \mathcal{I} \subset [n]$, then we can pass $\boldsymbol{A}(\mathcal{I})$ to Algorithm 4 in place of $\boldsymbol{A}$ without discarding $\boldsymbol{x}^\star$ from the feasible set. The speed up can be quite significant when $|\mathcal{I}| \ll n$.

Knowing with certainty that $\text{support}(\boldsymbol{x}^\star) \subseteq \mathcal{I} \subset [n]$ a priori is too strong an assumption, however a more reasonable assumption is knowing a priori that with high probability there exists a good solution $\bar{\boldsymbol{x}}$ with $\text{support}(\bar{\boldsymbol{x}}) \subseteq \mathcal{I} \subset [n]$. In this setting, we can still pass $\boldsymbol{A}(\mathcal{I})$ to Algorithm 4 and benefit from an improved runtime at the expense of sacrificing optimality guarantees. In this setting, the columns of $\boldsymbol{A}(\mathcal{I})$ can be interpreted as a backbone for (3.3) [21]. In practice, $\mathcal{I}$ can be taken to be the set of features selected by some heuristic method. In Section 3.6, we take $\mathcal{I} = \{i : |\bar{x}_i| \geq 10^{-6}\}$ where $\bar{\boldsymbol{x}}$ is an optimal solution to (3.5).

Early Stopping

A common property of branch and bound algorithms is that the algorithm quickly arrives at an optimal (or near-optimal) solution early during the optimization procedure and spends the majority of its execution time improving the lower bound to obtain a certificate of optimality.

Accordingly, this motivates halting Algorithm 4 before it terminates and taking its upper bound at the time of termination to be its output. Doing so is likely to still yield a high quality solution while reducing the Algorithm's runtime. In Section 3.6, we place an explicit time limit on Algorithm 4 and return the current upper bound if the Algorithm has not already terminated before reaching the time limit. Note that this approach shares strong connections with early stopping in the training of neural networks [73]. A well studied property of over-parameterized neural networks is that as the optimization procedure progresses, the error on the training data continues to decrease though the validation error plateaus and sometimes even increases. Given that the validation error is the metric of greater import, a common network training technique is to stop the optimization procedure after the validation error has not decreased for a prespecified number of iterations. To illustrate the connection in the case of Algorithm 4, the upper bound loosely plays the role of the validation error while the lower bound loosely plays the role of the training error. Note that the neural network literature suggests an alternate approach to early stopping Algorithm 4 (instead of an explicit time limit) by terminating the algorithm after the upper bound has remained unchanged after visiting some prespecified number of nodes in the enumeration tree.

3.6 Computational Results

We evaluate the performance of our branch and bound algorithm (Algorithm 4, with $\gamma = \sqrt{n}$), our second order cone lower bound (3.13) (with $\gamma = \sqrt{n}$) and our semidefinite lower bound (3.21) (with $\gamma = \sqrt{n}$ and $d = 1$) implemented in Julia 1.5.2 using the `JuMP.jl` package version 0.21.7, using `Gurobi` version 9.0.3 to solve all second order cone optimization (sub)problems and using `Mosek` version 9.3 to solve all semidefinite optimization problems. We compare our methods against Basis Pursuit Denoising (BPD) given by (3.5), Iterative Reweighted ℓ_1 Minimizaton (IRWL1) described in Section 3.2.2 and Orthogonal Matching Pursuit (OMP) described in Section 3.2.3. We perform experiments using synthetic data

and two real world data sets. We conduct our experiments on MIT's Supercloud Cluster [123], which hosts Intel Xeon Platinum 8260 processors and cores with 4GB RAM. To bridge the gap between theory and practice, we have made our code freely available on GitHub at https://github.com/NicholasJohnson2020/DiscreteCompressedSensing.jl.

3.6.1 Synthetic Data Experiments

To evaluate the performance of Algorithm 4, BPD, IRWL1 and OMP on synthetic data, we consider the sparsity of the solution returned by each method, its accuracy (ACC), true positive rate (TPR) and true negative rate (TNR). Let $\boldsymbol{x}^{true} \in \mathbb{R}^n$ denote the ground truth and consider an arbitrary vector $\hat{\boldsymbol{x}} \in \mathbb{R}^n$. Let $\mathcal{I}^{true} = \{i : |x_i^{true}| > 10^{-4}\}$, $\hat{\mathcal{I}} = \{i : |\hat{x}_i| > 10^{-4}\}$. The sparsity of $\hat{\boldsymbol{x}}$ is given by $|\hat{\mathcal{I}}|$. We define the accuracy of $\hat{\boldsymbol{x}}$ as

$$ACC(\hat{\boldsymbol{x}}) = \frac{\sum_{i \in \mathcal{I}^{true}} \mathbb{1}\{|\hat{x}_i| > 10^{-4}\} + \sum_{i \notin \mathcal{I}^{true}} \mathbb{1}\{|\hat{x}_i| \leq 10^{-4}\}}{n}.$$

Similarly, we define the true positive rate of $\hat{\boldsymbol{x}}$ as

$$TPR(\hat{\boldsymbol{x}}) = \frac{\sum_{i \in \mathcal{I}^{true}} \mathbb{1}\{|\hat{x}_i| > 10^{-4}\}}{|\hat{\mathcal{I}}|},$$

and we define the true negative rate of $\hat{\boldsymbol{x}}$ as

$$TNR(\hat{\boldsymbol{x}}) = \frac{\sum_{i \notin \mathcal{I}^{true}} \mathbb{1}\{|\hat{x}_i| \leq 10^{-4}\}}{n - |\hat{\mathcal{I}}|}.$$

To evaluate the performance of (3.13) and (3.21), we consider the strength of the lower bound and execution time of each method. We seek to answer the following questions:

1. How does the performance of Algorithm 4 compare to state-of-the-art methods such as BPD, IRWL1 and OMP on synthetic data?

2. How is the performance of Algorithm 4 affected by the number of features n, the

underlying sparsity k of the ground truth, and the tolerance parameter ϵ?

3. How does the strength of the lower bound produced by (3.21) compare to that produced by (3.13)?

Synthetic Data Generation

To generate synthetic data $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, we first select a random subset of indices $\mathcal{I}^{true} \subset [n]$ that has cardinality k ($|\mathcal{I}^{true}| = k$) and sample $x_i \sim N(0, \frac{\sigma^2}{n})$ for $i \in \mathcal{I}^{true}$ (for $i \notin \mathcal{I}^{true}$, we fix $x_i = 0$). Next, we sample $A_{ij} \sim N(0, \frac{\sigma^2}{n})$ where $\sigma > 0$ is a parameter that controls the signal to noise ratio. We fix $\sigma = 10$ and $m = 100$ throughout all experiments unless stated otherwise. Next, we set $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$ where $n_j \sim N(0, \sigma^2)$. Finally, we set $\epsilon = \alpha \|\boldsymbol{b}\|_2^2$. $\alpha \in [0, 1]$ is a parameter that can be thought of as controlling the proportion of observations that are allowed to go unexplained by a solution to (3.3).

Sensitivity to n

We present a comparison of Algorithm 4 with BPD, IRWL1 and OMP as we vary the number of features n. In these experiments, we fixed $k = 10$, and $\alpha = 0.2$ across all trials. We varied $n \in \{100, 200, 300, 400, 500, 600, 700, 800\}$ and we performed 100 trials for each value of n. We give Algorithm 4 a cutoff time of 10 minutes. For IRWL1, we terminate the algorithm after the 50^{th} iteration or after two subsequent iterates are equal up to numerical tolerance. Formally, letting $\bar{\boldsymbol{x}}_t$ denote the iterate after iteration t of IRWL1, we terminate the algorithm if either $t > 50$ or if $\|\bar{\boldsymbol{x}}_t - \bar{\boldsymbol{x}}_{t-1}\|_2 \leq 10^{-6}$. Additionally, we further sparsify the solutions returned by BPD (respectively IRWL1) by performing a greedy rounding following the procedure defined by Algorithm 3 where we pass the solution returned by BPD (respectively IRWL1) as input to the algorithm in place of an optimal solution to (3.26).

We report the sparsity, accuracy (ACC), true positive rate (TPR) and true negative rate (TNR) for each method in Figure 3.1. We additionally report the sparsity accuracy

and execution time for each method in Tables 3.1, 3.2 and 3.3. The performance metric of greatest interest is the sparsity. Our main findings from this set of experiments are:

1. Algorithm 4 systematically produces sparser solutions than OMP, IRWL1 and BPD. This trend holds in all but one trial (see Table 3.1). Algorithm 4 on average produces solutions that are 2.71% more sparse than OMP, 16.62% more sparse than BPD and 6.04% more sparse than IRWL1. BPD is the poorest performing method in terms of sparsity of the fitted solutions. We remind the reader that sparsity is computed only after a greedy rounding of the BPD (respectively IRWL1) solution. The sparsity of the BPD (respectively IRWL1) solution prior to rounding is much greater. Indeed, before further sparsifying the BPD (respectively IRWL1) solution, the solution returned by Algorithm 4 is on average 66.33% (respectively 6.21%) more sparse than the BPD (respectively IRWL1) solution. The sparsity of solutions returned by all methods increases as the number of features n increases.

2. Algorithm 4 marginally outperforms the benchmark methods on accuracy with the exception of the first two parameter configurations ($n = 100$ and $n = 200$, see Table 3.2). The accuracy of all methods tends to trend upwards with increasing n.

3. The TPR and TNR of all methods are roughly comparable across these experiments. The TPR of all methods decreases while the TNR increases as the number of features n is increased. The sharp drop off in the TPR of all methods as n increases from 100 to 400 is consistent with the number of features selected by each method increasing sharply as n increases from 100 to 400. The latter results in the denominator used in the TPR computation to grow sharply which produces the observed behavior.

Sensitivity to k

We present a comparison of Algorithm 4 with BPD, IRWL1 and OMP as we vary k the sparsity of the underlying ground truth signal. In these experiments, we fixed $n = 200$ and

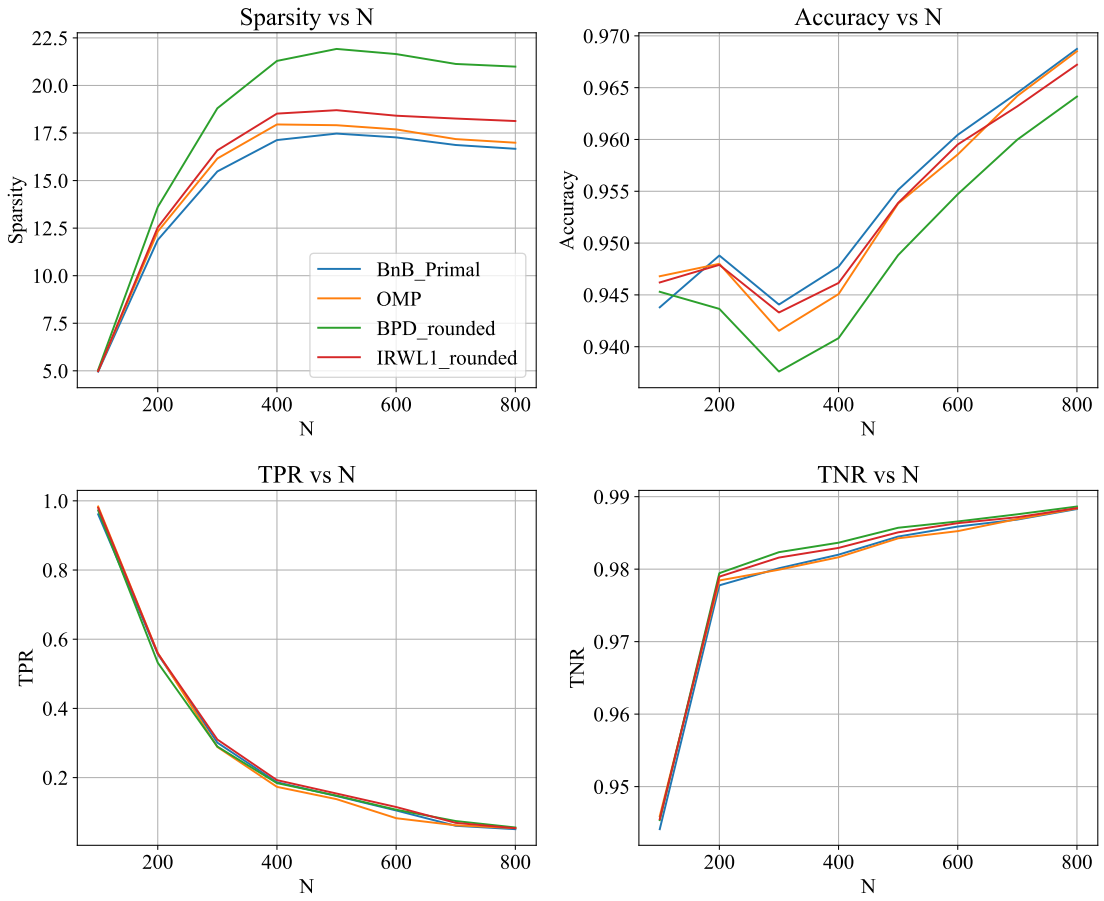

Figure 3.1: Sparsity (top left), accuracy (top right), true positive rate (bottom left) and true negative rate (bottom right) versus n with $k = 10$, and $\alpha = 0.2$. Averaged over 100 trials for each parameter configuration.

$\alpha = 0.2$ across all trials. We varied $k \in \{10, 15, 20, 25, 30, 35, 40, 45, 50, 55\}$ and we performed 100 trials for each value of k. We give Algorithm 4 a cutoff time of 10 minutes.

We report the sparsity, accuracy (ACC), true positive rate (TPR) and true negative rate (TNR) for each method in Figure 3.2. We additionally report the sparsity, accuracy and execution time for each method in Tables 3.4, 3.5 and 3.6. Our main findings from this set of experiments are:

1. Consistent with the results in the previous section, Algorithm 4 systematically produces sparser solutions than OMP, IRWL1 and BPD. This trend holds across trials (see Table 3.4. Algorithm 4 on average produces solutions that are 4.78% more sparse than OMP,

Table 3.1: Comparison of the sparsity of solutions returned by (4), OMP, IRWL1 and BPD for different values of n. Averaged over 100 trials for each parameter configuration.

	Sparsity			
N	Algorithm 4	OMP	IRWL1	BPD
100	**5.0**	**5.0**	**5.0**	**5.0**
200	**11.9**	12.3	12.5	13.6
300	**15.5**	16.2	16.6	18.8
400	**17.1**	17.9	18.5	21.3
500	**17.5**	17.9	18.7	21.9
600	**17.3**	17.7	18.4	21.6
700	**16.9**	17.2	18.3	21.1
800	**16.7**	17.0	18.1	21.0

Table 3.2: Comparison of the accuracy of solutions returned by (4), OMP, IRWL1 and BPD for different values of n. Averaged over 100 trials for each parameter configuration.

	Accuracy			
N	Algorithm 4	OMP	IRWL1	BPD
100	0.944	**0.947**	0.946	0.945
200	**0.949**	0.948	0.948	0.944
300	**0.944**	0.942	0.943	0.938
400	**0.948**	0.945	0.946	0.941
500	**0.955**	0.954	0.954	0.949
600	**0.960**	0.959	**0.960**	0.955
700	**0.965**	0.964	0.963	0.960
800	**0.969**	**0.969**	0.967	0.964

10.73% more sparse than BPD and 4.20% more sparse than IRWL1. Before further sparsifying the BPD (respectively IRWL1) solution, the solution returned by Algorithm 4 is on average 62.97% (respectively 4.29%) more sparse than the BPD (respectively IRWL1) solution. BPD is again the poorest performing method in terms of sparsity of the fitted solutions. IRWL1 and OMP produce comparably sparse solutions. The sparsity of solutions returned by all methods initially decreases than subsequently increases as the sparsity level k of the ground truth signal increases.

2. Algorithm 4 is competitive with OMP and IRWL1 on accuracy and slightly outperforms

Table 3.3: Comparison of the execution time of solutions returned by (4), OMP, IRWL1 and BPD for different values of n. Averaged over 100 trials for each parameter configuration.

	Execution Time (milliseconds)			
N	Algorithm 4	OMP	IRWL1	BPD
100	2048.646	**5.717**	463.636	146.111
200	334804.020	**13.212**	1109.263	234.263
300	574501.859	**25.141**	1630.212	297.849
400	601792.939	**42.919**	2181.636	351.717
500	601424.020	**72.535**	2435.141	405.131
600	601451.838	**110.364**	3118.465	433.626
700	601572.848	**166.525**	3674.980	504.626
800	601716.929	**231.980**	3865.788	540.859

BPD on accuracy for larger values of k The accuracy of all methods trends downwards with increasing k, suggesting that the feature identification problem becomes more challenging for larger values of k in this regime.

3. The TPR and TNR of Algorithm 4, OMP, and IRWL1 are comparable across these experiments. The TPR and TNR of BPD is competitive with the other methods for small values of k, but slightly deteriorates for larger values of k.

Table 3.4: Comparison of the sparsity of solutions returned by (4), OMP, IRWL1 and BPD for different values of k. Averaged over 100 trials for each parameter configuration.

	Sparsity			
K	Algorithm 4	OMP	IRWL1	BPD
10	**11.7**	12.2	12.3	13.4
15	**10.4**	10.7	11.0	11.7
20	**10.4**	10.8	10.7	11.3
25	**11.3**	11.8	11.8	12.5
30	**11.5**	12.0	11.9	12.8
35	**12.1**	12.8	12.6	13.5
40	**12.4**	13.2	13.0	13.9
45	**13.4**	14.3	14.1	15.2
50	**13.8**	14.8	14.4	15.6
55	**14.6**	15.6	15.4	16.8

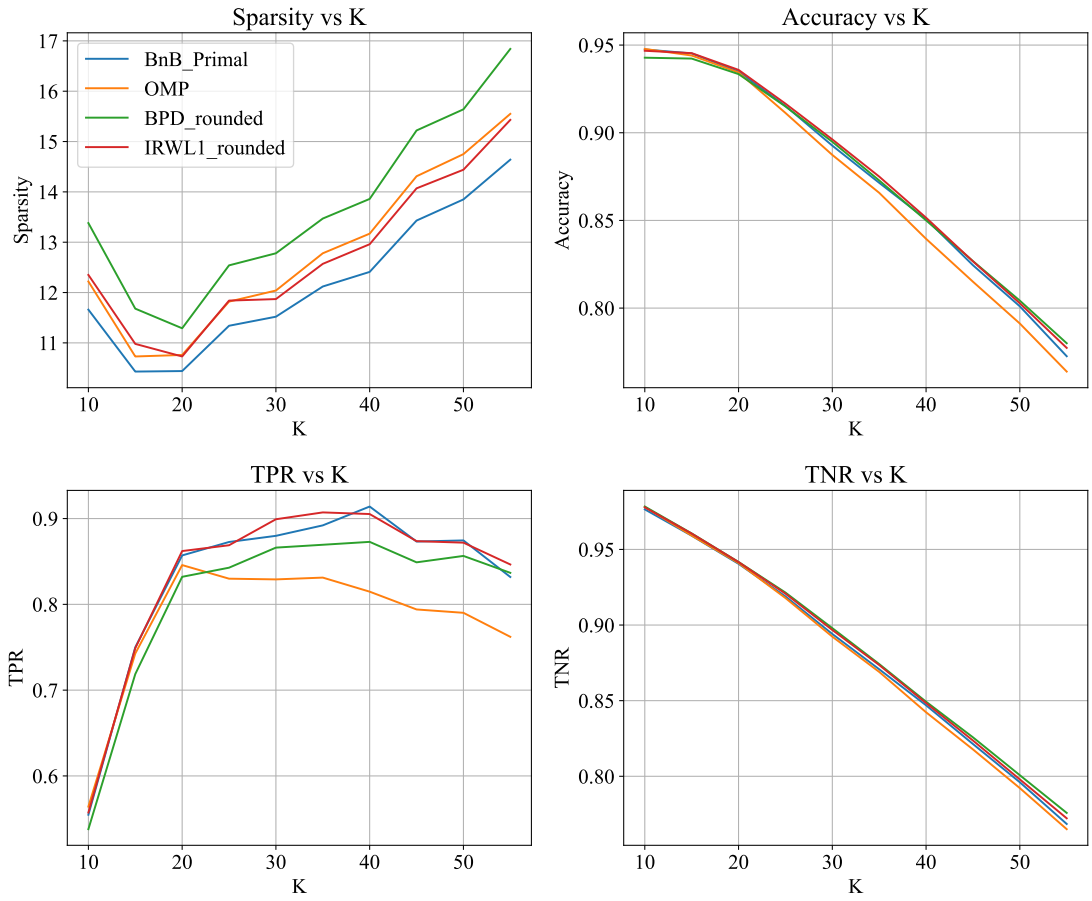

Figure 3.2: Sparsity (top left), accuracy (top right), true positive rate (bottom left) and true negative rate (bottom right) versus k with $N = 200$ and $\alpha = 0.2$. Averaged over 100 trials for each parameter configuration.

Sensitivity to ϵ

We present a comparison of Algorithm 4 with BPD, IRWL1 and OMP as we vary α which controls the value of the parameter ϵ. Recall we have $\epsilon = \alpha\|\boldsymbol{b}\|_2^2$, so α can loosely be interpreted as the fraction of the measurements $\boldsymbol{b}$ that can be unexplained by the returned solution to (3.3). In these experiments, we fixed $n = 200$ and $k = 10$ across all trials. We varied $\alpha \in \{0.05, 0.1, 0.15, \ldots, 0.9\}$ and we performed 100 trials for each value of α. We give Algorithm 4 a cutoff time of 10 minutes.

We report the sparsity, accuracy (ACC), true positive rate (TPR) and true negative rate (TNR) for each method in Figure 3.3, and we report the sparsity, accuracy and execution time

Table 3.5: Comparison of the accuracy of solutions returned by (4), OMP, IRWL1 and BPD for different values of k. Averaged over 100 trials for each parameter configuration.

	Accuracy			
K	Algorithm 4	OMP	IRWL1	BPD
10	**0.948**	**0.948**	0.947	0.943
15	**0.945**	0.944	**0.945**	0.942
20	0.935	0.934	**0.936**	0.933
25	0.915	0.911	**0.917**	0.915
30	0.893	0.887	**0.896**	0.895
35	0.872	0.866	**0.875**	0.873
40	0.851	0.840	**0.852**	0.850
45	0.825	0.815	0.827	**0.827**
50	0.801	0.791	0.803	**0.804**
55	0.772	0.764	0.777	**0.780**

for each method in Tables 3.7, 3.8 and 3.9. Consistent with previous experiments, Algorithm 4 outperforms the benchmark methods in terms of sparsity of the returned solution while having comparable performance on accuracy, TPR and TNR. Here, Algorithm 4 on average produces solutions that are 2.40% more sparse than OMP, 5.92% more sparse than BPD and 2.54% more sparse than IRWL1. Before further sparsifying the BPD (respectively IRWL1) solution, the solution returned by Algorithm 4 is on average 59.23% (respectively 2.62%) more sparse than the BPD (respectively IRWL1) solution.

Lower Bound Performance

In Section 3.4, we reformulated (3.3) exactly as a mixed-integer second order cone problem and illustrated multiple approaches to obtain lower bounds on the optimal value of the reformulation. In this Section, we compare the strength of the second order cone relaxation given by (3.13) and the semidefinite cone relaxation given by (3.21). We fixed $k = 10$ and we varied $\alpha \in \{0.05, 0.1, 0.15, \ldots, 0.9\}$. We report results for $(n, m) = (25, 100)$ in Figure 3.4 and $(n, m) = (50, 25)$ in Figure 3.5. We performed 100 trials for each value of α. Letting lb^{SOC} denote the optimal value of (3.13) and lb^{SOS} denote the optimal value of (3.21), we define the SOS lower bound improvement to be $\frac{lb^{SOS} - lb^{SOC}}{lb^{SOC}}$.

Table 3.6: Comparison of the execution time of solutions returned by (4), OMP, IRWL1 and BPD for different values of k. Averaged over 100 trials for each parameter configuration.

	Execution Time (milliseconds)			
K	Algorithm 4	OMP	IRWL1	BPD
10	305993.475	**13.182**	1270.000	341.454
15	199128.374	**12.556**	1144.818	284.071
20	119282.667	**12.646**	1080.535	278.596
25	139224.525	**13.263**	1081.202	327.151
30	171844.485	**12.909**	1169.798	314.192
35	193257.535	**12.798**	1163.121	361.485
40	231721.737	**13.404**	1151.455	277.647
45	314269.394	**13.495**	1142.374	308.919
50	351790.071	**13.727**	1219.707	315.081
55	412429.717	**14.010**	1260.899	289.616

Consistent with the Theorem 30, Problem (3.21) produces a stronger lower bound than Problem (3.13) at the expense of being more computationally intensive to compute due to the presence of positive semidefinite constraints. On average, the bound produced by (3.21) is 8.92% greater than the bound produced by (3.13). These results suggests that if Problem (3.21) can be solved to optimality or near optimality efficiently at scale, it could potentially be used to accelerate Algorithm 4 by producing stronger lower bounds than the current approach, thereby allowing for a more aggressive pruning of the feasible space. Off the shelf interior point methods suffer from scalability challenges for semidefinite optimization problems.

3.6.2 Electrocardiogram Signal Acquisition

We seek to answer the following question: how does the performance of Algorithm 4 compare to state-of-the-art methods such as BPD, IRWL1 and OMP on signal processing using real world data? To evaluate performance, we consider the problem of compressed sensing for electrocardiogram (ECG) acquisition [52]. We obtain real ECG recording samples from the MIT-BIH Arrhythmia Database (https://www.physionet.org/content/mitdb/1.0.

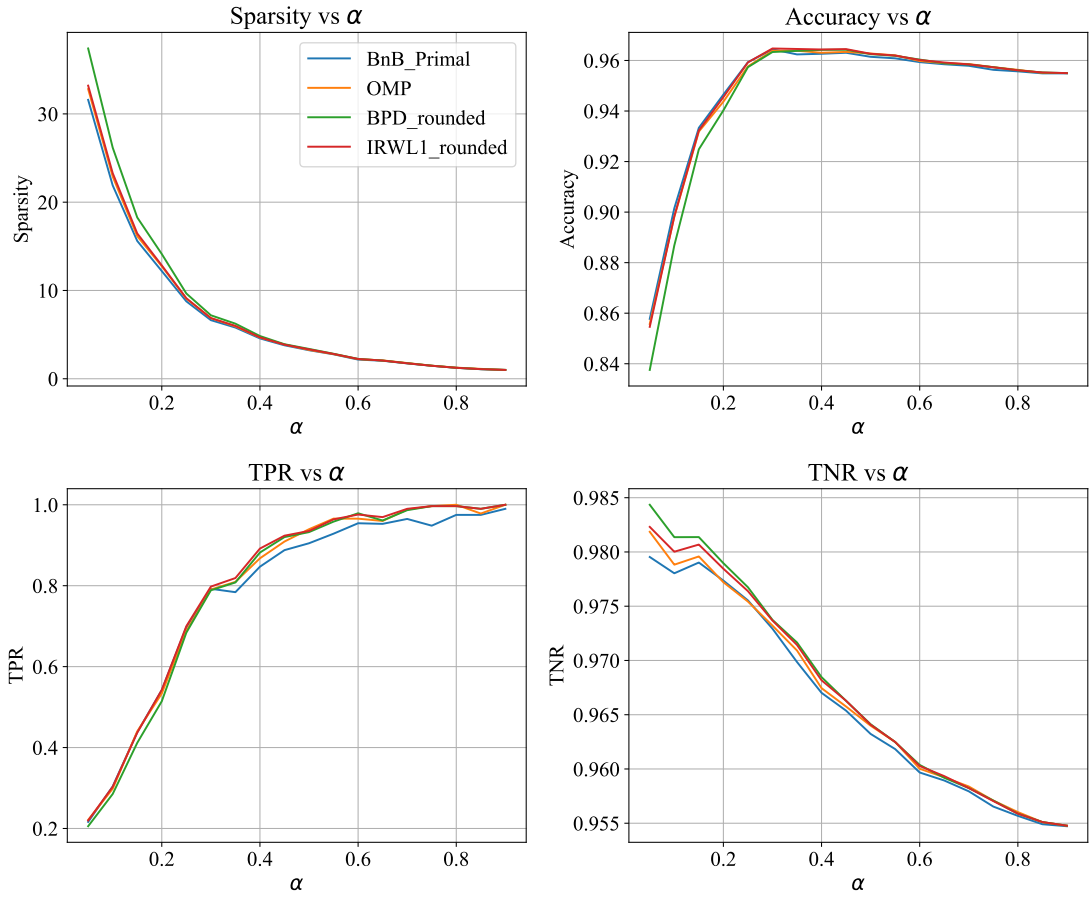

Figure 3.3: Sparsity (top left), accuracy (top right), true positive rate (bottom left) and true negative rate (bottom right) versus α with $n = 200$ and $k = 10$. Averaged over 100 trials for each parameter configuration.

0/) and consider the performance of the methods in terms of sparsity of the returned signal and reconstruction error between the returned signal and the true signal.

ECG Experiment Setup

We employ the same 100 ECG recordings sampled at 360 Hz from the MIT-BIH Arrhythmia Database that are used in [52]. These recordings collectively originate from 10 distinct patients (each contributing 10 independent recordings) and the recording length of an individual record is 1024. In keeping with [52], we use 30 ECG recordings as a training set to fit an overcomplete dictionary $\boldsymbol{D}$ via the K-SVD method [1]. We fit a dictionary with 2000 atoms, meaning that $\boldsymbol{D} \in \mathbb{R}^{1024\times 2000}$ and $\boldsymbol{X}^{train} \approx \boldsymbol{D\Theta}$ where $\boldsymbol{X}^{train} \in \mathbb{R}^{1024\times 30}$ is a matrix

Table 3.7: Comparison of the sparsity of solutions returned by (4), OMP, IRWL1 and BPD for different values of α. Averaged over 100 trials for each parameter configuration.

	Sparsity			
α	Algorithm 4	OMP	IRWL1	BPD
0.05	**31.6**	32.8	33.2	37.4
0.10	**21.9**	22.9	23.3	26.2
0.15	**15.6**	16.1	16.5	18.3
0.20	**12.2**	12.7	12.8	14.1
0.25	**8.8**	9.1	9.1	9.7
0.30	**6.6**	6.9	6.9	7.2
0.35	**5.8**	5.9	6.0	6.2
0.40	**4.6**	4.7	4.7	4.8
0.45	**3.8**	3.9	3.9	3.9
0.50	**3.2**	3.3	3.3	3.4
0.55	**2.8**	2.8	2.8	2.8
0.60	**2.2**	2.2	2.2	2.2
0.65	**2.0**	2.1	2.1	2.1
0.70	**1.7**	1.8	1.8	1.8
0.75	**1.5**	1.5	**1.5**	1.5
0.80	**1.2**	1.3	**1.2**	**1.2**
0.85	**1.1**	1.1	1.1	1.1
0.90	**1.0**	**1.0**	**1.0**	**1.0**

whose columns are the training ECG signals and $\boldsymbol{\Theta} \in \mathbb{R}^{2000\times 30}$ is a sparse matrix. Each column of $\boldsymbol{\Theta}$ should be thought of as a (sparse) representation of the corresponding column of $\boldsymbol{X}^{train}$ in the dictionary given by $\boldsymbol{D}$ ($\|\boldsymbol{\Theta}\|_0 \ll \|\boldsymbol{X}^{train}\|_0$). We employ the Bernouilli sensing matrix $\boldsymbol{B} \in \mathbb{R}^{40\times 1024}$ considered by [52]. Given an ECG signal $\boldsymbol{x}^{test} \in \mathbb{R}^{1024}$, we consider the perturbed observations $\boldsymbol{s} = \boldsymbol{B}(\boldsymbol{x}^{test} + \boldsymbol{\eta})$ where $\boldsymbol{\eta} \in \mathbb{R}^{1024}$ is a vector of mean 0 normal perturbations with variance $\left(\frac{\|\boldsymbol{x}^{test}\|_1}{4\cdot 1024}\right)^2 \boldsymbol{I}$. Figure 3.6 illustrates the ECG signal and perturbed ECG signal for record 31 of the dataset. With these preliminaries, we consider the reconstruction problem given by

$$\begin{aligned} \min_{\boldsymbol{\theta}\in\mathbb{R}^{2000}} \quad & \|\boldsymbol{\theta}\|_0 + \frac{1}{\gamma}\|\boldsymbol{\theta}\|_2^2 \\ \text{s.t.} \quad & \|\boldsymbol{BD\theta} - \boldsymbol{s}\|_2^2 \leq \epsilon. \end{aligned} \tag{3.28}$$

Table 3.8: Comparison of the accuracy of solutions returned by (4), OMP, IRWL1 and BPD for different values of α. Averaged over 100 trials for each parameter configuration.

	Accuracy			
α	Algorithm 4	OMP	IRWL1	BPD
0.05	**0.858**	0.856	0.855	0.838
0.10	**0.901**	0.898	0.898	0.887
0.15	**0.933**	0.932	0.932	0.925
0.20	**0.946**	0.944	0.946	0.940
0.25	**0.959**	0.958	0.959	0.957
0.30	0.964	0.964	**0.965**	0.963
0.35	0.962	0.964	**0.965**	0.964
0.40	0.963	0.963	**0.964**	0.964
0.45	0.963	0.963	**0.965**	0.964
0.50	0.962	**0.963**	0.963	0.963
0.55	0.961	**0.962**	0.962	0.962
0.60	0.959	0.960	0.960	**0.960**
0.65	0.959	0.959	**0.959**	0.959
0.70	0.958	**0.959**	0.958	0.958
0.75	0.956	**0.957**	0.957	0.957
0.80	0.956	**0.956**	0.956	0.956
0.85	0.955	0.955	**0.955**	**0.955**
0.90	0.955	**0.955**	**0.955**	**0.955**

where we set $\epsilon = 1.05 \cdot \|\boldsymbol{s} - \boldsymbol{B}\boldsymbol{x}^{test}\|_2^2$. Note that (3.28) is equivalent to (3.3) where $(\boldsymbol{\theta}, \boldsymbol{B}\boldsymbol{D}, \boldsymbol{s})$ play the role of $(\boldsymbol{x}, \boldsymbol{A}, \boldsymbol{b})$ and we have $(n, m) = (2000, 40)$. Letting $\hat{\boldsymbol{\theta}}$ denote a feasible solution to (3.28) returned by one of the solution methods, we employ 10^{-4} as the numerical threshold to compute the sparsity $\|\hat{\boldsymbol{\theta}}\|_0$ of $\hat{\boldsymbol{\theta}}$ and we define the ℓ_q reconstruction error of $\hat{\boldsymbol{\theta}}$ as $\frac{\|\boldsymbol{D}\hat{\boldsymbol{\theta}} - \boldsymbol{x}^{test}\|_q^q}{\|\boldsymbol{x}^{test}\|_q^q}$ for $q \in \{1, 2\}$.

ECG Computational Results

We present a comparison of BPD, IRWL1, OMP and Algorithm 4 as we vary the regularization parameter γ in (3.28). We considered values of γ in the set

$$\Gamma = \{(8a + 0.01)f(n) : a \in [14], f(n) \in \{\sqrt{n}, n, n^2\}\},$$

Table 3.9: Comparison of the execution time of solutions returned by (4), OMP, IRWL1 and BPD for different values of α. Averaged over 100 trials for each parameter configuration.

	Execution Time (milliseconds)			
α	Algorithm 4	OMP	IRWL1	BPD
0.05	603205.808	**22.798**	5164.919	934.717
0.10	577124.980	**18.061**	2158.465	522.212
0.15	454366.242	**15.535**	1801.596	636.172
0.20	343487.778	**14.636**	2013.323	967.657
0.25	181672.232	**13.970**	1477.374	654.091
0.30	81074.212	**14.253**	1087.737	510.162
0.35	61929.838	**13.475**	1503.990	904.788
0.40	41775.950	**13.838**	1213.343	684.101
0.45	15927.495	**18.939**	1056.717	531.091
0.50	10387.101	**13.687**	1384.343	872.040
0.55	7032.818	**18.091**	1001.253	530.556
0.60	858.253	**13.212**	1118.929	640.242
0.65	1012.121	**18.404**	1280.869	926.727
0.70	494.808	**18.333**	783.697	514.081
0.75	499.677	**19.899**	696.495	511.717
0.80	749.626	**20.283**	965.010	910.485
0.85	479.091	**20.606**	539.768	537.869
0.90	462.808	**13.394**	485.869	514.566

and we evaluate performance on the 70 ECG recordings that are not part of the training set used to fit the overcomplete dictionary $\boldsymbol{D}$. We give Algorithm 4 a cutoff time of 5 minutes. As in the synthetic experiments, we terminate IRWL1 after the 50^{th} iteration or after two subsequent iterates are equal up to numerical tolerance. Moreover, we sparsify the solutions returned by BPD and IRWL1 using the procedure given by Algorithm 3 as done in the synthetic experiments.

Figure 3.7 illustrates the average ℓ_1 error (left) and average ℓ_2 error (right) versus the average sparsity of solutions returned by each method. Each red dot corresponds to the performance of Algorithm 4 for a fixed value of $\gamma \in \Gamma$. Given that more sparse solutions and solutions with lesser ℓ_1 (respectively ℓ_2) error are desirable, Figure 3.7 demonstrates that as we vary γ, the solutions returned by Algorithm 4 trace out an efficient frontier that dominates the solutions returned by BPD, IRWL1 and OMP. Indeed, for all benchmark methods (BPD,

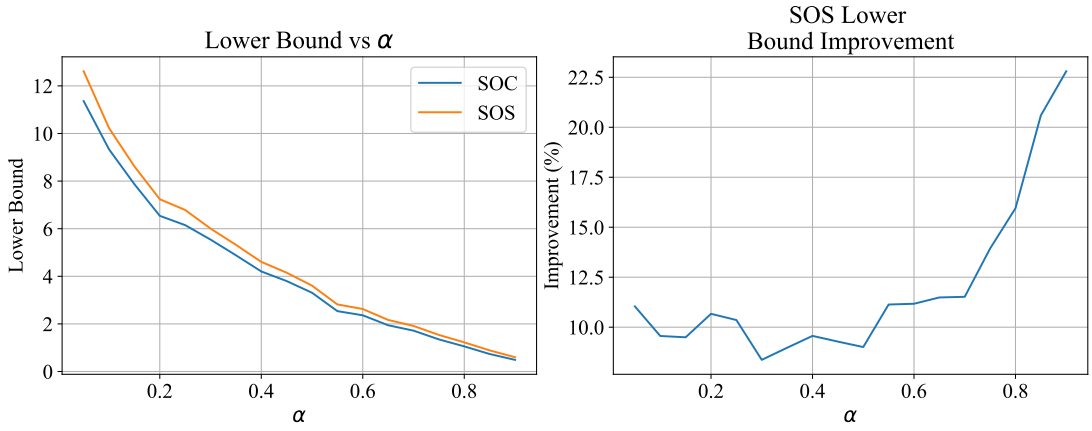

Figure 3.4: Problem (3.3) lower bound (left) produced by Problem (3.13) (SOC) and Problem (3.21) (SOS) with $d = 1$. Percent improvement of Problem (3.3) lower bound of compared to (right). $n = 25, m = 100$ and $k = 10$.

IRWL1 and OMP), there is a value of γ such that Algorithm 4 finds solutions that achieve lower sparsity and lower reconstruction error than the solution returned by the benchmark method. For the same ℓ_2 reconstruction error, Algorithm 4 can produce solutions that are on average 3.88% more sparse than IRWL1, 6.29% more sparse than BPD and 19.70% more sparse than OMP. For the same sparsity level, Algorithm 4 can produce solutions that have on average 1.42% lower ℓ_2 error than IRWL1, 2.66% lower ℓ_2 error than BPD and 28.23% lower ℓ_2 error than OMP. Thus, Algorithm 4 outperforms BPD, IRWL1 and OMP on this real world dataset.

3.6.3 Multi-Label Classification

We seek to answer the following question: how does the performance of Algorithm 4 compare to state-of-the-art methods such as BPD, IRWL1 and OMP when used as part of a learning task on real world data? To evaluate the performance of the aforementioned algorithms in this setting, we consider the problem of multi-label classification (MLC) [82]. In MLC, given a dataset $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ where $\boldsymbol{x}_i \in \mathbb{R}^m$ denotes a feature vector and $\boldsymbol{y}_i \in \{0,1\}^d$ denotes a label vector, we seek to learn a function $f : \mathbb{R}^m \to \{0,1\}^d$ that can correctly classify unseen

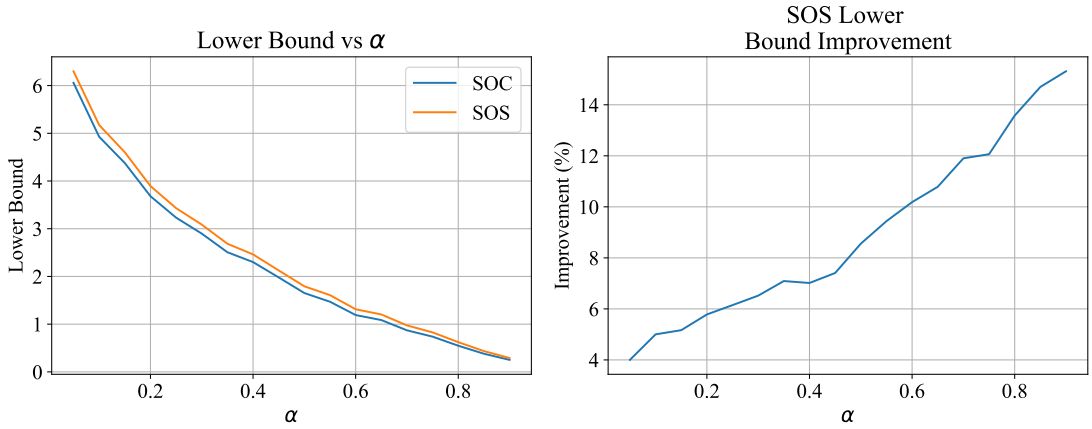

Figure 3.5: Problem (3.3) lower bound (left) produced by Problem (3.13) (SOC) and Problem (3.21) (SOS) with $d = 1$. Percent improvement of Problem (3.3) lower bound of compared to (right). $n = 50, m = 25$ and $k = 10$.

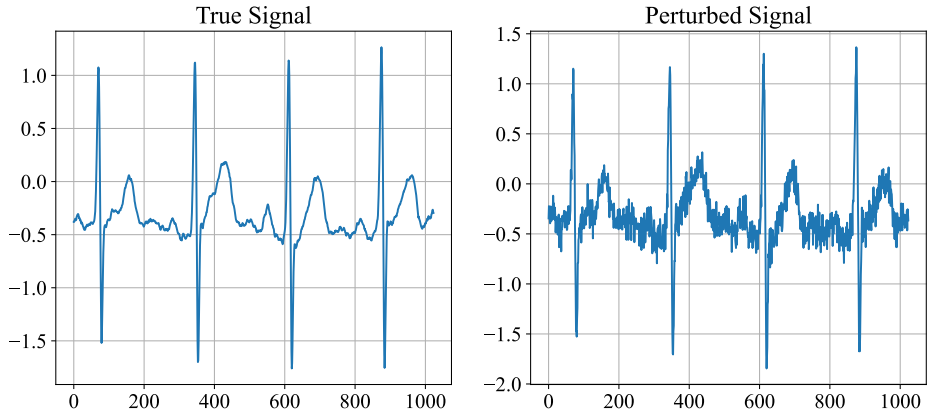

Figure 3.6: Ground truth ECG signal (left) and perturbed signal (right) for ECG record 31.

examples. Note that this is a more general problem than a typical multi-class classification problem with d classes since in this setting each example can be a member of multiple classes simultaneously. When the number of classes d is very large, training d individual classifiers in a one-against-all approach becomes prohibitively expensive. In this regime, if the label vectors tend to be sparse relative to the size of the latent dimension d, one approach is to project the labels into a smaller dimension $k \ll d$ using a linear compression function and then to learn k predictors mapping the feature space to the projected label space. Predictions can then be made by applying the k learned predictors on a given example and

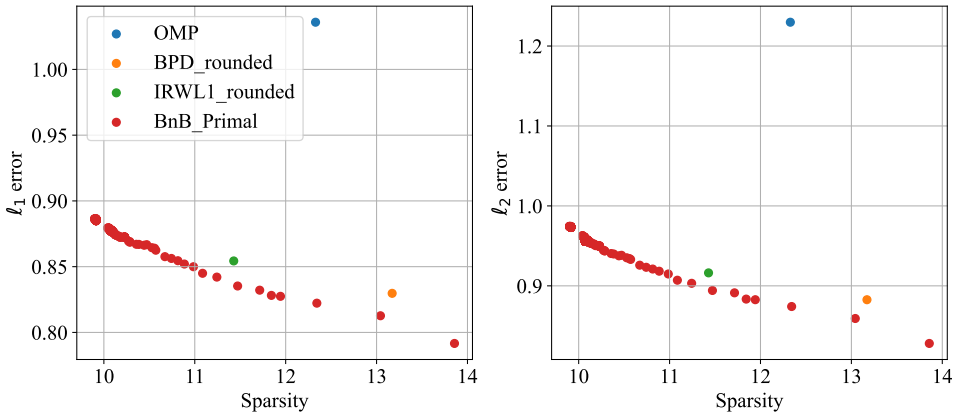

Figure 3.7: ℓ_1 reconstruction error (left) and ℓ_2 reconstruction error (right) versus ℓ_0 norm (Sparsity) for ECG reconstructions obtained using OMP, BPD, IRWL1 and Algorithm 4 for varying values of γ. $n = 2000$ and $m = 40$.

subsequently using a compressed sensing based reconstruction algorithm to transform the prediction from projected label space to the latent label space [82, 87, 88, 129]. Explicitly, given a linear map $\boldsymbol{P} \in \mathbb{R}^{k \times d}$ where $k \ll d$, we form a transformed dataset $\{(\boldsymbol{x}_i, \boldsymbol{P}\boldsymbol{y}_i)\}_{i=1}^n$ and for each $j \in \{1, \ldots, k\}$ learn a regression function $h_j : \mathbb{R}^m \to \mathbb{R}$ using the data given by $\{(\boldsymbol{x}_i, [\boldsymbol{P}\boldsymbol{y}_i]_j)\}_{i=1}^n$. Given a reconstruction function $g : \mathbb{R}^{k \times d} \times \mathbb{R}^k \to \mathbb{R}^d$ that outputs a sparse vector such that $\boldsymbol{P}[g(\boldsymbol{P}, \boldsymbol{l})] \approx \boldsymbol{l}$, we make predictions on a test point $\boldsymbol{x}_{test}$ by returning $g(\boldsymbol{P}, [h_1(\boldsymbol{x}_{test}), \ldots, h_k(\boldsymbol{x}_{test})])$. In Sections 3.6.3 and 3.6.3, we evaluate the performance of this approach when Algorithm 4, BPD, IRWL1 and OMP are used as the reconstruction function g.

MLC Experiment Setup

We obtain a text data set consisting of web pages and descriptive textual tags from the former social bookmarking service *del.icio.us* that was originally collected by [137]. The dataset consists of roughly 16000 web pages having $d = 983$ unique labels. Each web page on average is associated with 19 labels. The web pages are each represented as a bag-of-words feature vector. Further details can be found in [137]. We use 80% of the data for training and withhold 20% for testing. We fix $k = 100$ during our experiments. We fix the

linear map $\boldsymbol{P} = \frac{1}{\sqrt{k}}\boldsymbol{H}$ where $\boldsymbol{H} \in \mathbb{R}^{k\times d}$ is a matrix obtained by choosing k random rows from the $d \times d$ Hadamard matrix in keeping with the approach taken by [82]. We use ridge regression as our base learning algorithm for the regression functions $\{h_j\}_{j=1}^k$ and tune the ridge parameter via leave one out cross validation over the training set. Given a test data point $\boldsymbol{x}_{test} \in \mathbb{R}^m$, we output a predicted label $\hat{\boldsymbol{y}} \in \mathbb{R}^d$ by solving

$$\min_{\boldsymbol{y}\in\mathbb{R}^d} \|\boldsymbol{y}\|_0 + \frac{5}{\sqrt{d}}\|\boldsymbol{y}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{P}\boldsymbol{y} - h(\boldsymbol{x}_{test})\|_2^2 \leq \epsilon. \tag{3.29}$$

where $[h(\boldsymbol{x}_{test})]_j = h_j(\boldsymbol{x}_{test})$ and we set $\epsilon = 0.25 \cdot \|h(\boldsymbol{x}_{test})\|_2^2$. Letting $\hat{\boldsymbol{y}}$ denote a feasible solution to (3.29) returned by one of the solution methods, we are interested in measuring the accuracy, precision and recall of the support of $\hat{\boldsymbol{y}}$ relative to the true label $\boldsymbol{y}_{test}$. Let $\mathcal{I}^{true} = \{i : [\boldsymbol{y}_{test}]_i = 1\}$ and $\hat{\mathcal{I}} = \{i : |\hat{y}_i| > 10^{-4}\}$. We define the accuracy of $\hat{\boldsymbol{y}}$ as

$$ACC(\hat{\boldsymbol{y}}) = \frac{\sum_{i\in\mathcal{I}^{true}} \mathbb{1}\{|\hat{y}_i| > 10^{-4}\} + \sum_{i\notin\mathcal{I}^{true}} \mathbb{1}\{|\hat{y}_i| \leq 10^{-4}\}}{d}.$$

Similarly, we define the precision of $\hat{\boldsymbol{y}}$ as

$$Precision(\hat{\boldsymbol{y}}) = \frac{\sum_{i\in\mathcal{I}^{true}} \mathbb{1}\{|\hat{y}_i| > 10^{-4}\}}{|\mathcal{I}^{true}|},$$

and we define the recall (true positive rate) of $\hat{\boldsymbol{y}}$ as

$$TPR(\hat{\boldsymbol{y}}) = \frac{\sum_{i\in\mathcal{I}^{true}} \mathbb{1}\{|\hat{y}_i| > 10^{-4}\}}{|\hat{\mathcal{I}}|}.$$

MLC Computational Results

We present a comparison of test set classification performance when each of BPD, IRWL1, OMP and Algorithm 4 are used as the reconstruction algorithm g. We give Algorithm 4 a cutoff time of 10 minutes. As in the synthetic and ECG experiments, we terminate IRWL1 after the 50^{th} iteration or after two subsequent iterates are equal up to numerical

tolerance. Moreover, we sparsify the solutions returned by BPD and IRWL1 using the procedure given by Algorithm 3 as done in the synthetic and ECG experiments. Figure 3.8 illustrates the average accuracy achieved by each method on the test set. We see that that Algorithm 2 achieves slightly greater average accuracy than the benchmark methods. Moreover, Table 3.10 illustrates the percentage of test examples on which each method has the best performance for accuracy, precision and recall. We see that Algorithm 2 exhibits superior performance than the baseline methods across all metrics of interest.

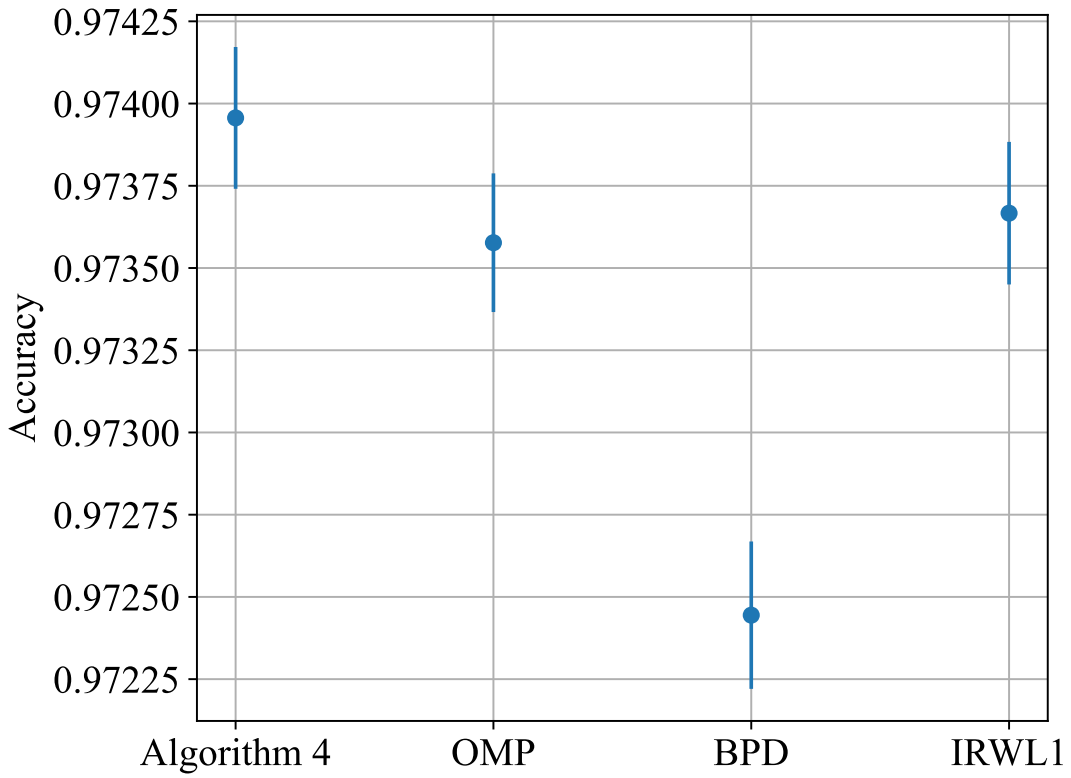

Figure 3.8: Test set prediction accuracy obtained using OMP, BPD, IRWL1 and Algorithm 4 as the label reconstruction algorithm.

3.6.4 Summary of Findings

We now summarize our findings from our numerical experiments. In Sections 3.6.1-3.6.1, we see that across all experiments using synthetic data, Algorithm 4 produces solutions that

Table 3.10: Comparison of the accuracy of solutions returned by (4), OMP, IRWL1 and BPD for different values of α. Averaged over 100 trials for each parameter configuration.

	Top Performing Algorithm Frequency (%)			
Metric	Algorithm 4	OMP	IRWL1	BPD
Accuracy	**53.47**	28.78	11.66	6.09
Precision	**49.39**	21.42	15.26	13.93
Recall	**54.90**	14.14	3.20	27.76

are on average 6.22% more sparse than the solutions returned by state of the art benchmark methods after they are further sparsified by greedy rounding. If we omit greedy rounding, Algorithm 4 produces solutions that are on average 17.17% more sparse in our synthetic experiments. In Section 3.6.1, we find that the bound produced by (3.21) is on average 8.92% greater than the bound produced by (3.13). In Section 3.6.2, we see that for a given level of ℓ_2 reconstruction error, Algorithm 4 produces solutions that are on average 9.95% more sparse than the solutions returned by state of the art benchmark methods after they are further sparsified by greedy rounding on the real world ECG dataset we experiment with. Furthermore, for a given sparsity level, Algorithm 4 produces solutions that have on average 10.77% lower ℓ_2 reconstruction error than benchmark methods. Finally, in Section 3.6.3, we see that Algorithm 4 outperforms benchmark methods when used as the reconstruction algorithm for compressed sensing multi-label classification in terms of accuracy, precision and recall.

3.7 Concluding Remarks

In this chapter, we introduced an ℓ_2 regularized formulation (3.3) for CS which emits a natural reformulation as a mixed-integer second order cone program (3.12). We presented a second order cone relaxation (3.13) and a stronger but more expensive semidefinite cone relaxation (3.21) to (3.12). We presented Algorithm 4, a custom branch and bound algorithm that can compute globally optimal solution for (3.3). We find that our approach produces

solutions that are on average 6.22% more sparse on synthetic data and 9.95% more sparse on real world ECG data when compared to three state of the art benchmark approaches. This improvement in accuracy comes at the cost of an increase in computation time by several orders of magnitude. When comparing only against the experiment-wise best performing benchmark method, our approach produces solutions that are on average 3.10% more sparse on synthetic data and 3.88% more sparse on real world ECG data. Moreover, our approach outperforms benchmark methods when used as part of a multi-label learning algorithm. Further work might focus on strengthening our convex relaxations by deriving additional valid inequalities for (3.12) or increasing the scalability of our branch and bound method. Algorithm 4 currently uses our second order cone relaxation to compute lower bounds. If fast problem specific solution methods could be derived for our positive semidefinite cone relaxation, employing the latter for lower bounds in Algorithm 4 could potentially lead to important scalability gains.

Chapter 4

Predictive Low Rank Matrix Learning under Partial Observations: Mixed-Projection ADMM

The work in this chapter is based on the preprint [25] which is joint work with Dimitris Bertsimas.

4.1 Introduction

In many real world applications, we are faced with the problem of recovering a (often large) matrix from a (often small) subset of its entries. This problem, known as Matrix Completion (MC), has gained significant attention due to its broad range of applications in areas such as signal processing [45], system identification [99] and image denoising [85]. The fundamental task in MC is to accurately reconstruct the missing entries of a matrix given a limited number of observed entries. The challenge is particularly pronounced when the number of observed entries is small relatively to the dimension of the matrix, yet this is the common scenario in practice.

One of the most prominent uses of MC is in recommendation systems, where the goal is to predict user preferences for items (e.g., movies, products) based on a partially observed user-item rating matrix [119]. The Netflix Prize competition highlighted the potential of MC techniques, where the objective was to predict missing ratings in a user-movie matrix to improve recommendation accuracy [9]. The success of such systems hinges on the assumption that the underlying rating matrix is low rank, meaning that the preferences of users can be well-approximated by a small number of factors. Indeed, it has been well studied that many real world datasets are low rank [138].

In many practical applications, in addition to a collection of observed matrix entries we additionally have access to auxiliary side information that can be leveraged when performing the reconstruction. For example, in a recommendation system, side information might consist of social network data or item attributes. The vast majority of existing approaches to MC in the presence of side information incorporate the side information by making additional structural restrictions on the reconstructed matrix beyond the usual low rank assumption (see, for example, [26, 58, 146]). In this work, we take an alternate approach by assuming that the side information can be well modelled as a linear function of the underlying full matrix. In this setting, the side information can be thought of as labels for a regression

problem where the unobserved matrix consists of the regression features. This assumption is in keeping with ideas from the predictive low rank kernel learning literature [5] (note however that low rank kernel learning assumes a fully observed input matrix).

Formally, let $\Omega \subseteq [n] \times [m]$ denote a collection of revealed entries of a partially observed matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, let $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ denote a matrix of side information and let k denote a specified target rank. We consider the problem given by

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times m}, \boldsymbol{\alpha} \in \mathbb{R}^{m \times d}} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 + \lambda \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\alpha}\|_F^2 + \gamma \|\boldsymbol{X}\|_\star \quad \text{s.t.} \quad \operatorname{rank}(\boldsymbol{X}) \leq k, \tag{4.1}$$

where $\lambda, \gamma > 0$ are hyperparameters that in practice can either take a default value or can be cross-validated by minimizing a validation metric [114] to obtain strong out-of-sample performance [34]. We assume that the ground truth matrix $\boldsymbol{A}$ has low rank and that the side information can be well approximated as $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\alpha} + \boldsymbol{N}$ for some weighting matrix $\boldsymbol{\alpha}$ and noise matrix $\boldsymbol{N}$. The first term in the objective function of (4.1) measures how well the observed entries of the unknown matrix are fit by the estimated matrix $\boldsymbol{X}$, the second term of the objective function measures how well the side information $\boldsymbol{Y}$ can be represented as a linear function of the estimated matrix $\boldsymbol{X}$ and the final term of the objective is a regularization term. To the best of our knowledge, Problem (4.1) has not previously been directly studied despite its very natural motivation.

4.1.1 Contribution and Structure

In this chapter, we tackle (4.1) by developing novel mixed-projection optimization techniques [19]. We show that solving (4.1) is equivalent to solving an appropriately defined robust optimization problem. We develop an exact reformulation of (4.1) by combining a parametrization of the $\boldsymbol{X}$ decision variable as the product of two low rank factors with the introduction of a projection matrix to model the column space of $\boldsymbol{X}$. We derive a semidefinite cone convex relaxation for our mixed-projection reformulation and we present an efficient,

scalable alternating direction method of multipliers (ADMM) algorithm that produces high quality feasible solutions to (4.1). Our numerical results show that across all synthetic data experiments in the small rank regime ($k \leq 15$), our algorithm outputs solutions that achieve on average 79% lower objective value in (4.1) and 90.1% lower ℓ_2 reconstruction error than the solutions returned by the best performing benchmark method on a per experiment basis. For the 5 synthetic data experiments with $k > 15$, the only benchmark that returns a solution with superior quality than that returned by our algorithm takes on average 3 times as long to execute. The runtime of our algorithm is competitive with and often superior to that of the benchmark methods. Our algorithm is able to solve problems with $n = 10000$ rows and $m = 10000$ columns in less than a minute. On large scale real world data, our algorithm produces solutions that achieve 67% lower out of sample error than benchmark methods in 97% less execution time.

The rest of the chapter is laid out as follows. In Section 4.2, we review previous work that is closely related to (4.1). In Section 4.3, we study (4.1) under a robust optimization lens and investigate formulating (4.1) as a two stage optimization problem where the inner stage is a regression problem that can be solved in closed form. We formulate (4.1) as a mixed-projection optimization problem in Section 4.4 and present a natural convex relaxation. In Section 4.5, we present and rigorously study our ADMM algorithm. Finally, in Section 4.6 we investigate the performance of our algorithm against benchmark methods on synthetic and real world data.

4.2 Literature Review

In this section, we review a handful of notable approaches from the literature that have been employed to solve MC and to solve general low rank optimization problems. As an exhaustive literature review of MC methods is outside of the scope of this chapter, we focus our review on a handful of well studied approaches which we will employ as benchmark methods in this

work. We additionally give an overview of the ADMM algorithmic framework which is of central relevance to this work. For a more detailed review of the MC literature, we refer the reader to [119] and [111].

4.2.1 Matrix Completion Methods

Iterative-SVD

Iterative-SVD is an expectation maximization style algorithm [60] that generates a solution to the MC problem by iteratively computing a singular value decomposition (SVD) of the current iterate and estimating the missing values by performing a regression against the low rank factors returned by SVD [136]. This is one of a handful of methods in the literature that leverage the SVD as their primary algorithmic workhorse [32, 126]. Concretely, given a partially observed matrix $\{X_{ij}\}_{(i,j)\in\Omega}$ where $\Omega \subseteq [n] \times [m]$ and a target rank $k \in \mathbf{N}_+$, Iterative-SVD proceeds as follows:

1. Initialize the iteration count $t \leftarrow 0$ and initialize missing entries of $X_{ij}, (i,j) \notin \Omega$ with the row average $X_{ij} = \frac{\sum_{l:(i,l)\in\Omega} X_{il}}{|\{l:(i,l)\in\Omega\}|}$.

2. Compute a rank k SVD $\boldsymbol{X}_t = \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^T$ of the current iterate where $\boldsymbol{U}_t \in \mathbb{R}^{n\times k}, \boldsymbol{\Sigma}_t \in \mathbb{R}^{k\times k}, \boldsymbol{V}_t \in \mathbb{R}^{m\times k}$.

3. For each $(i,j) \notin \Omega$, estimate the missing value $(\boldsymbol{X}_{t+1})_{ij}$ by regressing all other entries in row i against all except the j^{th} row of $\boldsymbol{V}_t$. Concretely, letting $\tilde{\boldsymbol{x}} = (\boldsymbol{X}_t)_{i,\star\setminus j} \in \mathbb{R}^{m-1}$ denote the column vector consisting of the i^{th} row of $\boldsymbol{X}_t$ excluding the j^{th} entry, letting $\tilde{\boldsymbol{V}} = (\boldsymbol{V}_t)_{\star\setminus j,\star} \in \mathbb{R}^{(m-1)\times k}$ denote the matrix formed by eliminating the j^{th} row from $\boldsymbol{V}_t$ and letting $\hat{\boldsymbol{v}} = (\boldsymbol{V}_t)_{j,\star} \in \mathbb{R}^k$ denote the column vector consisting of the j^{th} row of $\boldsymbol{V}_t$, we set $(\boldsymbol{X}_{t+1})_{ij} = \hat{\boldsymbol{v}}^T(\tilde{\boldsymbol{V}}^T\tilde{\boldsymbol{V}})^{-1}\tilde{\boldsymbol{V}}^T\tilde{\boldsymbol{x}}$.

4. Terminate if the total change between $\boldsymbol{X}_t$ and $\boldsymbol{X}_{t+1}$ is less than 0.01. Otherwise, increment t and return to Step 2.

Soft-Impute

Soft-Impute is a convex relaxation inspired algorithm that leverages the nuclear norm as a low rank inducing regularizer [105]. This approach is one of a broad class of methods that tackle MC from a nuclear norm minimization lens [43, 48, 66]. Seeking a reconstruction with minimum nuclear norm is typically motivated by the observation that the nuclear norm ball given by $\mathcal{B} = \{\boldsymbol{X} \in \mathbb{R}^{n \times n} : \|\boldsymbol{X}\|_\star \leq k\}$ is the convex hull of the non-convex set $\mathcal{X} = \{\boldsymbol{X} \in \mathbb{R}^{n \times n} : \text{rank}(\boldsymbol{X}) \leq k, \|\boldsymbol{X}\|_\sigma \leq 1\}$, where $\|\cdot\|_\sigma$ denotes the spectral norm. Moreover, several conditions have been established under which nuclear norm minimization methods are guaranteed to return the ground truth matrix [43, 48] though such conditions tend to be strong and hard to verify in practice. Soft-Impute proceeds by iteratively replacing the missing elements of the matrix with those obtained from a soft thresholded low rank singular value decomposition. Accordingly, similarly to Iterative-SVD, Soft-Impute relies on the computation of a low rank SVD as the primary algorithmic workhorse. The approach relies on the result that for an arbitrary matrix $\boldsymbol{X}$, the solution of the problem $\min_{\boldsymbol{Z}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Z}\|_F^2 + \lambda\|\boldsymbol{Z}\|_\star$ is given by $\hat{\boldsymbol{Z}} = S_\lambda(\boldsymbol{X})$ where $S_\lambda(\cdot)$ denotes the soft-thresholding operation [64]. Explicitly, Soft-Impute proceeds as follows for a given regularization parameter $\lambda > 0$ and termination threshold $\epsilon > 0$:

1. Initialize the iteration count $t \leftarrow 0$ and initialize $\boldsymbol{Z}_t = \boldsymbol{0}_{n \times m}$.
2. Compute $\boldsymbol{Z}_{t+1} = S_\lambda(P_\Omega(\boldsymbol{X}) + P_\Omega^\perp(\boldsymbol{Z}_t))$ where $P_\Omega(\cdot)$ denotes the operation that projects onto the revealed entries of $\boldsymbol{X}$ while $P_\Omega^\perp(\cdot)$ denotes the operation that projects onto the missing entries of $\boldsymbol{X}$.
3. Terminate if $\frac{\|\boldsymbol{Z}_t - \boldsymbol{Z}_{t+1}\|_F^2}{\|\boldsymbol{Z}_t\|_F^2}$. Otherwise, increment t and return to Step 2.

Fast-Impute

Fast-Impute is a projected gradient descent approach to MC that has desirable global convergence properties [26]. Fast-Impute belongs to the broad class of methods that solve MC

by factorizing the target matrix as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ where $\boldsymbol{U} \in \mathbb{R}^{n\times k}, \boldsymbol{V} \in \mathbb{R}^{m\times k}$ and performing some variant of gradient descent (or alternating minimization) on the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ [83, 86, 90, 153]. We note that we leverage this common factorization in the approach to (4.1) presented in this work. Gradient descent based methods have shown great success. Despite the non-convexity of the factorization, it has been shown that in many cases gradient descent and its variants will nevertheless converge to a globally optimal solution [26, 56, 68, 102, 130]. Fast-Impute takes the approach of expressing $\boldsymbol{U}$ as a closed form function of $\boldsymbol{V}$ after performing the facorization and directly performs projected gradient descent updates on $\boldsymbol{V}$ with classic Nesterov acceleration [109]. Moreover, to enhance scalability of their method, [26] design a stochastic gradient extension of Fast-Impute that estimates the gradient at each update step by only considering a sub sample of the rows and columns of the target matrix.

4.2.2 Low Rank Optimization Methods

ScaledGD

ScaledGD is a highly performant method to obtain strong solutions to low rank matrix estimation problems that take the following form:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times m}} f(\boldsymbol{X}) = \frac{1}{2}\|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{y}\|_2^2 \quad \text{s.t. } \operatorname{rank}(\boldsymbol{X}) \leq k,$$

where $\mathcal{A}(\cdot) : \mathbb{R}^{n\times m} \to \mathbb{R}^l$ models some measurement process and we have $\boldsymbol{y} \in \mathbb{R}^l$ [133]. ScaledGD proceeds by factorizing the target matrix as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ and iteratively performing gradient updates on the low rank factors $\boldsymbol{U}, \boldsymbol{V}$ after preconditioning the gradients with an adaptive matrix that is efficient to compute. Doing so yields a linear convergence rate that is notably independent of the condition number of the low rank matrix. In so doing, ScaledGD combines the desirable convergence rate of alternating minimization with the desirable low per-iteration cost of gradient descent. Explicitly, letting $\mathcal{L}(\boldsymbol{U}, \boldsymbol{V}) = f(\boldsymbol{U}\boldsymbol{V}^T)$, ScaledGD

updates the low rank factors as:

$$\boldsymbol{U}_{t+1} \leftarrow \boldsymbol{U}_t - \eta \nabla_{\boldsymbol{U}} \mathcal{L}(\boldsymbol{U}_t, \boldsymbol{V}_t)(\boldsymbol{V}^T\boldsymbol{V})^{-1},$$

$$\boldsymbol{V}_{t+1} \leftarrow \boldsymbol{V}_t - \eta \nabla_{\boldsymbol{V}} \mathcal{L}(\boldsymbol{U}_t, \boldsymbol{V}_t)(\boldsymbol{U}^T\boldsymbol{U})^{-1},$$

where $\eta > 0$ denotes the step size.

Mixed-Projection Conic Optimization

Mixed-projection conic optimization is a recently proposed modelling and algorithmic framework designed to tackle a broad class of matrix optimization problems [19, 20]. Specifically, this approach considers problems that have the following form:

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times m}} \langle \boldsymbol{C}, \boldsymbol{X} \rangle + \lambda \cdot \operatorname{rank}(\boldsymbol{X}) + \Omega(\boldsymbol{X}) \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{X} = \boldsymbol{B},\ \operatorname{rank}(\boldsymbol{X}) \leq k,\ \boldsymbol{X} \in \mathcal{K}, \tag{4.2}$$

where $\boldsymbol{C} \in \mathbb{R}^{n \times m}$ is a cost matrix, $\lambda > 0$, $k \in \mathbb{N}_+$, $\boldsymbol{A} \in \mathbb{R}^{l \times n}, \boldsymbol{B} \in \mathbb{R}^{l \times m}, \mathcal{K}$ denotes a proper cone in the sense of [35] and $\Omega(\cdot)$ is a Frobenius norm regularization function or a spectral norm regularization function of the input matrix. The main workhorse of mixed-projection conic optimization is the use of a projection matrix to cleverly model the rank terms in (4.2). This can be viewed as the matrix generalization of using binary variables to model the sparsity of a vector in mixed-integer optimization. [19] show that for an arbitrary matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, we have

$$\operatorname{rank}(\boldsymbol{X}) \leq k \iff \exists \boldsymbol{P} \in \mathbb{S}^n : \boldsymbol{P}^2 = \boldsymbol{P}, \operatorname{Tr}(\boldsymbol{P}) \leq k, \boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}.$$

Introducing projection matrices allows the rank functions to be eliminated from (4.2) at the expense of introducing non-convex quadratic equality constraints. From here, most existing works that leverage mixed-projection conic optimization have either focused on obtaining

strong semidefinite based convex relaxations [19, 20] or have focused on obtaining certifiably optimal solutions for small and moderately sized problem instances [15, 16]. In this work, we leverage the mixed-projection framework to scalably obtain high quality solutions to large problem instances.

4.2.3 Alternating Direction Method of Multipliers

Alternating direction method of multipliers (ADMM) is an algorithm that was originally designed to solve linearly constrained convex optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n,\boldsymbol{z}\in\mathbb{R}^m} \quad f(\boldsymbol{x})+g(\boldsymbol{z}) \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}=\boldsymbol{c}, \tag{4.3}$$

where we have $\boldsymbol{A}\in\mathbb{R}^{l\times n}, \boldsymbol{B}\in\mathbb{R}^{l\times m}, \boldsymbol{c}\in\mathbb{R}^l$ and the functions f and g are assumed to be convex [37]. The main benefit of ADMM is that it can combine the decomposition benefits of dual ascent with the desirable convergence properties of the method of multipliers. Letting $\boldsymbol{y}\in\mathbb{R}^l$ denote the dual variable, the augmented Lagrangian of (4.3) is given by

$$\mathcal{L}^A(\boldsymbol{x},\boldsymbol{z},\boldsymbol{y}) = f(\boldsymbol{x})+g(\boldsymbol{z})+\boldsymbol{y}^T(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c})+\frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{z}-\boldsymbol{c}\|_2^2,$$

where $\rho>0$ is the augmented Lagrangian parameter. ADMM then proceeds by iteratively updating the primal variable $\boldsymbol{x}$, updating the primal variable $\boldsymbol{z}$ and taking a gradient ascent step on the dual variable $\boldsymbol{y}$. Explicitly, ADMM consists of the following updates:

1. $\boldsymbol{x}_{t+1}\leftarrow \arg\min_{\boldsymbol{x}} \mathcal{L}^A(\boldsymbol{x},\boldsymbol{z}_t,\boldsymbol{y}_t)$,
2. $\boldsymbol{z}_{t+1}\leftarrow \arg\min_{\boldsymbol{z}} \mathcal{L}^A(\boldsymbol{x}_{t+1},\boldsymbol{z},\boldsymbol{y}_t)$,
3. $\boldsymbol{y}_{t+1}\leftarrow \boldsymbol{y}_t+\rho(\boldsymbol{A}\boldsymbol{x}_{t+1}+\boldsymbol{B}\boldsymbol{z}_{t+1}-\boldsymbol{c})$.

Under very mild regularity conditions on f, g and $\mathcal{L}^A$, it is well known that ADMM is guaranteed to produce a sequence of primal iterates that converges to the optimal value of

(4.3) and a sequence of dual iterates that converge to the optimal dual variable (note that there is no guarantee of primal variable convergence) [37]. Importantly, although ADMM was originally designed for linearly constrained convex optimization, it has often been applied to non-convex optimization problems and yielded empirically strong results [148]. This observation has motivated work to explore the theoretical convergence behavior of ADMM and its variants on specific classes of non-convex optimization problems [78, 141, 143].

4.3 Formulation Properties

In this section, we rigorously investigate certain key features of (4.1). Specifically, we establish an equivalence between (4.1) and an appropriately defined robust optimization problem. Moreover, we illustrate that (4.1) can be reduced to an optimization problem over only $\boldsymbol{X}$ and establish that the resulting objective function is not convex, not concave and non-smooth. Finally, we study how efficient evaluations of the reduced problem objective function can be performed.

4.3.1 Equivalence Between Regularization and Robustness

Real-world datasets frequently contain inaccuracies and missing values, which hinder the ability of machine learning models to generalize effectively to new data when these inconsistencies are not appropriately modelled. Consequently, robustness is a crucial quality for machine learning models, both in theory and application [23, 145]. In this section, we show that our regularized problem (4.1) can be viewed as a robust optimization (RO) problem. This finding justifies the inclusion of the nuclear norm regularization term in (4.1) and is in a similar flavor as known results from the robust optimization literature in the case of vector [13] and matrix [15] problems.

Proposition 32 *Problem (4.1) is equivalent to the following robust optimization problem:*

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times m},\boldsymbol{\alpha}\in\mathbb{R}^{m\times d}} \max_{\boldsymbol{\Delta}\in\mathcal{U}} \sum_{(i,j)\in\Omega} (X_{ij}-A_{ij})^2 + \lambda\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\alpha}\|_F^2 + \langle \boldsymbol{X},\boldsymbol{\Delta}\rangle \quad s.t. \quad rank(\boldsymbol{X}) \leq k_0, \tag{4.4}$$

where $\mathcal{U} = \{\boldsymbol{\Delta}\in\mathbb{R}^{n\times m} : \|\boldsymbol{\Delta}\|_\sigma \leq \gamma\}$.

Proof To establish this result, it suffices to argue that $\max_{\boldsymbol{\Delta}\in\mathcal{U}}\langle \boldsymbol{X},\boldsymbol{\Delta}\rangle = \gamma\|\boldsymbol{X}\|_\star$. This equivalence follows immediately from the fact that the nuclear norm is dual to the spectral norm. So as to keep this manuscript self contained, we present a proof of this equivalence below.

Consider any matrix $\bar{\boldsymbol{\Delta}}\in\mathbb{R}^{n\times m}$ such that $\|\bar{\boldsymbol{\Delta}}\|_\sigma \leq \gamma$. Let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a singular value decomposition of $\boldsymbol{X}$ where we let $r = \mathrm{rank}(\boldsymbol{X})$ and we have $\boldsymbol{U}\in\mathbb{R}^{n\times r}, \boldsymbol{\Sigma}\in\mathbb{R}^{r\times r}, \boldsymbol{V}\in\mathbb{R}^{m\times r}$. We have

$$\begin{aligned}\langle \boldsymbol{X},\bar{\boldsymbol{\Delta}}\rangle &= \mathrm{Tr}(\bar{\boldsymbol{\Delta}}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) = \mathrm{Tr}(\boldsymbol{V}^T\bar{\boldsymbol{\Delta}}^T\boldsymbol{U}\boldsymbol{\Sigma}) = \langle \boldsymbol{U}^T\bar{\boldsymbol{\Delta}}\boldsymbol{V},\boldsymbol{\Sigma}\rangle = \sum_{i=1}^r \Sigma_{ii}(\boldsymbol{U}^T\bar{\boldsymbol{\Delta}}\boldsymbol{V})_{ii} \\ &= \sum_{i=1}^r \Sigma_{ii}U_i^T\bar{\boldsymbol{\Delta}}V_i \leq \sum_{i=1}^r \Sigma_{ii}\sigma_1(\bar{\boldsymbol{\Delta}}) \leq \gamma\sum_{i=1}^r \Sigma_{ii} = \gamma\|\boldsymbol{X}\|_\star,\end{aligned}$$

where we have used the fact that $\boldsymbol{\Sigma}$ is a diagonal matrix and the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ have unit length. Thus, we have shown that $\gamma\|\boldsymbol{X}\|_\star$ is an upper bound for $\max_{\boldsymbol{\Delta}\in\mathcal{U}}\langle \boldsymbol{X},\boldsymbol{\Delta}\rangle$. To show that the upper bound is always achieved, consider the matrix $\tilde{\boldsymbol{\Delta}} = \gamma\boldsymbol{U}\boldsymbol{V}^T \in \mathbb{R}^{n\times m}$ where $\boldsymbol{U}$ and $\boldsymbol{V}$ are taken from a singular value decomposition of $\boldsymbol{X}$. Observe that

$$\|\tilde{\boldsymbol{\Delta}}\|_\sigma = \gamma\|\boldsymbol{U}\boldsymbol{V}^T\|_\sigma \leq \gamma \implies \tilde{\boldsymbol{\Delta}}\in\mathcal{U}.$$

We conclude by noting that

$$\langle \boldsymbol{X},\tilde{\boldsymbol{\Delta}}\rangle = Tr(\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\gamma\boldsymbol{U}\boldsymbol{V}^T) = \gamma Tr(\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{U}) = \gamma\mathrm{Tr}(\boldsymbol{I}\boldsymbol{\Sigma}\boldsymbol{I}) = \gamma\|\boldsymbol{X}\|_\star.$$

■

Proposition 32 implies that solving the nuclear norm regularized (4.1) is equivalent to solving an unregularized robust optimization problem that protects against adversarial perturbations that are bounded in spectral norm. This result is not surprising given the duality of norms, yet is nevertheless insightful.

4.3.2 A Partial Minimization

Let $g(\boldsymbol{X}, \boldsymbol{\alpha})$ denote the objective function of (4.1). Note that $g(\boldsymbol{X}, \boldsymbol{\alpha})$ is bi-convex in $(\boldsymbol{X}, \boldsymbol{\alpha})$ but is not jointly convex due to the product $\boldsymbol{X}\boldsymbol{\alpha}$. Observe that we can simplify (4.1) by performing a partial minimization in $\boldsymbol{\alpha}$. For any $\boldsymbol{X}$, the problem in $\boldsymbol{\alpha}$ requires finding the unconstrained minimum of a convex quadratic function. The gradient of g with respect to $\boldsymbol{\alpha}$ is given by $\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{X}, \boldsymbol{\alpha}) = 2\lambda \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{Y})$. Setting $\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{X}, \boldsymbol{\alpha})$ to 0 yields $\boldsymbol{\alpha}^\star = (\boldsymbol{X}^T\boldsymbol{X})^\dagger \boldsymbol{X}^T\boldsymbol{Y}$ as a minimizer of g over $\boldsymbol{\alpha}$. Note that $\boldsymbol{M}^\dagger$ denotes the pseudo-inverse of a (possibly rank deficient) square matrix $\boldsymbol{M} \in \mathbb{R}^{l \times l}$. Specifically, letting $r = \text{rank}(\boldsymbol{M})$ and $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a singular value decomposition of $\boldsymbol{M}$ with $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{l \times r}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{l \times l}$, we have $\boldsymbol{M}^\dagger = \boldsymbol{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^T$. Letting $f(\boldsymbol{X})$ correspond to the partially minimized objective function of (4.1), we have

$$\begin{aligned} f(\boldsymbol{X}) = \min_{\boldsymbol{\alpha}} g(\boldsymbol{X}, \boldsymbol{\alpha}) &= \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda \|(\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger \boldsymbol{X}^T)\boldsymbol{Y}\|_F^2 + \gamma \|\boldsymbol{X}\|_\star \\ &= \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda \text{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger \boldsymbol{X}^T)\boldsymbol{Y}\big) + \gamma \|\boldsymbol{X}\|_\star. \end{aligned}$$

We note that $\boldsymbol{\alpha}^\star$ corresponds to the well studied ordinary least squares solution. When $\boldsymbol{X}^T\boldsymbol{X}$ has full rank, $\boldsymbol{\alpha}$ is the unique minimizer of g. If $\boldsymbol{X}^T\boldsymbol{X}$ is rank deficient, $\boldsymbol{\alpha}^\star$ corresponds to the minimizer with minimum norm.

Though we have simplified the objective function of (4.1), $f(\boldsymbol{X})$ is not a particularly well behaved function. We formalize this statement in Proposition (33).

Proposition 33 *The function $f(\boldsymbol{X})$ is in general neither convex nor concave and is non-*

smooth.

Proof To illustrate that $f(\boldsymbol{X})$ is in general neither convex nor concave, suppose that $\Omega = \emptyset$, $n = 2$ and $m = d = \lambda = \gamma = 1$. In this setting, we have $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{2\times 1}$. Assuming that $\boldsymbol{x} \neq \boldsymbol{0}_2$, we can write the objective function as

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{Tr}(\boldsymbol{y}^T(\boldsymbol{I}_2 - \boldsymbol{x}(\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}^T)\boldsymbol{y}) + \|\boldsymbol{x}\|_\star \\ &= \boldsymbol{y}^T\boldsymbol{y} - \frac{\boldsymbol{y}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{y}}{\boldsymbol{x}^T\boldsymbol{x}} + \|\boldsymbol{x}\|_2 \\ &= \boldsymbol{y}^T\boldsymbol{y} - \frac{(\boldsymbol{y}^T\boldsymbol{x})^2}{\boldsymbol{x}^T\boldsymbol{x}} + \sqrt{\boldsymbol{x}^T\boldsymbol{x}}. \end{aligned}$$

For $\boldsymbol{x} = \boldsymbol{0}_2$, the objective value $f(\boldsymbol{0}_2)$ is equal to $\boldsymbol{y}^T\boldsymbol{y}$. Let $\boldsymbol{y} = \boldsymbol{1}_2$ and consider the line in $\boldsymbol{R}^2$ defined by $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 : x_2 = x_1 + 1\}$. The restriction of $f(\boldsymbol{x})$ to the line defined by $\mathcal{X}$ is a univariate function given by

$$f_{\mathcal{X}}(t) = 2 - \frac{(2t+1)^2}{2t^2 + 2t + 1} + \sqrt{2t^2 + 2t + 1},$$

where $t \in \mathbb{R}$ is a dummy variable. Observe that we have $f_{\mathcal{X}}(-1) = f_{\mathcal{X}}(0) = 2, f_{\mathcal{X}}(-0.5) = 2 + \frac{\sqrt{2}}{2}$ and $f_{\mathcal{X}}(-4) = f_{\mathcal{X}}(3) = 5.04$. Thus, the point $(-0.5, f_{\mathcal{X}}(0.5))$ lies above the chord connecting $(-1, f_{\mathcal{X}}(-1))$ and $(0, f_{\mathcal{X}}(0))$, so $f_{\mathcal{X}}(t)$ is not a convex function. Moreover, the point $(-1, f_{\mathcal{X}}(-1))$ lies below the chord connecting $(-4, f_{\mathcal{X}}(-4))$ and $(-0.5, f_{\mathcal{X}}(-0.5))$, so $f_{\mathcal{X}}(t)$ is not a concave function. Since a function is convex (respectively concave) if and only if its restriction to every line is convex (respectively concave), we have established that $f(\boldsymbol{X})$ is neither convex nor concave since $f_{\mathcal{X}}(t)$ is neither convex nor concave. To conclude the proof of Proposition 33, note that the non-smooth property of $f(\boldsymbol{X})$ follows immediately from the non-smooth property of the nuclear norm function. ■

Although the above closed form partial minimization in $\boldsymbol{\alpha}$ eliminates $m \times d$ variables form (4.1), this comes at the expense of introducing a $m \times m$ matrix pseudo-inverse term

into the objective function which can be computationally expensive to evaluate. Efficient evaluation of an objective function is crucial in many optimization problems to quickly measure solution quality. A plethora of modern optimization techniques require iterative objective function evaluations. As a result, the computational cost of evaluating an objective function can quickly become the bottleneck of an algorithm's complexity. Directly evaluating $f(\boldsymbol{X})$ naively requires $O(|\Omega|)$ operations for the first term, $O(m^2n+m^3+n^2d)$ for the second term (forming the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is $O(m^2n)$, taking the pseudo-inverse is $O(m^3)$, computing the products involving $\boldsymbol{Y}$ is $O(n^2d)$) and requires $O(mn\min(m,n))$ for the third term (the nuclear norm can be evaluated by computing a singular value decomposition of $\boldsymbol{X}$). We observe that computing the second term of $f(\boldsymbol{X})$ involving the pseudo-inverse dominates the complexity calculation. Indeed, the overall complexity of evaluating $f(\boldsymbol{X})$ naively is $O(m^2n + m^3 + n^2d)$.

Fortunately, it is possible to make evaluations of $f(\boldsymbol{X})$ without explicitly forming the product $\boldsymbol{X}^T\boldsymbol{X}$ or taking a pseudo-inverse. Proposition 34 illustrates that it suffices (in terms of computational complexity) to take a singular value decomposition of $\boldsymbol{X}$. Moreover, a large class of optimization algorithms require only function evaluations for feasible solutions. If we consider only those values of $\boldsymbol{X}$ that are feasible to (4.1), it is sufficient (in terms of computational complexity) to take a rank k truncated singular value decomposition of $\boldsymbol{X}$ to make functions evaluations of $f(\boldsymbol{X})$.

Proposition 34 *The function $f(\boldsymbol{X})$ can equivalently be written as*

$$f(\boldsymbol{X}) = \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda\, Tr\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{U}\boldsymbol{U}^T)\boldsymbol{Y}\big) + \gamma\|\boldsymbol{X}\|_\star,$$

where $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a singular value decomposition of $\boldsymbol{X}$ where we let $r = rank(\boldsymbol{X})$ and we have $\boldsymbol{U} \in \mathbb{R}^{n\times r}, \boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}, \boldsymbol{V} \in \mathbb{R}^{m\times r}$.

Proof To establish the result, it suffices to show that

$$\mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger\boldsymbol{X}^T\boldsymbol{Y}\big) = \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Y}\big).$$

Let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a singular value decomposition of $\boldsymbol{X}$ where $r = \mathrm{rank}(\boldsymbol{X})$ and $\boldsymbol{U} \in \mathbb{R}^{n\times r}, \boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}, \boldsymbol{V} \in \mathbb{R}^{m\times r}$. Observe that

$$\begin{aligned}
\mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^\dagger\boldsymbol{X}^T\boldsymbol{Y}\big) &= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^\dagger\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T)^\dagger\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-2}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{Y}\big)\\
&= \mathrm{Tr}\big(\boldsymbol{Y}^T\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Y}\big),
\end{aligned}$$

where we have repeatedly invoked the property that $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_r$. ■

In light of Proposition 34, evaluating $f(\boldsymbol{X})$ for feasible solutions still requires $O(|\Omega|)$ operations for the first term, but the second term can be evaluated using $O(kn(m+d))$ operations (performing a truncated singular value decomposition is $O(knm)$ and computing the products involving $\boldsymbol{Y}$ is $O(knd)$) and the third term can be evaluated using $O(knm)$ operations (by performing a truncated singular value decomposition) for an overall complexity of $O(kn(m+d))$. This is significantly less expensive than the $O(m^2n + m^3 + n^2d)$ complexity of naive direct evaluation of $f(\boldsymbol{X})$ introduced previously.

4.4 An Exact Mixed-Projection Formulation

In this section, we reformulate (4.1) as a mixed-projection optimization problem and further reduce the dimension of the resulting problem in a commonly studied manner by parameterizing $\boldsymbol{X}$ as the matrix product of two low dimensional matrices. Thereafter, we illustrate how to employ the matrix generalization of the perspective relaxation [15, 19, 20, 77] to

construct a convex relaxation of (4.1).

We first note that given the result of Section 4.3.2, we can rewrite (4.1) as an optimization problem only over $\boldsymbol{X}$ as follows:

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{R}^{n \times m}} \quad & \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 + \lambda \mathrm{Tr}\big(\boldsymbol{Y}^T (\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{X})^\dagger \boldsymbol{X}^T) \boldsymbol{Y}\big) + \gamma \|\boldsymbol{X}\|_\star \\ \text{s.t.} \quad & \mathrm{rank}(\boldsymbol{X}) \leq k. \end{aligned} \tag{4.5}$$

Observe that the matrix $\boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{X})^\dagger \boldsymbol{X}^T$ is the linear transformation that projects vectors onto the subspace spanned by the columns of the matrix $\boldsymbol{X}$. Drawing on ideas presented in [15, 19, 20], we introduce an orthogonal projection matrix $\boldsymbol{P} \in \mathcal{P}_k$ to model the column space of $\boldsymbol{X}$ where $\mathcal{P}_\eta = \{\boldsymbol{P} \in \mathcal{S}^n : \boldsymbol{P}^2 = \boldsymbol{P}, \mathrm{tr}(\boldsymbol{P}) \leq \eta\}$ for $\eta \geq 0$. We can express the desired relationship between $\boldsymbol{P}$ and $\boldsymbol{X}$ as $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}$ since projecting a matrix onto its own column space leaves the matrix unchanged. This gives the following reformulation of (4.1):

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{R}^{n \times m}, \boldsymbol{P} \in \mathbb{R}^{n \times n}} \quad & \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 + \lambda \mathrm{Tr}\big(\boldsymbol{Y}^T (\boldsymbol{I}_n - \boldsymbol{P}) \boldsymbol{Y}\big) + \gamma \|\boldsymbol{X}\|_\star \\ \text{s.t.} \quad & (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{X} = \boldsymbol{0}_{n \times m}, \ \boldsymbol{P} \in \mathcal{P}_{\min(k, \mathrm{rank}(\boldsymbol{X}))}. \end{aligned} \tag{4.6}$$

Observe that the matrix pseudo-inverse term has been eliminated from the objective function, however we have introduced the bilinear constraint $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}$ which is non-convex in the optimization variables as well as the non-convex constraint $\boldsymbol{P} \in \mathcal{P}_{\min(k, \mathrm{rank}(\boldsymbol{X}))}$. We now have the following result:

Proposition 35 *Problem (4.6) is a valid reformulation of (4.5).*

Proof We show that given a feasible solution to (4.6), we can construct a feasible solution to (4.5) that achieves the same objective value and vice versa.

Consider an arbitrary feasible solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ to (4.6). Since $\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{P}} \in \mathcal{P}_{\min(k, \mathrm{rank}(\bar{\boldsymbol{X}}))}$, we have $\mathrm{rank}(\bar{\boldsymbol{X}}) \leq k$. We claim that $\bar{\boldsymbol{X}}$ achieves the same objective value

in (4.5) as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ achieves in (4.6). To show this, it suffices to illustrate that for all $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ feasible to (4.6) we have $H(\bar{\boldsymbol{X}}) := \bar{\boldsymbol{X}}(\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}})^\dagger\bar{\boldsymbol{X}}^T = \bar{\boldsymbol{P}}$. The matrix $\bar{\boldsymbol{P}}$ is an orthogonal projection matrix since it is symmetric and satisfies $\bar{\boldsymbol{P}}^2 = \bar{\boldsymbol{P}}$. Moreover, since $\text{rank}(\bar{\boldsymbol{P}}) = \text{rank}(\bar{\boldsymbol{X}})$ and $\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}$ we know that $\bar{\boldsymbol{P}}$ is an orthogonal projection onto the subspace spanned by the columns of $\bar{\boldsymbol{X}}$. Similarly, it can easily be verified that $H(\bar{\boldsymbol{X}})$ is symmetric and satisfies $H(\bar{\boldsymbol{X}})^2 = H(\bar{\boldsymbol{X}})$, $\text{rank}(H(\bar{\boldsymbol{X}})) = \text{rank}(\bar{\boldsymbol{X}})$ and $H(\bar{\boldsymbol{X}})\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}$. Thus, $H(\bar{\boldsymbol{X}})$ is also an orthogonal projection matrix onto the subspace spanned by the columns of $\bar{\boldsymbol{X}}$. To conclude, we invoke the property that given a subspace $\mathcal{V} \subset \mathbb{R}^n$ the orthogonal projection onto $\mathcal{V}$ is uniquely defined. To see this, suppose $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are two orthogonal projections onto $\mathcal{V}$. Let $l = \dim(\mathcal{V})$. Let $\{\boldsymbol{e}_i\}_{i=1}^l$ be an orthogonal basis for $\mathcal{V}$ and let $\{\boldsymbol{e}_i\}_{l+1}^n$ be an orthogonal basis for $\mathcal{V}^\perp$. Since $\boldsymbol{P}_1$ is an orthogonal projection onto $\mathcal{V}$, we have $\boldsymbol{P}_1\boldsymbol{e}_i = \boldsymbol{e}_i$ for all $1 \leq i \leq l$ and $\boldsymbol{P}_1\boldsymbol{e}_i = \boldsymbol{0}_n$ for all $l+1 \leq i \leq n$. However, the same must hold for $\boldsymbol{P}_2$ which implies that $\boldsymbol{P}_1 = \boldsymbol{P}_2$.

Consider an arbitrary feasible solution $\bar{\boldsymbol{X}}$ to (4.5). Let $r = \text{rank}(\bar{\boldsymbol{X}})$ and $\bar{\boldsymbol{X}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^T$ be a singular value decomposition of $\bar{\boldsymbol{X}}$ where we have $\bar{\boldsymbol{U}} \in \mathbb{R}^{n\times r}, \bar{\boldsymbol{\Sigma}} \in \mathbb{R}^{r\times r}, \bar{\boldsymbol{V}} \in \mathbb{R}^{m\times r}$. Define $\bar{\boldsymbol{P}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{U}}^T$. By construction, we have $\bar{\boldsymbol{P}} \in \mathcal{P}_{\min(k,\text{rank}(\bar{\boldsymbol{X}}))}$ since $r \leq k$. Moreover, it is easy to verify that

$$\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{U}}^T\bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^T = \bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^T = \bar{\boldsymbol{X}},$$

where we have used the property $\bar{\boldsymbol{U}}^T\bar{\boldsymbol{U}} = \boldsymbol{I}_r$. Finally, Proposition 34 immediately implies that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ achieves the same objective in (4.6) as $\bar{\boldsymbol{X}}$ achieves in (4.5). This completes the proof. ■

Optimizing explicitly over the space of $n \times m$ matrices can rapidly become prohibitively costly in terms of runtime and memory requirements. Accordingly, we adopt the common approach of factorizing $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ as $\boldsymbol{U}\boldsymbol{V}^T$ for $\boldsymbol{U} \in \mathbb{R}^{n\times k}, \boldsymbol{V} \in \mathbb{R}^{m\times k}$. This leads to the following formulation:

$$\begin{aligned} \min_{\substack{\boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{V}\in\mathbb{R}^{m\times k},\\ \boldsymbol{P}\in\mathbb{R}^{n\times n}}} \quad & \sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij}-A_{ij})^2+\lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Y}\big)+\frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2+\|\boldsymbol{V}\|_F^2) \\ \text{s.t.} \quad & (\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{U}=\boldsymbol{0}_{n\times k},\ \boldsymbol{P}\in\mathcal{P}_{\min(k,\mathrm{rank}(\boldsymbol{U}\boldsymbol{V}^T))}. \end{aligned} \tag{4.7}$$

Notice that we have replaced $n \times m$ optimization variables with $k \times (n + m)$ optimization variables, an often significant dimension reduction in practice. Attentive readers may object that though this is true, we have introduced n^2 decision variables through the introduction of the projection matrix variable $\boldsymbol{P}$ which nullifies any savings introduced through the factorization of $\boldsymbol{X}$. Note, however, that it is possible to factor any feasible projection matrix as $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T$ for some $\boldsymbol{M} \in \mathbb{R}^{n\times k}$. In Section 4.5, we leverage this fact so that the presence of the projection matrix incurs a cost of $n \times k$ additional variables rather than n^2 variables. We have the following result:

Proposition 36 *Problem* (4.7) *is a valid reformulation of* (4.6).

Proof We show that given a feasible solution to (4.7), we can construct a feasible solution to (4.6) that achieves the same or lesser objective value and vice versa.

Consider an arbitrary feasible solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ to (4.7). Let $\bar{\boldsymbol{X}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T$. We will show that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ is feasible to (4.6) and achieves the same or lesser objective as $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ does in (4.7). Feasibility of (4.7) implies that $\bar{\boldsymbol{P}} \in \mathcal{P}_{\min(k,\mathrm{rank}(\bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T))} = \mathcal{P}_{\min(k,\mathrm{rank}(\bar{\boldsymbol{X}}))}$ and also that

$$(\boldsymbol{I}_n-\bar{\boldsymbol{P}})\bar{\boldsymbol{X}}=(\boldsymbol{I}_n-\bar{\boldsymbol{P}})\bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T=\boldsymbol{0}_{n\times k}\bar{\boldsymbol{V}}^T=\boldsymbol{0}_{n\times m},$$

thus the solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ is certainly feasible for (4.6). To see that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ achieves the same or lesser objective value, it suffices to argue that $\|\bar{\boldsymbol{X}}\|_\star \leq \frac{1}{2}(\|\bar{\boldsymbol{U}}\|_F^2 + \|\bar{\boldsymbol{V}}\|_F^2)$. This follows immediately from the following lemma established by [105] (see Appendix A.5 in their paper for a proof):

Lemma 37 *For any matrix $\boldsymbol{Z}$, the following holds:*

$$\|\boldsymbol{Z}\|_\star = \min_{\boldsymbol{U},\boldsymbol{V}:\boldsymbol{Z}=\boldsymbol{U}\boldsymbol{V}^T} \frac{1}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2).$$

If rank$(\boldsymbol{Z}) = k \leq \min(m,n)$, then the minimum above is attained at a factor decomposition $\boldsymbol{U}_{n\times k}\boldsymbol{V}_{m\times k}^T$. Letting $\boldsymbol{Z}_{n\times m} = \boldsymbol{L}_{n\times k}\boldsymbol{\Sigma}_{k\times k}\boldsymbol{R}_{m\times k}^T$ denote a singular value decomposition of $\boldsymbol{Z}$, the minimum above is attained at $\boldsymbol{U}_{n\times k} = \boldsymbol{L}_{n\times k}\boldsymbol{\Sigma}_{k\times k}^{\frac{1}{2}}, \boldsymbol{V}_{m\times k} = \boldsymbol{R}_{m\times k}\boldsymbol{\Sigma}_{k\times k}^{\frac{1}{2}}$.

Consider now an arbitrary feasible solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ to (4.6). Let $\bar{\boldsymbol{X}} = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}^T$ be a singular value decomposition of $\bar{\boldsymbol{X}}$ where $\boldsymbol{L} \in \mathbb{R}^{n\times k}, \boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}, \boldsymbol{R} \in \mathbb{R}^{m\times k}$ and define $\bar{\boldsymbol{U}} = \boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}}, \bar{\boldsymbol{V}} = \boldsymbol{R}\boldsymbol{\Sigma}^{\frac{1}{2}}$. Feasibility of $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ in (4.6) implies that $\bar{\boldsymbol{P}} \in \mathcal{P}_{\min(k,\text{rank}(\bar{\boldsymbol{X}}))} = \mathcal{P}_{\min(k,\text{rank}(\bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T))}$. Moreover, since the columns of $\boldsymbol{L}$ form an orthogonal basis for the columns space of $\bar{\boldsymbol{X}}$, the condition $(\boldsymbol{I}_n - \bar{\boldsymbol{P}})\bar{\boldsymbol{X}} = \boldsymbol{0}_{n\times m}$ implies that

$$(\boldsymbol{I}_n - \bar{\boldsymbol{P}})\bar{\boldsymbol{U}} = (\boldsymbol{I}_n - \bar{\boldsymbol{P}})\boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{0}_{n\times k}\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{0}_{n\times k}.$$

Thus, the solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ is feasible to (4.7). Moreover, by Lemma 37 we have $\frac{1}{2}(\|\bar{\boldsymbol{U}}\|_F^2 + \|\bar{\boldsymbol{V}}\|_F^2) = \|\bar{\boldsymbol{X}}\|_\star$ so $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ achieves the same objective in (4.7) as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ does in (4.6). This completes the proof.

■

In the remainder of the chapter, we will relax the constraint $\boldsymbol{P} \in \mathcal{P}_{\min(k,\text{rank}(\boldsymbol{U}\boldsymbol{V}^T))}$ to $\boldsymbol{P} \in \mathcal{P}_k$ and develop a scalable algorithm to obtain high quality feasible solutions. Explicitly, we consider the problem given by:

$$\begin{aligned} \min_{\substack{\boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{V}\in\mathbb{R}^{m\times k},\\ \boldsymbol{P}\in\mathbb{R}^{n\times n}}} \quad & \sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij}-A_{ij})^2+\lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Y}\big)+\frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2+\|\boldsymbol{V}\|_F^2) \\ \text{s.t.} \quad & (\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{U}=\boldsymbol{0}_{n\times k},\ \boldsymbol{P}\in\mathcal{P}_k. \end{aligned} \tag{4.8}$$

It is straightforward to see that the optimal value of (4.8) is no greater than the optimal value of (4.7). Unfortunately, the converse does not necessarily hold. To see why the optimal value of (4.8) can be strictly less than that of (4.7) in certain pathological cases, suppose we had $k = n = m$, $\Omega = \emptyset$. In this setting, letting $\bar{\boldsymbol{P}} = \boldsymbol{I}_n$, $\boldsymbol{U} = \boldsymbol{0}_{n\times k}$ and $\bar{\boldsymbol{V}} = \boldsymbol{0}_{m\times k}$, the solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ would be feasible to (4.8) and achieve an objective value of 0. However the optimal value of (4.7) would be strictly greater than 0 in this setting as long as $\boldsymbol{Y} \neq 0$. Although (4.8) is a relaxation of (4.1), we will see in Section 4.6 that the solutions we will obtain to (4.8) will be high quality solutions for (4.1), the main problem of interest.

4.4.1 A Positive Semidefinite Cone Relaxation

Convex relaxations are useful in non-convex optimization primarily for two reasons. Firstly, given the objective value achieved by an arbitrary feasible solution, strong convex relaxations can be used to upperbound the worst case suboptimality of said solution. Secondly, convex relaxations can often be used as building blocks for global optimization procedures. In this section, we present a natural convex relaxation of (4.8) that leverages the matrix generalization of the perspective relaxation [15, 19, 20, 77].

Rather than working directly with (4.8), consider the equivalent formulation (4.6) with $\mathcal{P}_{\min(k,\mathrm{rank}(\boldsymbol{X}))}$ replaced by $\mathcal{P}_k$. Before proceeding, we will assume knowledge of an upper bound $M \in \mathbb{R}_+$ on the spectral norm of an optimal $\boldsymbol{X}$ to (4.6). Tighter bounds M are desirable as they will lead to stronger convex relaxations of (4.6). We note that it is always possible to specify such an upper bound M without prior knowledge of an optimal solution to (4.6). To see this, note that setting $\boldsymbol{X} = \boldsymbol{0}_{n\times m}$ in (4.6) produces an objective value

of $\sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2$. Thus, any $\boldsymbol{X}$ such that $\gamma\|\boldsymbol{X}\|_\star > \sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2$ cannot possibly be optimal to (4.6). Finally, since the nuclear norm is an upper bound on the spectral norm of a matrix, we must have

$$\|\boldsymbol{X}\|_\sigma \leq \frac{\sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2}{\gamma},$$

for any matrix $\boldsymbol{X}$ that is optimal to (4.6). We can therefore take $M = \frac{\sum_{(i,j)\in\Omega} A_{ij}^2 + \lambda\|\boldsymbol{Y}\|_F^2}{\gamma}$.

Notice that the non-convexity in (4.6) is captured entirely by the bilinear constraint $(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{X} = \boldsymbol{0}_{n\times m}$ and the quadratic constraint $\boldsymbol{P}^2 = \boldsymbol{P}$. In keeping with the approach presented in [15, 20], we leverage the matrix perspective to convexify the bilinear term and solve over the convex hull of the set $\mathcal{P}_k$. Recalling that the nuclear norm is semidefinite representable, we have the following formulation:

$$\begin{aligned}
\min_{\substack{\boldsymbol{P},\boldsymbol{W}_1\in\mathbb{R}^{n\times n},\\ \boldsymbol{X}\in\mathbb{R}^{n\times m},\boldsymbol{W}_2\in\mathbb{R}^{m\times m}}} \quad & \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) + \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{W}_1) + \mathrm{Tr}(\boldsymbol{W}_2)) \\
\text{s.t.} \quad & \boldsymbol{I}_n \succeq \boldsymbol{P} \succeq 0,\ \mathrm{Tr}(\boldsymbol{P}) \leq k, \\
& \begin{pmatrix} M\boldsymbol{P} & \boldsymbol{X} \\ \boldsymbol{X}^T & M\boldsymbol{I}_m \end{pmatrix} \succeq 0, \begin{pmatrix} \boldsymbol{W}_1 & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{W}_2 \end{pmatrix} \succeq 0.
\end{aligned} \tag{4.9}$$

We now have the following result:

Proposition 38 *Problem (4.9) is a valid convex relaxation of (4.8).*

Proof Problem (4.9) is clearly a convex optimization problem. We will show that the optimal value of (4.9) is a lower bound on the optimal value of (4.8) by showing that given any optimal solution to (4.8), we can construct a feasible solution to (4.9) that achieves the same objective value.

Consider any optimal solution $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ to (4.8). From the proof of Proposition 36, we know that the solution $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ where $\bar{\boldsymbol{X}} = \bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^T$ is feasible to (4.6) (where we replace the

constraint $\boldsymbol{P} \in \mathcal{P}_{\min(k,\mathrm{rank}(\boldsymbol{U}\boldsymbol{V}^T))}$ with $\boldsymbol{P} \in \mathcal{P}_k$) and must also be optimal. Let $\bar{\boldsymbol{X}} = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}^T$ be a singular value decomposition of $\bar{\boldsymbol{X}}$ with $\boldsymbol{L} \in \mathbb{R}^{n\times k}, \boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times k}$. Let $\bar{\boldsymbol{W}}_1 = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T$ and $\bar{\boldsymbol{W}}_2 = \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T$. We claim that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2)$ is feasible to (4.9) and achieves the same objective value as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ does in (4.6).

From the feasibility of $\bar{\boldsymbol{P}}$ in (4.8), we know that $\bar{\boldsymbol{P}} \in \mathcal{P}_k$ which implies $\boldsymbol{I}_n \succeq \bar{\boldsymbol{P}} \succeq 0$ and $\mathrm{Tr}(\bar{\boldsymbol{P}}) \leq k$. By the generalized Schur complement lemma (see [36], Equation 2.41), we know that

$$\begin{pmatrix} M\bar{\boldsymbol{P}} & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & M\boldsymbol{I}_m \end{pmatrix} \succeq 0 \iff M\boldsymbol{I}_m \succeq 0, \text{ and } M\boldsymbol{I}_m - \bar{\boldsymbol{X}}^T(M\bar{\boldsymbol{P}})^\dagger\bar{\boldsymbol{X}} \succeq 0.$$

We trivially have $M\boldsymbol{I}_m \succeq 0$. To see that the second condition holds, note that since $\bar{\boldsymbol{P}}$ is a projection matrix and $\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}}$, we have $\bar{\boldsymbol{X}}^T(M\bar{\boldsymbol{P}})^\dagger\bar{\boldsymbol{X}} = \frac{1}{M}\bar{\boldsymbol{X}}^T\bar{\boldsymbol{P}}\bar{\boldsymbol{X}} = \frac{1}{M}\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}$. Furthermore, since $\bar{\boldsymbol{X}}$ is optimal to (4.6), we have $\|\bar{\boldsymbol{X}}\|_\sigma \leq M$ by assumption. Thus, we have

$$\|\bar{\boldsymbol{X}}\|_\sigma \leq M \implies \|\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}\|_\sigma \leq M^2 \implies M^2\boldsymbol{I}_m \succeq \bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}} \implies M\boldsymbol{I}_m \succeq \frac{1}{M}\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}.$$

Finally, observe that

$$\begin{pmatrix} \bar{\boldsymbol{W}}_1 & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \bar{\boldsymbol{W}}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T & \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}^T \\ \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{L}^T & \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T \end{pmatrix} = \begin{pmatrix} \boldsymbol{L} \\ \boldsymbol{R} \end{pmatrix} \boldsymbol{\Sigma} \begin{pmatrix} \boldsymbol{L} \\ \boldsymbol{R} \end{pmatrix}^T.$$

Since $\boldsymbol{\Sigma}$ is a diagonal matrix with non negative entries, the matrix $\begin{pmatrix} \bar{\boldsymbol{W}}_1 & \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^T & \bar{\boldsymbol{W}}_2 \end{pmatrix}$ is certainly positive semidefinite. Thus we have shown that $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2)$ is indeed feasible to (4.9). To conclude the proof, we note that

$$\begin{aligned} \frac{\gamma}{2}(\mathrm{Tr}(\bar{\boldsymbol{W}}_1) + \mathrm{Tr}(\bar{\boldsymbol{W}}_2)) &= \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T) + \mathrm{Tr}(\boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T)) = \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{\Sigma}) + \mathrm{Tr}(\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{\Sigma})) \\ &= \frac{\gamma}{2}(\mathrm{Tr}(\boldsymbol{\Sigma}) + \mathrm{Tr}(\boldsymbol{\Sigma})) = \gamma\|\bar{\boldsymbol{X}}\|_\star, \end{aligned}$$

thus $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{W}}_1, \bar{\boldsymbol{W}}_2)$ achieves the same objective value in (4.9) as $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{P}})$ achieves in (4.6). ■

In general, an optimal solution to (4.9) will have $\boldsymbol{P} \notin \mathcal{P}_k$. We briefly note that to obtain a stronger convex relaxation, one could leverage eigenvector disjunctions [16, 127] to iteratively cut off solutions to (4.9) with $\boldsymbol{P} \notin \mathcal{P}_k$ and form increasingly tighter disjunctive approximations to the set $\mathcal{P}_k$.

4.5 Mixed-Projection ADMM

In this section, we present an alternating direction method of multipliers (ADMM) algorithm that is scalable and obtains high quality solutions for (4.8) and we investigate its convergence properties. Rather than forming the augmented Lagrangian directly for (4.8), we first modify our problem formulation by introducing a dummy variable $\boldsymbol{Z} \in \mathbb{R}^{n\times k}$ that is an identical copy of $\boldsymbol{U}$. Additionally, rather than directly enforcing the constraint $\boldsymbol{P} \in \mathcal{P}_k$, we introduce an indicator function penalty $\mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P})$ into the objective function where $\mathbb{I}_{\mathcal{X}}(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \in \mathcal{X}$, otherwise $\mathbb{I}_{\mathcal{X}}(\boldsymbol{x}) = \infty$. Explicitly, we consider the following problem:

$$\begin{aligned} \min_{\substack{\boldsymbol{U},\boldsymbol{Z}\in\mathbb{R}^{n\times k},\\ \boldsymbol{V}\in\mathbb{R}^{m\times k},\\ \boldsymbol{P}\in\mathbb{R}^{n\times n}}} \quad & \sum_{(i,j)\in\Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \lambda \mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Y}\big) + \frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P}) \\ \text{s.t.} \quad & (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{U} = \boldsymbol{0}_{n\times k},\ \boldsymbol{U} - \boldsymbol{Z} = \boldsymbol{0}_{n\times k}. \end{aligned} \tag{4.10}$$

It is trivial to see that (4.10) is equivalent to (4.8). We will see in this section that working with formulation (4.10) leads to an ADMM algorithm with favorable decomposition properties. Introducing dual variables $\boldsymbol{\Phi}, \boldsymbol{\Psi} \in \mathbb{R}^{n\times k}$ for the constraints $(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{U} = \boldsymbol{0}_{n\times k}$ and $\boldsymbol{U} - \boldsymbol{Z} = \boldsymbol{0}_{n\times k}$ respectively, the augmented Lagrangian $\mathcal{L}^A$ for (4.10) is given by:

$$\begin{aligned}\mathcal{L}^A(\boldsymbol{U},\boldsymbol{V},\boldsymbol{P},\boldsymbol{Z},\boldsymbol{\Phi},\boldsymbol{\Psi}) = &\sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij}-A_{ij})^2+\lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Y}\big)\\ &+\frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2+\|\boldsymbol{V}\|_F^2)+\mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P})+\mathrm{Tr}(\boldsymbol{\Phi}^T(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Z})\\ &+\mathrm{Tr}(\boldsymbol{\Psi}^T(\boldsymbol{Z}-\boldsymbol{U}))+\frac{\rho_1}{2}\|(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Z}\|_F^2+\frac{\rho_2}{2}\|\boldsymbol{Z}-\boldsymbol{U}\|_F^2,\end{aligned} \tag{4.11}$$

where $\rho_1, \rho_2 > 0$ are non-negative penalty parameters. In what follows, we show that performing a partial minimization of the augmented Lagrangian (4.11) over each of the primal variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}$ yields a subproblem that can be solved efficiently. We present each subproblem and investigate the complexity of computing the corresponding subproblem solutions.

4.5.1 Subproblem in $\boldsymbol{U}$

First, suppose we fix variables $\boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{U}$. Eliminating terms that do not depend on $\boldsymbol{U}$, the resulting subproblem is given by

$$\min_{\boldsymbol{U}\in\mathbb{R}^{n\times k}}\sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij}-A_{ij})^2+\frac{\gamma}{2}\|\boldsymbol{U}\|_F^2-\mathrm{Tr}(\boldsymbol{\Psi}^T\boldsymbol{U})+\frac{\rho_2}{2}\|\boldsymbol{Z}-\boldsymbol{U}\|_F^2. \tag{4.12}$$

We now have the following result:

Proposition 39 *The optimal solution $\bar{\boldsymbol{U}}$ for (4.12) is given by*

$$\bar{U}_{i,\star} = [2\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}+(\gamma+\rho_2)\boldsymbol{I}_k]^{-1}[2\boldsymbol{V}^T\boldsymbol{W}_iA_{i,\star}+\Psi_{i,\star}+\rho_2 Z_{i,\star}], \tag{4.13}$$

for each $i \in \{1,\ldots,n\}$ where each $\boldsymbol{W}_i \in \mathbb{R}^{m\times m}$ is a diagonal matrix satisfying $(\boldsymbol{W}_i)_{jj} = 1$ if $(i,j) \in \Omega$, otherwise $(\boldsymbol{W}_i)_{jj} = 0$. Here, the column vectors $\bar{U}_{i,\star} \in \mathbb{R}^k, A_{i,\star} \in \mathbb{R}^m, \Psi_{i,\star} \in \mathbb{R}^k, Z_{i,\star} \in \mathbb{R}^k$ denote the i^{th} row of the matrices $\bar{\boldsymbol{U}}, \boldsymbol{A}, \boldsymbol{\Psi}, \boldsymbol{Z}$ respectively, where the unknown entries of $\boldsymbol{A}$ are taken to be 0.

Proof Let $f(\boldsymbol{U})$ denote the objective function of (4.12). With $\{\boldsymbol{W}_i\}_{i=1}^n$ defined as in Proposition 39, observe that we can write $f(\boldsymbol{U})$ as

$$\begin{aligned} f(\boldsymbol{U}) &= \sum_{i=1}^n \|\boldsymbol{W}_i(\boldsymbol{V}U_{i,\star} - A_{i,\star})\|_2^2 + \frac{\gamma}{2}\sum_{i=1}^n \|U_{i,\star}\|_2^2 - \sum_{i=1}^n \Psi_{i,\star}^T U_{i,\star} + \frac{\rho_2}{2}\sum_{i=1}^n \|Z_{i,\star} - U_{i,\star}\|_2^2 \\ &= \sum_{i=1}^n \left[\|\boldsymbol{W}_i(\boldsymbol{V}U_{i,\star} - A_{i,\star})\|_2^2 + \frac{\gamma}{2}\|U_{i,\star}\|_2^2 - \Psi_{i,\star}^T U_{i,\star} + \frac{\rho_2}{2}\|Z_{i,\star} - U_{i,\star}\|_2^2\right] = \sum_{i=1}^n g_i(\boldsymbol{U}), \end{aligned}$$

where we define $g_i(\boldsymbol{U}) = \|\boldsymbol{W}_i(\boldsymbol{V}U_{i,\star} - A_{i,\star})\|_2^2 + \frac{\gamma}{2}\|U_{i,\star}\|_2^2 - \Psi_{i,\star}^T U_{i,\star} + \frac{\rho_2}{2}\|Z_{i,\star} - U_{i,\star}\|_2^2$. Thus, we have shown that $f(\boldsymbol{U})$ is separable over the rows of the matrix $\boldsymbol{U}$. Each function $g_i(\boldsymbol{U})$ is a (strongly) convex quadratic. Thus, we can minimize $g_i(\boldsymbol{U})$ by setting its gradient to 0. For any fixed row $i \in \{1, \ldots, n\}$, we can differentiate and obtain

$$\nabla_{\boldsymbol{U}_{i,\star}} g_i(\boldsymbol{U}) = 2\boldsymbol{V}^T\boldsymbol{W}_i(\boldsymbol{V}U_{i,\star} - A_{i\star}) + \gamma U_{i,\star} - \Psi_{i,\star} - \rho_2(Z_{i,\star} - U_{i,\star}).$$

By equating the gradient $\nabla_{\boldsymbol{U}_{i,\star}} g_i(\boldsymbol{U})$ to 0 and rearranging, we obtain that the optimal vector $\bar{U}_{i,\star}$ is given by (4.13). This completes the proof. ■

Observe that since the matrix $\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}$ is positive semidefinite and $\gamma + \rho_2 > 0$, the matrix inverse $[2\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V} + (\gamma+\rho_2)\boldsymbol{I}_k]^{-1}$ is well defined for all $i \in \{1, \ldots, n\}$. Computing the optimal solution to (4.12) requires computing n different $k \times k$ matrix inverses (where in general $k \ll \min\{m, n\}$). Computing a single $k \times k$ matrix inverse requires $O(k^3)$ time and forming the matrix product $\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}$ requires $O(k^2 m)$ time for a given i. Thus, the complexity of computing the optimal solution for a single column is $O(k^3 + k^2 m)$. Notice that each column of $\bar{\boldsymbol{U}}$ can be computed independently of the other columns. We leverage this observation by developing a multi-threaded implementation of the algorithm presented in this section. Letting w denote the number of compute threads available, computing the optimal solution $\bar{\boldsymbol{U}}$ of (4.12) requires $O\left(\frac{k^3 n + k^2 mn}{\min\{w,n\}}\right)$ time (the term $\min\{w, n\}$ in the denominator reflects that fact that increasing the number of available compute threads beyond the number of columns

of $\bar{\boldsymbol{U}}$ does not yield additional reduction in compute complexity).

4.5.2 Subproblem in $\boldsymbol{V}$

Now, suppose we fix variables $\boldsymbol{U}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{V}$. Eliminating terms that do not depend on $\boldsymbol{V}$, the resulting subproblem is given by

$$\min_{\boldsymbol{V} \in \mathbb{R}^{m \times k}} \sum_{(i,j) \in \Omega} ((\boldsymbol{U}\boldsymbol{V}^T)_{ij} - A_{ij})^2 + \frac{\gamma}{2} \|\boldsymbol{V}\|_F^2. \tag{4.14}$$

We now have the following result:

Proposition 40 *The optimal solution $\bar{\boldsymbol{V}}$ for (4.14) is given by*

$$\bar{V}_{j,\star} = [2\boldsymbol{U}^T \boldsymbol{W}_j \boldsymbol{U} + \gamma \boldsymbol{I}_k]^{-1} [2\boldsymbol{U}^T \boldsymbol{W}_j A_{\star,j}], \tag{4.15}$$

for each $j \in \{1, \ldots, m\}$ where each $\boldsymbol{W}_j \in \mathbb{R}^{n \times n}$ is a diagonal matrix satisfying $(\boldsymbol{W}_j)_{ii} = 1$ if $(i, j) \in \Omega$, otherwise $(\boldsymbol{W}_j)_{ii} = 0$. Here, the column vector $\bar{V}_{j,\star} \in \mathbb{R}^k$ denotes the j^{th} row of $\bar{\boldsymbol{V}}$ while the column vector $A_{\star,j} \in \mathbb{R}^n$ denotes the j^{th} column of $\boldsymbol{A}$ where the unknown entries of $\boldsymbol{A}$ are taken to be 0.

Proof This proof follows the proof of Proposition 39. Let $f(\boldsymbol{V})$ denote the objective function of (4.14). With $\{\boldsymbol{W}_j\}_{j=1}^m$ defined as in Proposition 40, observe that we can write $f(\boldsymbol{V})$ as

$$\begin{aligned} f(\boldsymbol{V}) &= \sum_{j=1}^m \|\boldsymbol{W}_j(\boldsymbol{U}V_{j,\star} - A_{\star,j})\|_2^2 + \frac{\gamma}{2} \sum_{j=1}^m \|V_{j,\star}\|_2^2 \\ &= \sum_{j=1}^m \left[\|\boldsymbol{W}_j(\boldsymbol{U}V_{j,\star} - A_{\star,j})\|_2^2 + \frac{\gamma}{2} \|V_{j,\star}\|_2^2 \right] = \sum_{j=1}^m g_j(\boldsymbol{V}), \end{aligned}$$

where we define $g_j(\boldsymbol{V}) = \|\boldsymbol{W}_j(\boldsymbol{U}V_{j,\star} - A_{\star,j})\|_2^2 + \frac{\gamma}{2}\|V_{j,\star}\|_2^2$. Thus, we have shown that $f(\boldsymbol{V})$ is separable over the rows of the matrix $\boldsymbol{V}$. Each function $g_j(\boldsymbol{V})$ is a (strongly) convex quadratic. Thus, we can minimize $g_j(\boldsymbol{V})$ by setting its gradient to 0. For any fixed row

$j \in \{1, \ldots, m\}$, we can differentiate and obtain

$$\nabla_{\boldsymbol{V}_{j,\star}} g_j(\boldsymbol{V}) = 2\boldsymbol{U}^T \boldsymbol{W}_j (\boldsymbol{U} V_{j,\star} - A_{\star,j}) + \gamma V_{j,\star}.$$

By equating the gradient $\nabla_{\boldsymbol{V}_{j,\star}} g_j(\boldsymbol{V})$ to 0 and rearranging, we obtain that the optimal vector $\bar{U}_{i,\star}$ is given by (4.15). This completes the proof. ■

Observe that since the matrix $\boldsymbol{U}^T \boldsymbol{W}_j \boldsymbol{U}$ is positive semidefinite and $\gamma > 0$, the matrix inverse $[2\boldsymbol{U}^T \boldsymbol{W}_j \boldsymbol{U} + \gamma \boldsymbol{I}_k]^{-1}$ is well defined for all $j \in \{1, \ldots, m\}$. Computing the optimal solution to (4.14) requires computing m different $k \times k$ matrix inverses. Forming the matrix product $\boldsymbol{U}^T \boldsymbol{W}_j \boldsymbol{U}$ requires $O(k^2 n)$ time for a given j. Thus, the complexity of computing the optimal solution for a single column is $O(k^3 + k^2 n)$. Notice that, similarly to the solution of (4.12), each column of $\bar{\boldsymbol{V}}$ can be computed independently of the other columns. As before, we leverage this observation in our multi-threaded implementation of the algorithm presented in this section. Letting w denote the number of compute threads available, computing the optimal solution $\bar{\boldsymbol{V}}$ of (4.14) requires $O\left(\frac{k^3 m + k^2 m n}{\min\{w, m\}}\right)$ time.

The optimal solution $\bar{\boldsymbol{V}}$ to (4.14) reveals that the Frobenius norm regularization term on $\boldsymbol{V}$ in (4.8) (which emerges from the nuclear norm regularization term on $\boldsymbol{X}$ in (4.1)) has computational benefits. Indeed, if we had $\gamma = 0$, it is possible that the matrix $\boldsymbol{U}^T \boldsymbol{W}_j \boldsymbol{U}$ be singular at certain iterates of our ADMM algorithm, in which case the corresponding matrix inverse would be undefined. This observation is in keeping with several recent works in the statistics, machine learning and operations research literatures where the presence of a regularization penalty in the objective function yields improved out of sample performance as well as benefits in computational tractability (see for example [15, 18, 20, 24, 27]).

4.5.3 Subproblem in $\boldsymbol{P}$

Now, suppose we fix variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{P}$. Eliminating terms that do not depend on $\boldsymbol{P}$, the resulting subproblem is given by

$$\min_{\boldsymbol{P} \in \mathbb{S}^n_+} -\lambda \mathrm{Tr}(\boldsymbol{Y}^T \boldsymbol{P} \boldsymbol{Y}) - \mathrm{Tr}(\boldsymbol{\Phi}^T \boldsymbol{P} \boldsymbol{Z}) + \frac{\rho_1}{2} \|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{P}^2 = \boldsymbol{P},\ \mathrm{Tr}(\boldsymbol{P}) \leq k. \tag{4.16}$$

We now have the following result:

Proposition 41 *Let $\boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T$ be a rank k truncated singular value decomposition for the matrix given by:*

$$\Big(\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T + \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{Z}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T)\Big),$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{k \times k}, \boldsymbol{M} \in \mathbb{R}^{n \times k}, \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}_k$. The optimal solution $\bar{\boldsymbol{P}}$ for (4.16) is given by $\bar{\boldsymbol{P}} = \boldsymbol{M}\boldsymbol{M}^T$.

Proof Let $f(\boldsymbol{P})$ denote the objective function of (4.16). Observe that for any $\boldsymbol{P}$ that is feasible to (4.16), we can write $f(\boldsymbol{P})$ as:

$$\begin{aligned} f(\boldsymbol{P}) &= -\lambda \mathrm{Tr}(\boldsymbol{Y}^T \boldsymbol{P} \boldsymbol{Y}) - \mathrm{Tr}(\boldsymbol{\Phi}^T \boldsymbol{P} \boldsymbol{Z}) + \frac{\rho_1}{2} \|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}\|_F^2 \\ &= -\lambda \mathrm{Tr}(\boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{P}) - \mathrm{Tr}(\boldsymbol{Z}\boldsymbol{\Phi}^T \boldsymbol{P}) + \frac{\rho_1}{2} \mathrm{Tr}(\boldsymbol{Z}\boldsymbol{Z}^T(\boldsymbol{I}_n - \boldsymbol{P})) \\ &= \frac{\rho_1}{2}\mathrm{Tr}(\boldsymbol{Z}\boldsymbol{Z}^T) - \langle \lambda \boldsymbol{Y}\boldsymbol{Y}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T, \boldsymbol{P} \rangle. \end{aligned}$$

Thus, it is immediately clear that a solution will be optimal to (4.16) if and only if it is optimal to the problem given by:

$$\max_{\boldsymbol{P} \in \mathbb{S}^n_+} \langle \boldsymbol{C}, \boldsymbol{P} \rangle \quad \text{s.t.} \quad \boldsymbol{P}^2 = \boldsymbol{P},\ \mathrm{Tr}(\boldsymbol{P}) \leq k, \tag{4.17}$$

where we define the matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ as $\boldsymbol{C} = \lambda \boldsymbol{Y}\boldsymbol{Y}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T$. Let $\bar{\boldsymbol{C}} = \frac{1}{2}(\boldsymbol{C} + \boldsymbol{C}^T)$ denote the symmetric part of $\boldsymbol{C}$. Observe that for any symmetric matrix $\boldsymbol{P}$, we can consider

$\langle \bar{\boldsymbol{C}}, \boldsymbol{P} \rangle$ in place of $\langle \boldsymbol{C}, \boldsymbol{P} \rangle$ since we have

$$\begin{aligned}\langle \boldsymbol{C}, \boldsymbol{P} \rangle &= \sum_{i=1}^{n} \sum_{j=1}^{n} P_{ij} C_{ij} = \sum_{i=1}^{n} P_{ii} C_{ii} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P_{ij} (C_{ij} + C_{ji}) \\ &= \sum_{i=1}^{n} P_{ii} \bar{C}_{ii} + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P_{ij} \bar{C}_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} P_{ij} \bar{C}_{ij} = \langle \bar{\boldsymbol{C}}, \boldsymbol{P} \rangle.\end{aligned}$$

Let $\bar{\boldsymbol{C}} = \boldsymbol{M} \boldsymbol{\Sigma} \boldsymbol{M}^T$ be a full singular value decomposition of $\bar{\boldsymbol{C}}$ with $\boldsymbol{M}, \boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$, $\boldsymbol{M}^T \boldsymbol{M} = \boldsymbol{M} \boldsymbol{M}^T = \boldsymbol{I}_n$. The matrix $\boldsymbol{\Sigma}$ is the diagonal matrix of (ordered) singular values of $\bar{\boldsymbol{C}}$ and we let σ_i denote the i^{th} singular value. Any feasible matrix $\boldsymbol{P}$ to (4.17) can be written as $\boldsymbol{P} = \boldsymbol{L} \boldsymbol{L}^T$ where $\boldsymbol{L} \in \mathbb{R}^{n \times k}, \boldsymbol{L}^T \boldsymbol{L} = \boldsymbol{I}_k$. Thus, for any $\boldsymbol{P}$ feasible to (4.17) we express the objective value as:

$$\langle \bar{\boldsymbol{C}}, \boldsymbol{P} \rangle = \operatorname{Tr}(\boldsymbol{M} \boldsymbol{\Sigma} \boldsymbol{M}^T \boldsymbol{L} \boldsymbol{L}^T) = \operatorname{Tr}(\boldsymbol{\Sigma} (\boldsymbol{M}^T \boldsymbol{L} \boldsymbol{L}^T \boldsymbol{M})) = \sum_{i=1}^{n} \sigma_i \| (\boldsymbol{M}^T \boldsymbol{L})_{i,\star} \|_2^2.$$

Let $\boldsymbol{N} = \boldsymbol{M}^T \boldsymbol{L} \in \mathbb{R}^{n \times k}$. Note that we have $\boldsymbol{N}^T \boldsymbol{N} = \boldsymbol{L}^T \boldsymbol{M} \boldsymbol{M}^T \boldsymbol{L} = \boldsymbol{L}^T \boldsymbol{L} = \boldsymbol{I}_k$, which implies that the columns of $\boldsymbol{N}$ are orthonormal. This immediately implies that we have $N_{i,\star}^T N_{i,\star} = \| (\boldsymbol{M}^T \boldsymbol{L})_{i,\star} \|_2^2 \leq 1$. Moreover, we have

$$\sum_{i=1}^{n} \| (\boldsymbol{M}^T \boldsymbol{L})_{i,\star} \|_2^2 = \sum_{i=1}^{n} N_{i,\star}^T N_{i,\star} = \sum_{i=1}^{n} \sum_{j=1}^{k} N_{ij}^2 = \sum_{j=1}^{k} \sum_{i=1}^{n} N_{ij}^2 = \sum_{j=1}^{k} 1 = k.$$

We can therefore upper bound the optimal objective value of (4.17) as

$$\langle \bar{\boldsymbol{C}}, \boldsymbol{P} \rangle = \sum_{i=1}^{n} \sigma_i \| (\boldsymbol{M}^T \boldsymbol{L})_{i,\star} \|_2^2 \leq \sum_{i=1}^{k} \sigma_i.$$

To conclude the proof, notice that by taking $\bar{\boldsymbol{P}} = \bar{\boldsymbol{M}} \bar{\boldsymbol{M}}^T$ where $\bar{\boldsymbol{M}} \in \mathbb{R}^{n \times k}$ is the matrix that consists of the first k columns of $\boldsymbol{M}$ we can achieve the upper bound on (4.17):

$$\langle \bar{\boldsymbol{C}}, \bar{\boldsymbol{P}} \rangle = \operatorname{Tr}(\boldsymbol{M} \boldsymbol{\Sigma} \boldsymbol{M}^T \bar{\boldsymbol{M}} \bar{\boldsymbol{M}}^T) = \operatorname{Tr}(\bar{\boldsymbol{M}}^T \boldsymbol{M} \boldsymbol{\Sigma} \boldsymbol{M}^T \bar{\boldsymbol{M}}) = \sum_{i=1}^{k} \sigma_i.$$

■

To compute the optimal solution of (4.16), we need to compute a rank k singular value decomposition of the matrix $\bar{\boldsymbol{C}} = \left(\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{\rho_1}{2}\boldsymbol{Z}\boldsymbol{Z}^T + \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{Z}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T)\right)$ which requires $O(kn^2)$ time since $\bar{\boldsymbol{C}} \in \mathbb{R}^{n\times n}$. Moreover, explicitly forming the matrix $\bar{\boldsymbol{C}}$ in memory from its constituent matrices $\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{\Phi}$ requires $O(n^2(d+k))$ operations. Thus, naively computing the optimal solution to (4.16) has complexity $O(n^2(d+k))$ where the bottleneck operation from a complexity standpoint is explicitly forming the matrix $\bar{\boldsymbol{C}}$.

Fortunately, it is possible to compute the optimal solution to (4.16) more efficiently. Observe that we can equivalently express the matrix $\bar{\boldsymbol{C}}$ as $\bar{\boldsymbol{C}} = \boldsymbol{F}_1\boldsymbol{F}_2^T$ where $\boldsymbol{F}_1, \boldsymbol{F}_2 \in \mathbb{R}^{n\times(d+3k)}$ are defined as

$$\boldsymbol{F}_1 = \left(\sqrt{\lambda}\boldsymbol{Y} \;\middle|\; \sqrt{\tfrac{\rho_1}{2}}\boldsymbol{Z} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{\Phi} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{Z}\right),$$
$$\boldsymbol{F}_2 = \left(\sqrt{\lambda}\boldsymbol{Y} \;\middle|\; \sqrt{\tfrac{\rho_1}{2}}\boldsymbol{Z} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{Z} \;\middle|\; \sqrt{\tfrac{1}{2}}\boldsymbol{\Phi}\right).$$

Computing a truncated singular value decomposition requires only computing repeated matrix vector products. Therefore, rather than explicitly forming the matrix $\bar{\boldsymbol{C}}$ in memory at a cost of $O(n^2(d+k))$ operations, in our implementation we design a custom matrix class where matrix vector products between $\bar{\boldsymbol{C}}$ and arbitrary vectors $\boldsymbol{x} \in \mathbb{R}^n$ are computed by first evaluating the matrix vector product $\boldsymbol{\nu} = \boldsymbol{F}_2^T\boldsymbol{x}$ and subsequently evaluating the matrix vector product $\bar{\boldsymbol{C}}\boldsymbol{x} = \boldsymbol{F}_1\boldsymbol{\nu}$. In so doing, we can evaluate matrix vector products $\bar{\boldsymbol{C}}\boldsymbol{x}$ in $O(n(d+k))$ time rather than $O(n^2)$ time (in general, we will have $d+k \ll n$). Computing a truncated singular value decomposition of $\bar{\boldsymbol{C}}$ with this methodology of evaluating matrix vector products requires only $O(k^2n + knd)$ operations. Thus, our custom matrix class implementation avoids needing to explicitly for $\bar{\boldsymbol{C}}$ in memory and allows the optimal solution to (4.16) to be computed in $O(k^2n + knd)$ time.

4.5.4 Subproblem in $\boldsymbol{Z}$

Now, suppose we fix variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ and seek to minimize $\mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ over $\boldsymbol{Z}$. Eliminating terms that do not depend on $\boldsymbol{Z}$, the resulting subproblem is given by

$$\min_{\boldsymbol{Z} \in \mathbb{R}^{n \times k}} \operatorname{Tr}(\boldsymbol{\Phi}^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}) + \operatorname{Tr}(\boldsymbol{\Psi}^T\boldsymbol{Z}) + \frac{\rho_1}{2}\|(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z}\|_F^2 + \frac{\rho_2}{2}\|\boldsymbol{Z} - \boldsymbol{U}\|_F^2. \tag{4.18}$$

We now have the following result:

Proposition 42 *The optimal solution $\bar{\boldsymbol{Z}}$ for (4.18) is given by*

$$\begin{aligned} \bar{\boldsymbol{Z}} &= \frac{1}{\rho_1 + \rho_2}\Big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\Big)\Big(\rho_2\boldsymbol{U} - (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} - \boldsymbol{\Psi}\Big) \\ &= \frac{1}{\rho_1 + \rho_2}\Big(\rho_2\boldsymbol{U} - \boldsymbol{\Phi} + \boldsymbol{P}\boldsymbol{\Phi} - \boldsymbol{\Psi} + \rho_1\boldsymbol{P}\boldsymbol{U} - \frac{\rho_1}{\rho_2}\boldsymbol{P}\boldsymbol{\Psi}\Big). \end{aligned} \tag{4.19}$$

Proof Let $f(\boldsymbol{Z})$ denote the objective function of (4.18). The function $f(\boldsymbol{Z})$ is a convex quadratic, thus it can be minimized by setting its gradient to 0. Differentiating $f(\boldsymbol{Z})$, we obtain:

$$\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z}) = (\boldsymbol{I}_n - \boldsymbol{P})^T\boldsymbol{\Phi} + \boldsymbol{\Psi} + \rho_1(\boldsymbol{I}_n - \boldsymbol{P})^T(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z} + \rho_2(\boldsymbol{Z} - \boldsymbol{U}).$$

Moreover, for any matrix $\boldsymbol{P}$ for which the augmented Lagrangian (4.11) takes finite value, we will have $\boldsymbol{P} \in \mathcal{P}_k$ which implies that $\boldsymbol{P}^T = \boldsymbol{P}$ and $\boldsymbol{P}^2 = \boldsymbol{P}$. We can therefore simplify $\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z})$ as:

$$\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z}) = (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} + \boldsymbol{\Psi} + \rho_1(\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{Z} + \rho_2(\boldsymbol{Z} - \boldsymbol{U}).$$

By equating the gradient $\nabla_{\boldsymbol{Z}} f(\boldsymbol{Z})$ to 0 and rearranging, we obtain that the optimal matrix $\bar{\boldsymbol{Z}}$ is given by:

$$\bar{\boldsymbol{Z}} = \Big(\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2\boldsymbol{I}_n\Big)^{-1}\Big(\rho_2\boldsymbol{U} - (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} - \boldsymbol{\Psi}\Big)$$

To conclude the proof, it remains to show that $\big(\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2\boldsymbol{I}_n\big)^{-1} = \frac{1}{\rho_1+\rho_2}\big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\big)$. Let $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T$ where $\boldsymbol{M} \in \mathbb{R}^{n\times k}, \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}_k$. Such a matrix $\boldsymbol{M}$ is guaranteed to exist for any $\boldsymbol{P} \in \mathcal{P}_k$. We have

$$\begin{aligned}
\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2\boldsymbol{I}_n &= \rho_1(\boldsymbol{I}_n - \boldsymbol{M}\boldsymbol{M}^T) + \rho_2\boldsymbol{I}_n \\
&= (\rho_1+\rho_2)\boldsymbol{I}_n + \boldsymbol{M}(-\rho_1\boldsymbol{I}_n)\boldsymbol{M}^T \\
&= \frac{1}{\rho_1+\rho_2}\boldsymbol{I}_n - \frac{1}{(\rho_1+\rho_2)^2}\boldsymbol{M}\left(\frac{1}{\rho_1+\rho_2}\boldsymbol{M}^T\boldsymbol{M} - \frac{1}{\rho_1}\boldsymbol{I}_k\right)^{-1}\boldsymbol{M}^T \\
&= \frac{1}{\rho_1+\rho_2}\boldsymbol{I}_n - \frac{1}{(\rho_1+\rho_2)^2}\boldsymbol{M}\left(\frac{-\rho_1(\rho_1+\rho_2)}{\rho_2}\boldsymbol{I}_k\right)\boldsymbol{M}^T \\
&= \frac{1}{\rho_1+\rho_2}\Big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\Big),
\end{aligned}$$

where the third equality follows from the Woodbury matrix inversion lemma (see [117], Section 3.2.2). As a sanity check, one can verify that the product of $\big(\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2\boldsymbol{I}_n\big)$ and $\frac{1}{\rho_1+\rho_2}\big(\boldsymbol{I}_n + \frac{\rho_1}{\rho_2}\boldsymbol{P}\big)$ is indeed the n dimensional identity matrix. ■

Evaluating the optimal solution to (4.18) requires only matrix-matrix multiplications. Computing the products of $\boldsymbol{P}\boldsymbol{\Phi}, \boldsymbol{P}\boldsymbol{U}, \boldsymbol{P}\boldsymbol{\Psi}$ in the definition of $\bar{\boldsymbol{Z}}$ from (4.19) requires $O(kn^2)$ operations. Thus, the naive cost of forming $\bar{\boldsymbol{Z}}$ is $O(kn^2)$. However, notice that if we had a factored representation of the matrix $\boldsymbol{P}$ as $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T$ with $\boldsymbol{M} \in \mathbb{R}^{n\times k}$, for any matrix $\boldsymbol{R} \in \mathbb{R}^{n\times k}$ we could compute matrix-matrix products $\boldsymbol{P}\boldsymbol{R}$ by first computing $\boldsymbol{S} = \boldsymbol{M}^T\boldsymbol{R}$ and thereafter computing $\boldsymbol{P}\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}$ for a total complexity of $O(k^2n)$. One might object that this ignores the time required to compute such a matrix $\boldsymbol{M}$. However, observe that in computing a matrix $\boldsymbol{P}$ that is optimal to (4.16), we in fact must already generate such a matrix $\boldsymbol{M}$ (see proposition 41). In fact, in our implementation we never explicitly form a $n \times n$ matrix $\boldsymbol{P}$ as it suffices to only store a copy of its low rank factorization matrix $\boldsymbol{M}$. Thus, the optimal solution to (4.18) can be evaluated in $O(k^2n)$ time.

4.5.5 An ADMM Algorithm

Having illustrated that the partial minimization of the Lagrangian (4.11) across each of the primal variables (Problems (4.12), (4.14), (4.16), (4.18)) can be solved efficiently, we can now present the overall approach Algorithm 5.

Algorithm 5: Mixed-Projection ADMM

Data: $n, m, k \in \mathbb{Z}^+, \Omega \subset [n] \times [m], \{A_{ij}\}_{(i,j)\in\Omega}, \lambda, \gamma \in \mathbb{R}^+$. Tolerance parameter $\epsilon > 0$. Maximum iteration parameter $T \in \mathbb{Z}^+$

Result: $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}})$ that is feasible to (4.8).

$(\boldsymbol{U}_0, \boldsymbol{P}_0, \boldsymbol{V}_0, \boldsymbol{Z}_0) \leftarrow (\boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}}, \boldsymbol{L}\boldsymbol{L}^T, \boldsymbol{R}\boldsymbol{\Sigma}^{\frac{1}{2}}, \boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}})$ where $\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}$ is a rank k truncated SVD of $\boldsymbol{A}$ and missing entries are filled in with 0;

$(\boldsymbol{\Phi}_0, \boldsymbol{\Psi}_0) \leftarrow (\mathbf{1}_{n\times k}, \mathbf{1}_{n\times k})$;

$t \leftarrow 0$;

while $t < T$ *and* $\max\{\|(\boldsymbol{I}_n - \boldsymbol{P}_t)\boldsymbol{Z}_t\|_F^2, \|\boldsymbol{Z}_t - \boldsymbol{U}_t\|_F^2\} > \epsilon$ **do**

- $(\boldsymbol{U}_{t+1}, \boldsymbol{P}_{t+1}) \leftarrow \arg\min_{\boldsymbol{U},\boldsymbol{P}} \mathcal{L}^A(\boldsymbol{U}, \boldsymbol{V}_t, \boldsymbol{P}, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$;
- $(\boldsymbol{V}_{t+1}, \boldsymbol{Z}_{t+1}) \leftarrow \arg\min_{\boldsymbol{V},\boldsymbol{Z}} \mathcal{L}^A(\boldsymbol{U}_{t+1}, \boldsymbol{V}, \boldsymbol{P}_{t+1}, \boldsymbol{Z}, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$;
- $\boldsymbol{\Phi}_{t+1} \leftarrow \boldsymbol{\Phi}_t + \rho_1(\boldsymbol{I} - \boldsymbol{P}_{t+1})\boldsymbol{Z}_{t+1}$;
- $\boldsymbol{\Psi}_{t+1} \leftarrow \boldsymbol{\Psi}_t + \rho_2(\boldsymbol{Z}_{t+1} - \boldsymbol{U}_{t+1})$;
- $t \leftarrow t + 1$;

end

return $(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t)$

We initialize primal iterates $\boldsymbol{U}_0 = \boldsymbol{Z}_0 = \boldsymbol{L}\boldsymbol{\Sigma}^{\frac{1}{2}}, \boldsymbol{P}_0 = \boldsymbol{L}\boldsymbol{L}^T, \boldsymbol{V}_0 = \boldsymbol{R}\boldsymbol{\Sigma}^{\frac{1}{2}}$ where $\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}$ denotes a rank k truncated singular value decomposition of $\boldsymbol{A}$ (the missing entries of $\boldsymbol{A}$ are filled in with 0s) and we initialize dual iterates $\boldsymbol{\Phi}_0 = \boldsymbol{\Psi}_0 = \mathbf{1}_{n\times k}$. Observe that the subproblems (4.12) and (4.16) can be solved simultaneously. Similarly, the subprobems (4.14) and (4.18) can be solved simultaneously. At each iteration of Algorithm 5, we first update the iterates $\boldsymbol{U}_{t+1}, \boldsymbol{P}_{t+1}$ by solving problems (4.12) and (4.16) with $(\boldsymbol{V}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$ fixed. Next, we update the iterates $\boldsymbol{V}_{t+1}, \boldsymbol{Z}_{t+1}$ by solving problems (4.14) and (4.18) with $(\boldsymbol{U}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)$ fixed. Finally, we update the dual iterates $\boldsymbol{\Phi}, \boldsymbol{\Psi}$ by taking a gradient ascent step. The gradients of the augmented Lagrangian (4.11) with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are given by the primal residuals $(\boldsymbol{I}_n - \boldsymbol{P}_{t+1})\boldsymbol{Z}_{t+1}$ and $\boldsymbol{Z}_{t+1} - \boldsymbol{U}_{t+1}$ respectively. We use ρ_1 and ρ_2 respectively as the step size. We proceed until the squared norm of each primal residual

is below a numerical tolerance parameter ϵ or until we reach an input maximum number of iterations T. We know have the following result:

Proposition 43 *Assume that the number of compute threads w is less than $\min\{n, m\}$. The per iteration complexity of Algorithm 5 is $O\big(k^2 n + knd + \frac{k^3(n+m)+k^2nm}{w}\big)$.*

Proof The result follows from the complexity analysis of problems (4.12), (4.14), (4.16) and (4.18). ■

Having presented Algorithm 5 in extensive detail, it is natural to consider what types of guarantees can be made on the final output solution $(\boldsymbol{U}_T, \boldsymbol{V}_T, \boldsymbol{P}_T)$. We explore this in the following theorem:

Theorem 44 *Let $\{(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$ denote a sequence generated by Algorithm 5 (assuming we allow Algorithm 5 to iterate indefinitely). Suppose the dual variable sequence $\{(\boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$ is bounded and satisfies*

$$\sum_{t=0}^{\infty}(\|\boldsymbol{\Phi}_{t+1} - \boldsymbol{\Phi}_t\|_F^2 + \|\boldsymbol{\Psi}_{t+1} - \boldsymbol{\Psi}_t\|_F^2) < \infty. \tag{4.20}$$

Let $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ denote any accumulation point of $\{(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$. If the set of k leading eigenvectors of the matrix $[\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$ is the same as the set of k leading eigenvectors of the matrix $\bar{\boldsymbol{Z}}\bar{\boldsymbol{Z}}^T$, then $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ satisfies the first order optimality conditions for (4.10).

Proof We leverage a proof technique similar to the technique used to establish Theorem 2.1 from [147]. Note that from (4.20), we immediately have $\boldsymbol{\Phi}_{t+1} - \boldsymbol{\Phi}_t \to 0$ and $\boldsymbol{\Psi}_{t+1} - \boldsymbol{\Psi}_t \to 0$. We will first show that we also have $\boldsymbol{U}_{t+1} - \boldsymbol{U}_t \to 0, \boldsymbol{V}_{t+1} - \boldsymbol{V}_t \to 0$ and $\boldsymbol{Z}_{t+1} - \boldsymbol{Z}_t \to 0$.

Let $\mathcal{L}^A(\boldsymbol{U}; \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ denote the augmented Lagrangian (4.11) viewed as a function of $\boldsymbol{U}$. Notice that $\mathcal{L}^A(\boldsymbol{U}; \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}) - \frac{\gamma+\rho_2}{2}\|\boldsymbol{U}\|_F^2$ is a convex function, which implies that $\mathcal{L}^A(\boldsymbol{U}; \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ is a strongly convex function of $\boldsymbol{U}$ with parameter $\gamma + \rho_2$.

Similarly, $\mathcal{L}^A(\boldsymbol{V}; \boldsymbol{U}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ is a strongly convex function of $\boldsymbol{V}$ with parameter γ and $\mathcal{L}^A(\boldsymbol{Z}; \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ is a strongly convex function of $\boldsymbol{Z}$ with parameter ρ_2. By strong convexity, we know that for any matrices $\boldsymbol{U}, \boldsymbol{\Delta U}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ we have

$$\mathcal{L}^A(\boldsymbol{U} + \boldsymbol{\Delta U}) - \mathcal{L}^A(\boldsymbol{U}) \geq \langle \nabla_{\boldsymbol{U}} \mathcal{L}^A(\boldsymbol{U}), \boldsymbol{\Delta U} \rangle + (\gamma + \rho_2) \|\boldsymbol{\Delta U}\|_F^2, \tag{4.21}$$

where the shorthand notation $\mathcal{L}^A(\boldsymbol{U})$ is understood to denote the augmented Lagrangian (4.11) viewed as a function only of $\boldsymbol{U}$ with the other variables held fixed. Letting $\boldsymbol{U} = \boldsymbol{U}_{t+1}, \boldsymbol{\Delta U} = \boldsymbol{U}_t - \boldsymbol{U}_{t+1}$, and noting that $\langle \nabla_{\boldsymbol{U}} \mathcal{L}^A(\boldsymbol{U}_{t+1}), \boldsymbol{\Delta U} \rangle \geq 0$ since $\boldsymbol{U}_{t+1}$ minimizes $\mathcal{L}^A(\boldsymbol{U})$ at iteration t, from (4.21) we have $\mathcal{L}^A(\boldsymbol{U}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}) \geq (\gamma + \rho_2) \|\boldsymbol{U}_t - \boldsymbol{U}_{t+1}\|_F^2$. Similarly, we have $\mathcal{L}^A(\boldsymbol{V}_t) - \mathcal{L}^A(\boldsymbol{V}_{t+1}) \geq \gamma \|\boldsymbol{V}_t - \boldsymbol{V}_{t+1}\|_F^2$ and $\mathcal{L}^A(\boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{Z}_{t+1}) \geq \rho_2 \|\boldsymbol{Z}_t - \boldsymbol{Z}_{t+1}\|_F^2$. Moreover, since $\boldsymbol{P}_{t+1}$ minimizes $\mathcal{L}^A(\boldsymbol{P})$ at iteration t, we have $\mathcal{L}^A(\boldsymbol{P}_t) - \mathcal{L}^A(\boldsymbol{P}_{t+1}) \geq 0$. Observe that we can express the difference in the value of the augmented Lagrangian between iteration t and iteration $t+1$ as

$$\begin{aligned} &\mathcal{L}^A(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}, \boldsymbol{V}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{Z}_{t+1}) = \\ &\qquad \mathcal{L}^A(\boldsymbol{U}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}) + \mathcal{L}^A(\boldsymbol{V}_t) - \mathcal{L}^A(\boldsymbol{V}_{t+1}) + \mathcal{L}^A(\boldsymbol{P}_t) - \mathcal{L}^A(\boldsymbol{P}_{t+1}) \\ &\qquad + \mathcal{L}^A(\boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{Z}_{t+1}) + \mathcal{L}^A(\boldsymbol{\Phi}_t) - \mathcal{L}^A(\boldsymbol{\Phi}_{t+1}) + \mathcal{L}^A(\boldsymbol{\Psi}_t) - \mathcal{L}^A(\boldsymbol{\Psi}_{t+1}). \end{aligned} \tag{4.22}$$

Recognizing that we have $\mathcal{L}^A(\boldsymbol{\Phi}_t) - \mathcal{L}^A(\boldsymbol{\Phi}_{t+1}) = -\rho_1 \|\boldsymbol{\Phi}_t - \boldsymbol{\Phi}_{t+1}\|_F^2, \mathcal{L}^A(\boldsymbol{\Psi}_t) - \mathcal{L}^A(\boldsymbol{\Psi}_{t+1}) = -\rho_2 \|\boldsymbol{\Psi}_t - \boldsymbol{\Psi}_{t+1}\|_F^2$, (4.22) implies that

$$\begin{aligned} &\mathcal{L}^A(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t) - \mathcal{L}^A(\boldsymbol{U}_{t+1}, \boldsymbol{V}_{t+1}, \boldsymbol{P}_{t+1}, \boldsymbol{Z}_{t+1}) \geq \\ &\qquad (\gamma + \rho_2) \|\boldsymbol{U}_t - \boldsymbol{U}_{t+1}\|_F^2 + \gamma \|\boldsymbol{V}_t - \boldsymbol{V}_{t+1}\|_F^2 + \rho_2 \|\boldsymbol{Z}_t - \boldsymbol{Z}_{t+1}\|_F^2 \\ &\qquad - \rho_1 \|\boldsymbol{\Phi}_t - \boldsymbol{\Phi}_{t+1}\|_F^2 - \rho_2 \|\boldsymbol{\Psi}_t - \boldsymbol{\Psi}_{t+1}\|_F^2. \end{aligned} \tag{4.23}$$

We claim that the augmented Lagrangian is bounded from below. To see this, note that $\mathcal{L}^A$

can equivalently be written as

$$\begin{aligned}\mathcal{L}^A(\boldsymbol{U},\boldsymbol{V},\boldsymbol{P},\boldsymbol{Z},\boldsymbol{\Phi},\boldsymbol{\Psi}) = &\sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij}-A_{ij})^2+\lambda\|(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Y}\|_F^2\\ &+\frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2+\|\boldsymbol{V}\|_F^2)+\mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P})+\frac{\rho_1}{2}\|(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Z}+\frac{\boldsymbol{\Phi}}{\rho_1}\|_F^2\\ &+\frac{\rho_2}{2}\|\boldsymbol{Z}-\boldsymbol{U}+\frac{\boldsymbol{\Psi}}{\rho_2}\|_F^2-\frac{1}{2\rho_1}\|\boldsymbol{\Phi}\|_F^2+\frac{1}{2\rho_2}\|\boldsymbol{\Psi}\|_F^2,\end{aligned} \tag{4.24}$$

and recall that by assumption the dual variables $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are bounded. Thus, the boundedness of $\mathcal{L}^A$ coupled with summing (4.23) over t implies that

$$\begin{aligned}&\sum_{t=0}^{\infty}c_1(\|\boldsymbol{U}_t-\boldsymbol{U}_{t+1}\|_F^2+\|\boldsymbol{V}_t-\boldsymbol{V}_{t+1}\|_F^2+\|\boldsymbol{Z}_t-\boldsymbol{Z}_{t+1}\|_F^2)\\ &-\sum_{t=0}^{\infty}c_2(\|\boldsymbol{\Phi}_t-\boldsymbol{\Phi}_{t+1}\|_F^2+\|\boldsymbol{\Psi}_t-\boldsymbol{\Psi}_{t+1}\|_F^2)<\infty,\end{aligned} \tag{4.25}$$

where $c_1=\min\{\gamma,\rho_2\}$ and $c_2=\max\{\rho_1,\rho_2\}$. By assumption, the second term of (4.25) is finite which implies that the first term must also be finite. This immediately implies that $\boldsymbol{U}_{t+1}-\boldsymbol{U}_t\to 0, \boldsymbol{V}_{t+1}-\boldsymbol{V}_t\to 0$ and $\boldsymbol{Z}_{t+1}-\boldsymbol{Z}_t\to 0$ as desired.

We are now ready to prove the main result of the theorem. The (unaugmented) Lagrangian $\mathcal{L}$ for (4.10) is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{U},\boldsymbol{V},\boldsymbol{P},&\boldsymbol{Z},\boldsymbol{\Phi},\boldsymbol{\Psi}) = \sum_{(i,j)\in\Omega}((\boldsymbol{U}\boldsymbol{V}^T)_{ij}-A_{ij})^2+\lambda\mathrm{Tr}\big(\boldsymbol{Y}^T(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Y}\big)\\ &+\frac{\gamma}{2}(\|\boldsymbol{U}\|_F^2+\|\boldsymbol{V}\|_F^2)+\mathbb{I}_{\mathcal{P}_k}(\boldsymbol{P})+\mathrm{Tr}(\boldsymbol{\Phi}^T(\boldsymbol{I}_n-\boldsymbol{P})\boldsymbol{Z})+\mathrm{Tr}(\boldsymbol{\Psi}^T(\boldsymbol{Z}-\boldsymbol{U})).\end{aligned} \tag{4.26}$$

The corresponding first order optimality conditions can be expressed as

$$[2\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V} + \gamma\boldsymbol{I}_k]U_{i,\star} = 2\boldsymbol{V}^T\boldsymbol{W}_iA_{i,\star} + \Psi_{i,\star} \quad i \in [n], \tag{4.27a}$$

$$[2\boldsymbol{U}^T\boldsymbol{W}_j\boldsymbol{U} + \gamma\boldsymbol{I}_k]V_{j,\star} = 2\boldsymbol{U}^T\boldsymbol{W}_jA_{\star,j} \quad j \in [m], \tag{4.27b}$$

$$\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T \text{where } \boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T \text{ is a rank } k \text{ SVD of } \lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{Z}^T + \boldsymbol{Z}\boldsymbol{\Phi}^T), \tag{4.27c}$$

$$\boldsymbol{\Phi} + \boldsymbol{\Psi} = \boldsymbol{P}\boldsymbol{\Phi}, \tag{4.27d}$$

$$\boldsymbol{Z} = \boldsymbol{P}\boldsymbol{Z}, \tag{4.27e}$$

$$\boldsymbol{Z} = \boldsymbol{U}, \tag{4.27f}$$

where the diagonal matrices $\boldsymbol{W}_i, \boldsymbol{W}_j$ are defined as in Propositions 39 and 40 respectively. Let $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ denote any limit point of $\{(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{P}_t, \boldsymbol{Z}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Psi}_t)\}$. Recalling the Algorithm 5 updates for $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, the conditions $\boldsymbol{\Phi}_{t+1} - \boldsymbol{\Phi}_t \to 0$ and $\boldsymbol{\Psi}_{t+1} - \boldsymbol{\Psi}_t \to 0$ imply that (4.27e) and (4.27f) hold at $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$. Moreover, when (4.27f) holds the Algorithm 5 update for $\boldsymbol{U}$ given by Proposition 39 reduces to (4.27a) while the update for $\boldsymbol{V}$ given by Proposition 40 enforces (4.27b). From the proof of Propsition (42), we know that Algorithm 5 updates $\boldsymbol{Z}$ to satisfy the following

$$\Big(\rho_1(\boldsymbol{I}_n - \boldsymbol{P}) + \rho_2\boldsymbol{I}_n\Big)\boldsymbol{Z} = \Big(\rho_2\boldsymbol{U} - (\boldsymbol{I}_n - \boldsymbol{P})\boldsymbol{\Phi} - \boldsymbol{\Psi}\Big). \tag{4.28}$$

Since (4.27e) and (4.27f) hold, (4.28) immediately implies (4.27d) is satisfied by $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$. It remains to verify that $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ satisfies (4.27c). By Proposition 41, we know that we have $\bar{\boldsymbol{P}} = \boldsymbol{L}\boldsymbol{L}^T$ where $\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T$ is a rank k truncated SVD of the matrix $[\lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{\rho_1}{2}\bar{\boldsymbol{Z}}\bar{\boldsymbol{Z}}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$. If the set of k leading eigenvectors of the matrix $\bar{\boldsymbol{Z}}\bar{\boldsymbol{Z}}^T$ is the same as the set of k leading eigenvectors of $[\lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$, it follows immediately that $\boldsymbol{L}\Sigma'\boldsymbol{L}^T$ will be a rank k SVD of $[\lambda\boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$ for some diagonal matrix Σ'. Thus, in this setting, $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ satisfies (4.27c). this completes the proof.

■

In words, Theorem 44 states that if the sequence of dual variable iterates produced by Algorithm 5 is bounded and the primal residuals converge to zero quickly enough (specifically, it is required that the norm of successive dual variable differences is summable), then any accumulation point $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{Z}}, \bar{\boldsymbol{\Phi}}, \bar{\boldsymbol{\Psi}})$ of the sequence of iterates produced by Algorithm 5 satisfies the first order optimality conditions of (4.10) if the rank k approximation of the matrix $[\lambda \boldsymbol{Y}\boldsymbol{Y}^T + \frac{1}{2}(\bar{\boldsymbol{\Phi}}\bar{\boldsymbol{Z}}^T + \bar{\boldsymbol{Z}}\bar{\boldsymbol{\Phi}}^T)]$ shares the same column space as the matrix $\bar{\boldsymbol{Z}}$. We note that this condition can only be verified upon termination of Algorithm 5 since it depends on the algorithm output in addition to the problem data. Accordingly, Theorem 44 provides an a posteriori convergence result. We note that this condition was satisfied in our numerical experiments.

4.6 Computational Results

We evaluate the performance of Algorithm 5 implemented in Julia 1.7.3. Throughout, we fix $\rho_1 = \rho_2 = 10$, set the maximum number of iterations $T = 20$ and set the number of compute threads $w = 24$. Note that given the novelty of Problem (4.1), there are no pre-existing specialized methods to benchmark against. Accordingly, we compare the performance of Algorithm 5 against well studied methods for the very closely related MC problem as well as a highly performant generic method for low rank matrix optimization problems. The MC methods we consider are Fast-Impute [26], Soft-Impute [105] and Iterative-SVD [136] which we introduced formally in Section 4.2.1. We utilize the implementation of Fast-Impute made publicly available by [26] while we use the implementation of Soft-Impute and Iterative-SVD from the python package fancyimpute 0.7.0 [125]. The matrix optimization method we consider is ScaledGD (scaled gradient descent) [133] which we introduced formally in Section 4.2.2 and implement ourselves. We perform experiments using both synthetic data and real world data on MIT's Supercloud Cluster [123], which hosts Intel Xeon Platinum 8260 processors. To bridge the gap between theory and practice, we have made our code freely available

on GitHub at https://github.com/NicholasJohnson2020/LearningLowRankMatrices.

To evaluate the performance of Algorithm 5, Fast-Impute, Soft-Impute, Iterative-SVD and ScaledGD on synthetic data, we consider the objective value achieved by a returned solution in (4.1), the ℓ_2 reconstruction error between a returned solution and the ground truth, the coefficient of determination (R^2) when the returned solution is used as a predictor for the side information, the numerical rank of a returned solution and the execution time of each algorithm. Explicitly, let $\hat{\boldsymbol{X}} \in \mathbb{R}^{n \times m}$ denote the solution returned by a given method (where we define $\hat{\boldsymbol{X}} = \hat{\boldsymbol{U}}\hat{\boldsymbol{V}}^T$ if the method outputs low rank factors $\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}$) and let $\boldsymbol{A}^{true} \in \mathbb{R}^{n \times m}$ denote the ground truth matrix. We define the the ℓ_2 reconstruction error of $\hat{\boldsymbol{X}}$ as

$$ERR_{\ell_2}(\hat{\boldsymbol{X}}) = \frac{\|\hat{\boldsymbol{X}} - \boldsymbol{A}^{true}\|_F^2}{\|\boldsymbol{A}^{true}\|_F^2}.$$

We compute the numerical rank of $\hat{\boldsymbol{X}}$ by calling the default rank function from the Julia LinearAlgebra package. We aim to answer the following questions:

1. How does the performance of Algorithm 5 compare to existing methods such as Fast-Impute, Soft-Impute, Iterative-SVD and ScaledGD on synthetic and real world data?

2. How is the performance of Algorithm 5 affected by the number of rows n, the number of columns m, the dimension of the side information d and the underlying rank k of the ground truth?

3. Empirically, which subproblem solution update is the computational bottleneck of Algorithm 5?

4.6.1 Synthetic Data Generation

To generate synthetic data, we specify a number of rows $n \in \mathbb{Z}_+$, a number of columns $m \in \mathbb{Z}_+$, a desired rank $k \in \mathbb{Z}_+$ with $k < \min\{n, m\}$, the dimension of the side information $d \in \mathbb{Z}_+$, a fraction of missing values $\alpha \in (0, 1)$ and a noise parameter $\sigma \in \mathbb{R}_+$ that controls

the signal to noise ratio. We sample matrices $\boldsymbol{U} \in \mathbb{R}^{n\times k}, \boldsymbol{V} \in \mathbb{R}^{m\times k}, \boldsymbol{\beta} \in \mathbb{R}^{m\times d}$ by drawing each entry $U_{ij}, V_{ij}, \beta_{ij}$ independently from the uniform distribution on the interval $[0, 1]$. Furthermore, we sample a noise matrix $\boldsymbol{N} \in \mathbb{R}^{n\times d}$ by drawing each entry N_{ij} independently from the univariate normal distribution with mean 0 and variance σ^2. We let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T$ and we let $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\beta} + \boldsymbol{N}$. Lastly, we sample $\lfloor \alpha \cdot n \cdot m \rfloor$ indices uniformly at random from the collection $\mathcal{I} = \{(i, j) : 1 \leq i \leq n, 1 \leq j \leq m\}$ to be the set of missing indices, which we denote by Γ. The set of revealed entries can then be defined as $\Omega = \mathcal{I} \setminus \Gamma$. We fix $\alpha = 0.9, \sigma = 2$ throughout our experiments and report numerical results for various different combinations of (n, m, d, k).

4.6.2 Sensitivity to Row Dimension

We present a comparison of Algorithm 5 with ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD as we vary the number of rows n. In these experiments, we fixed $m = 100, k = 5$, and $d = 150$ across all trials. We varied $n \in \{100, 200, 400, 800, 1000, 2000, 5000, 10000\}$ and we performed 20 trials for each value of n. For ScaledGD, we set the step size to be $\eta = \frac{1}{10\sigma_1(\boldsymbol{A})}$ where $\sigma_1(\boldsymbol{A})$ denotes the largest singular value of the input matrix $\boldsymbol{A}$ where we fill the unobserved entries with the value 0. Letting $f(\boldsymbol{U}_t, \boldsymbol{V}_t)$ denote the objective value achieved after iteration t of ScaledGD, we terminate ScaledGD when either $t > 1000$ or $\frac{f(\boldsymbol{U}_{t-1},\boldsymbol{V}_{t-1}) - f(\boldsymbol{U}_t,\boldsymbol{V}_t)}{f(\boldsymbol{U}_{t-1},\boldsymbol{V}_{t-1})} < 10^{-3}$. In words, we terminate ScaledGD after 1000 iterations or after the relative objective value improvement between two iterations is less than 0.1%.

We report the objective value, ℓ_2 reconstruction error, side information R^2 and execution time for Algorithm 5, Fast-Impute, Soft-Impute and Iterative-SVD in Figure 4.1. We additionally report the objective value, reconstruction error, side information R^2 and execution time for ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and Iterative-SVD in Tables 4.1, 4.2, 4.3 and 4.4 of Section 4.8. In Figure 4.2, we plot the average cumulative time spent solving subproblems (4.12), (4.14), (4.16), (4.18) during the execution of Algorithm 5 versus n. Our main findings from this set of experiments are:

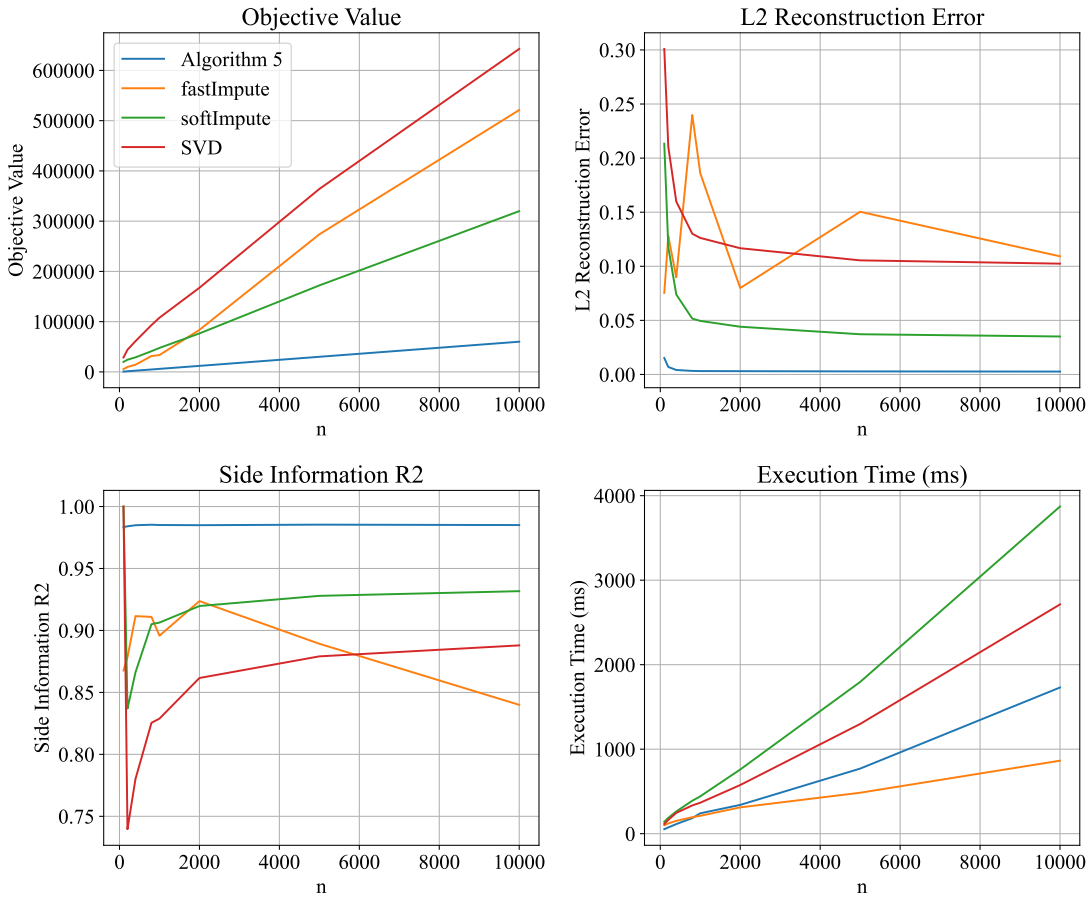

Figure 4.1: Objective value (top left), ℓ_2 reconstruction error (top right), side information R^2 (bottom left) and execution time (bottom right) versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

1. Algorithm 5 systematically produces higher quality solutions than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table 4.1), sometimes achieving an objective value that is an order of magnitude superior than the next best method. On average, Algorithm 5 outputs a solution whose objective value is 86% lesser than the objective value achieved by the best performing alternative method (Fast-Impute). We remind the reader that Fast-Impute, Soft-Impute and Iterative-SVD are methods designed for the generic MC problem and are not custom built to solve (4.1) so it should not come as a surprise that Algorithm 5 significantly outperforms these 3 methods in terms of objective value. ScaledGD however has explicit knowledge of the objective

function of (4.1) along with its gradient, yet surprisingly produces the weakest average objective value across these experiments. We note that we use the default hyperparameters for ScaledGD recommended by the authors of this method [133]. We observe that the objective value achieved by all methods increases linearly as the number of rows n increases.

2. In terms of ℓ_2 reconstruction error, Algorithm 5 again systematically produces solutions that are of higher quality than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table 4.2), often achieving an error that is an order of magnitude superior than the next best method. On average, Algorithm 5 outputs a solution whose ℓ_2 reconstruction error is 92% lesser than the reconstruction error achieved by the best performing alternative method (Soft-Impute in all but one parameter configuration). This is especially noteworthy since Algorithm 5 is not designed explicitly with reconstruction error minimization as the objective, unlike Fast-Impute and Soft-Impute, and suggests that the side information $\boldsymbol{Y}$ is instrumental in recovering high quality low rank estimates of the partially observed data matrix.

3. With the exception of the experiments for which $n = 100$, Algorithm 5 always produced solutions that achieved a superior R^2 value when used as a predictor for the side information compared to Fast-Impute, Soft-Impute, Iterative-SVD and ScaledGD. This observation supports the notion that encouraging that the reconstructed matrix is a good linear predictor of the side information is instrumental in producing higher quality estimates of the partially observed matrix.

4. The runtime of Algorithm 5 is competitive with that of the other methods. The runtime of Algorithm 5 is less than of Soft-Impute and Iterative-SVD but greater than that of Fast-Impute. For experiments with $n \leq 2000$, Table 4.4 illustrates that ScaledGD was the method with the fastest execution time (however as previously mentioned the returned solutions were of low quality). The runtime of Algorithm 5, Fast-Impute,

Soft-Impute and iterate SVD appear to grow linearly with n.

5. Figure 4.2 illustrates that the computation of the solution for (4.12) is the computational bottleneck in the execution of Algorithm 5 in this set of experiments, followed next by the computation of the solution for (4.16). Empirically, we observe that the solution time of (4.12), (4.14), (4.16) and (4.18) appear to scale linearly with the number of rows n. This observation is consistent with the computational complexities derived for each subproblem of Algorithm 5 in Section 4.5.

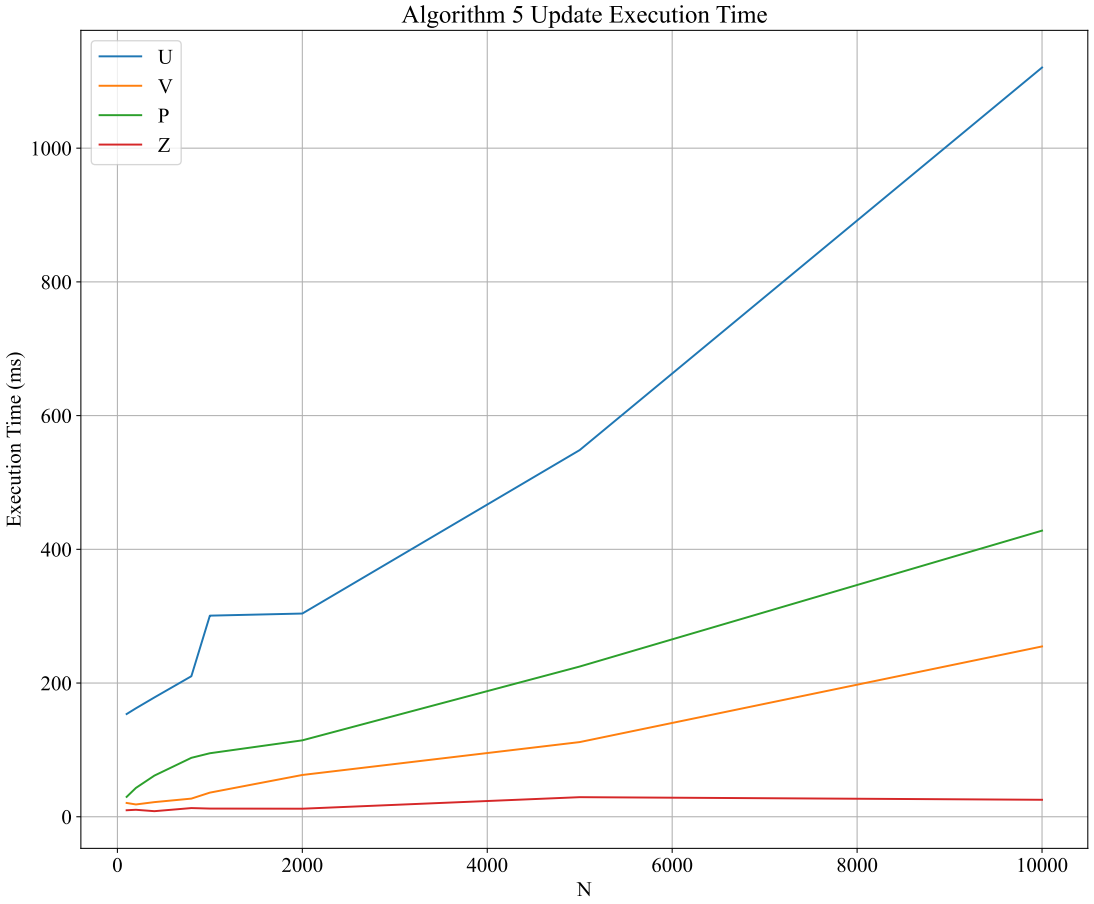

Figure 4.2: Cumulative time spent solving each subproblem of Algorithm 5 versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

4.6.3 Sensitivity to Column Dimension

Here, we present a comparison of Algorithm 5 with ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD as we vary the number of columns m. We fixed $n = 1000, k = 5$, and $d = 150$ across all trials. We varied $m \in \{100, 200, 400, 800, 1000, 2000, 5000, 10000\}$ and we performed 20 trials for each value of m.

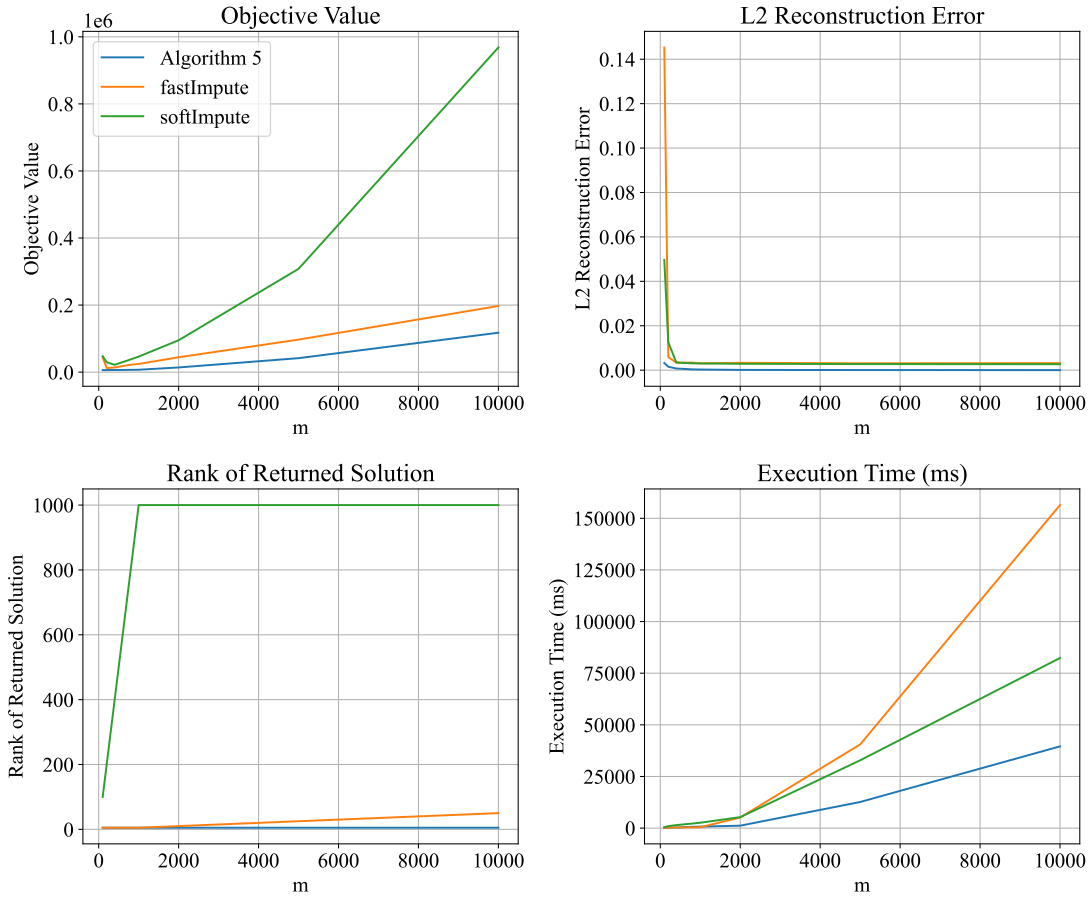

Figure 4.3: Objective value (top left), ℓ_2 reconstruction error (top right), fitted rank (bottom left) and execution time (bottom right) versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

We report the objective value, ℓ_2 reconstruction error, fitted rank and execution time for Algorithm 5, Fast-Impute and Soft-Impute in Figure 4.3. We additionally report the objective value, reconstruction error and execution time for ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and Iterative-SVD in Tables 4.5, 4.6 and 4.7 of Section 4.8. In Figure

4.4, we plot the average cumulative time spent solving subproblems (4.12), (4.14), (4.16), (4.18) during the execution of Algorithm 5 versus m. Our main findings from this set of experiments are a follows:

1. Here again, Algorithm 5 systematically produces higher quality solutions than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table 4.5). On average, Algorithm 5 outputs a solution whose objective value is 62% lesser than the objective value achieved by the best performing alternative method (Fast-Impute). Here again, ScaledGD produces the weakest average objective value across these experiments. We observe that the objective value achieved by each method appears to increase super-linearly as the number of columns m increases.

2. In terms of ℓ_2 reconstruction error, Algorithm 5 again systematically produces solutions that are of higher quality than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table 4.6), often achieving an error that is an order of magnitude superior than the next best method. On average, Algorithm 5 outputs a solution whose ℓ_2 reconstruction error is 90% lesser than the reconstruction error achieved by the best performing alternative method (Soft-Impute in all but one parameter configuration).

3. We observe that the fitted rank of the solutions returned by Algorithm 5, ScaledGD and Fast-Impute always matched the specified target rank as would be expected, but surprisingly the solutions returned by Soft-Impute and Iterative-SVD were always of full rank despite the fact that these methods were provided with the target rank explicitly. This is potentially due to a numerical issues in the computation of the rank due to presence of extremely small singular values.

4. The runtime of Algorithm 5 exhibits the most favorable scaling behavior among the methods tested in these experiments. For instances with $m \geq 2000$, Table 4.7 shows that Algorithm 5 had the fastest runtime. For instances with $m < 2000$, ScaledGD

had the fastest execution time but produced low quality solutions. The runtime of all methods tested grow super-linearly with m.

5. Figure 4.4 illustrates that the computation of the solution for (4.12) and (4.14) are the computational bottlenecks in the execution of Algorithm 5 in this set of experiments while the computation of the solution for (4.18) and (4.16) appear to be a constant function of m. This observation is consistent with the complexity analysis performed for each subproblem of Algorithm 5 in Section 4.5. Indeed, this analysis indicated that solve times for (4.18) and (4.16) are independent of m while the solve times for (4.12) and (4.14) scale linearly with m when the number of threads w satisfies $w < m$.

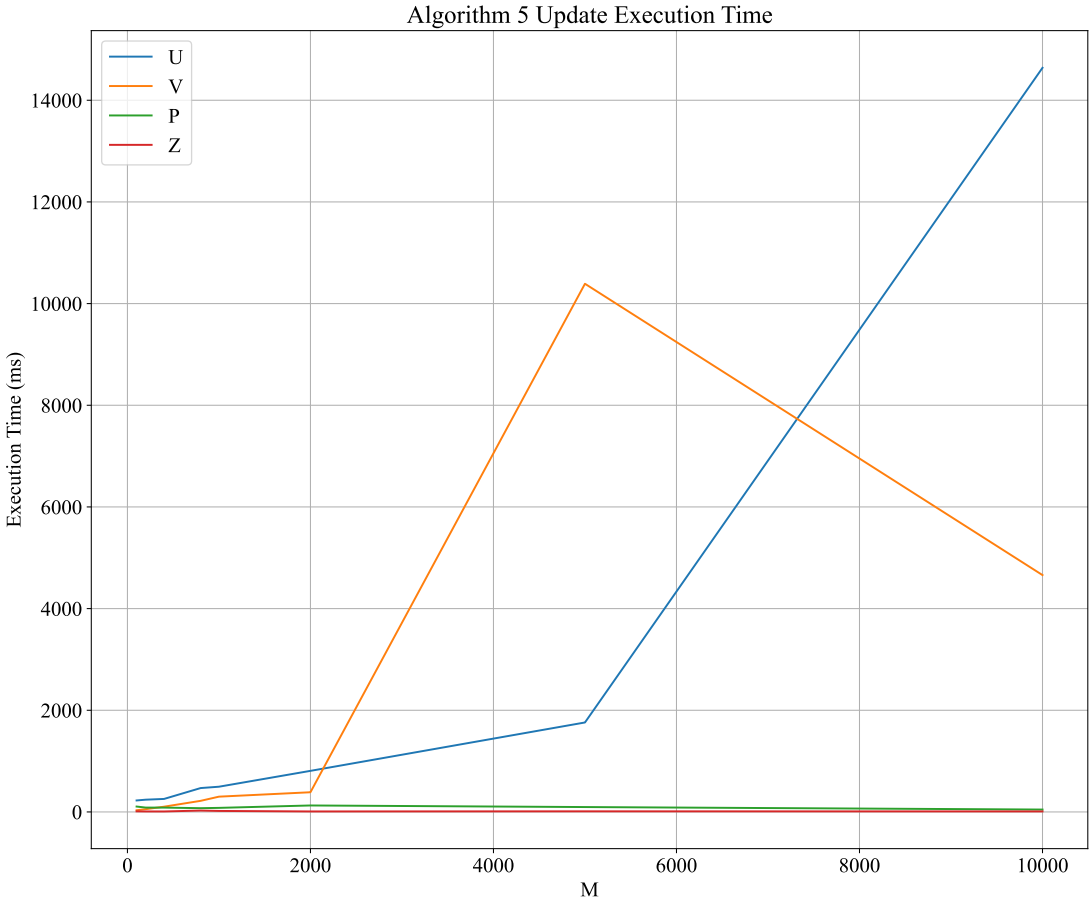

Figure 4.4: Cumulative time spent solving each subproblem of Algorithm 5 versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

4.6.4 Sensitivity to Side Information Dimension

We present a comparison of Algorithm 5 with ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD as we vary the dimension of the side information d. In these experiments, we fixed $n = 1000, m = 100$ and $k = 5$ across all trials. We varied $d \in \{10, 50, 100, 150, 200, 250, 500, 1000\}$ and we performed 20 trials for each value of d.

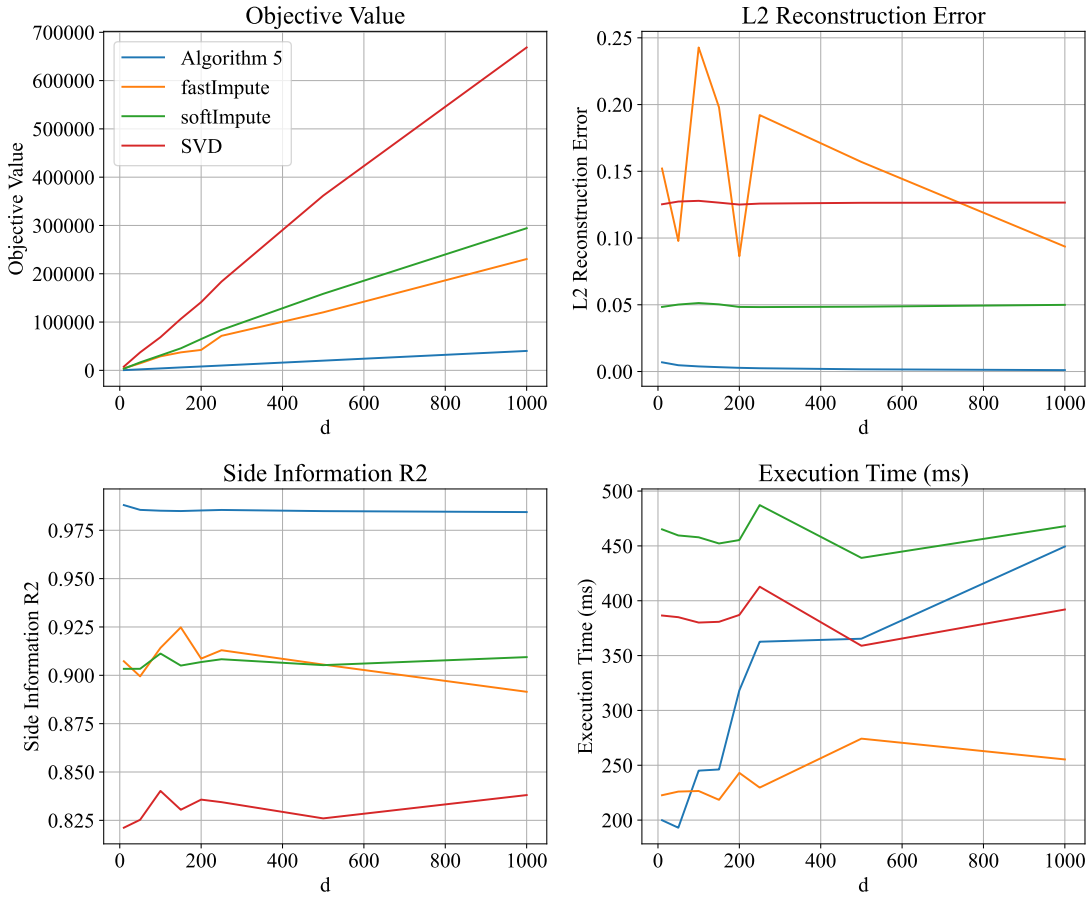

Figure 4.5: Objective value (top left), ℓ_2 reconstruction error (top right), side information R^2 (bottom left) and execution time (bottom right) versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

We report the objective value, ℓ_2 reconstruction error, side information R^2 and execution time for Algorithm 5, Fast-Impute, Soft-Impute and Iterative-SVD in Figure 4.5. We additionally report the objective value, reconstruction error, side information R^2 and execution time for ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and Iterative-SVD in Tables 4.8,

4.9, 4.10 and 4.11 of Section 4.8. In Figure 4.6, we plot the average cumulative time spent solving subproblems (4.12), (4.14), (4.16), (4.18) during the execution of Algorithm 5 versus d. Our main findings from this set of experiments are:

1. Just as in Sections 4.6.2 and 4.6.3, Algorithm 5 systematically produces higher quality solutions than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table 4.8). On average, Algorithm 5 outputs a solution whose objective value is 85% lesser than the objective value achieved by the best performing alternative method (Fast-Impute). ScaledGD produces the weakest average objective value across these experiments. The objective value achieved by each method appears to increase linearly as the dimension d of the side information increases.

2. In terms of ℓ_2 reconstruction error, just as in Sections 4.6.2 and 4.6.3 Algorithm 5 produces solutions that are of higher quality than ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD (see Table 4.9), often achieving an error that is an order of magnitude superior than the next best method. On average, Algorithm 5 outputs a solution whose ℓ_2 reconstruction error is 93% lesser than the reconstruction error achieved by the best performing alternative method (Soft-Impute). The performance of Algorithm 5 improves as d increases, consistent with the intuition that recovering the partially observed matrix $\boldsymbol{A}$ becomes easier as more side information becomes available.

3. Here, Algorithm 5 always produced solutions that achieved a strictly greater R^2 value when used as a predictor for the side information compared to the benchmark methods. The R^2 achieved by each method is roughly constant as the value of d increases.

4. The runtime of Algorithm 5 is competitive with that of the other methods. The runtime of Algorithm 5 is less than of Soft-Impute and Iterative-SVD but greater than that of Fast-Impute. Table 4.11 illustrates that ScaledGD was the fastest performing method, however its solutions were of the lowest quality. The runtime of Algorithm 5 and ScaledGD grows with d while Fast-Impute, Soft-Impute and iterate SVD are constant

with d which should be expected as these methods do not act on the side information matrix $\boldsymbol{Y}$.

5. Figure 4.6 illustrates that the computation of the solution for (4.16) is the computational bottleneck in the execution of Algorithm 5 in this set of experiments, followed next by the computation of the solution to (4.12). The solution times for (4.12), (4.14) and (4.18) appear constant as a function of d. This is consistent with the complexity analysis from Section 4.5 which found that the solve time for (4.16) is linear in d while the solve time for the 3 other subproblems are independent of d.

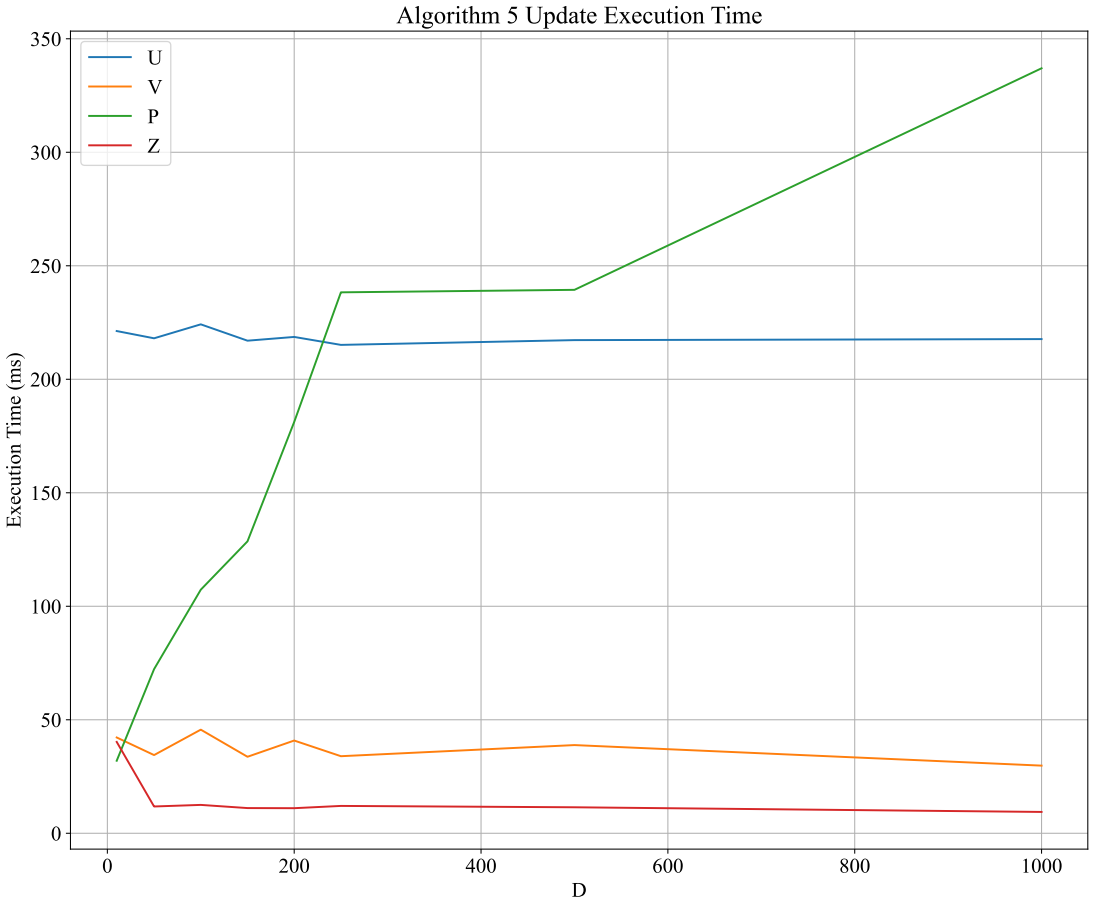

Figure 4.6: Cumulative time spent solving each subproblem of Algorithm 5 versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

4.6.5 Sensitivity to Target Rank

We present a comparison of Algorithm 5 with ScaledGD, Fast-Impute, Soft-Impute and Iterative-SVD as we vary the rank of the underlying matrix k. In these experiments, we fixed $n = 1000, m = 100$ and $d = 150$ across all trials. We varied $k \in \{5, 10, 15, 20, 25, 30, 35, 40\}$ and we performed 20 trials for each value of d.

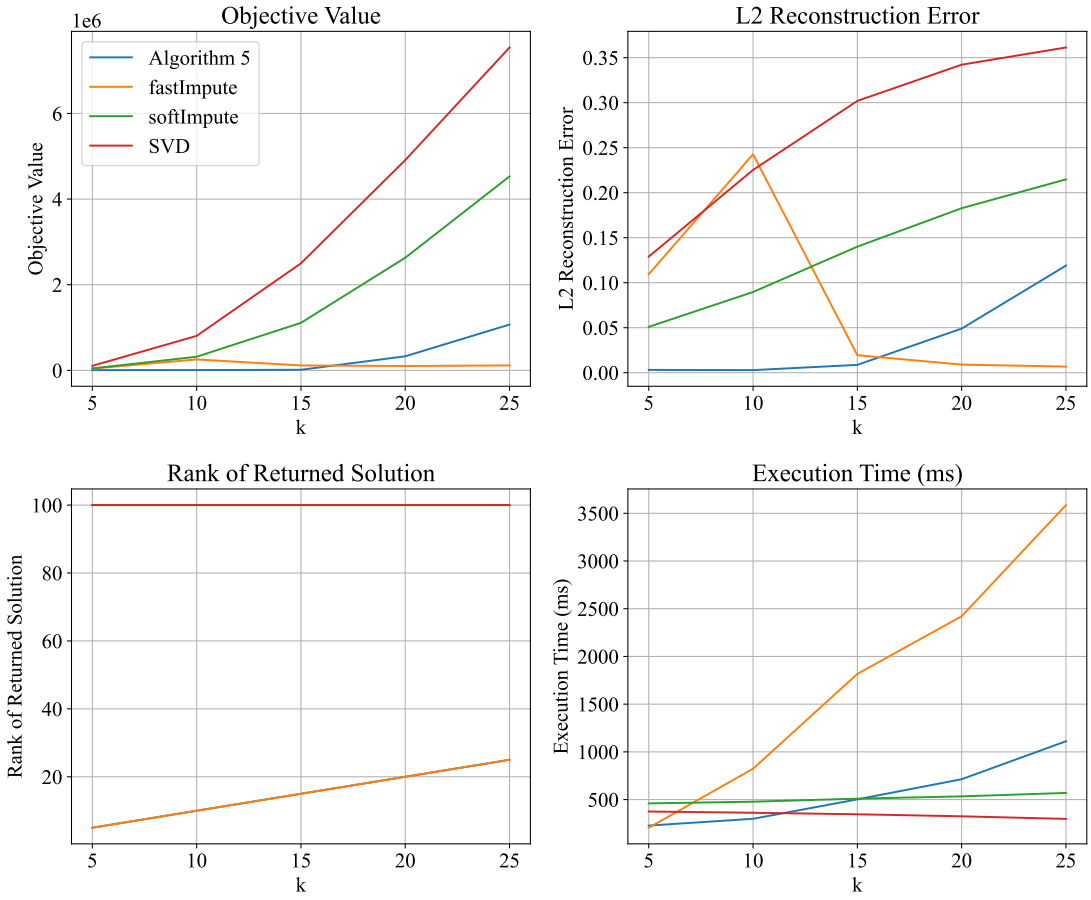

Figure 4.7: Objective value (top left), ℓ_2 reconstruction error (top right), fitted rank (bottom left) and execution time (bottom right) versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

We report the objective value, ℓ_2 reconstruction error, fitted rank and execution time for Algorithm 5, Fast-Impute, Soft-Impute and Iterative-SVD in Figure 4.7. We additionally report the objective value, reconstruction error and execution time for ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and Iterative-SVD in Tables 4.12, 4.13 and 4.14 of Section 4.8.

In Figure 4.8, we plot the average cumulative time spent solving subproblems (4.12), (4.14), (4.16), (4.18) during the execution of Algorithm 5 versus k. Our main findings from this set of experiments are as follows:

1. Unlike in Sections 4.6.2, 4.6.3 and 4.6.4, Algorithm 5 only produced higher quality solutions than all benchmark methods in 3 out of 8 of the tested parameter configurations where $k \leq 15$ (see Table 4.12). Fast-Impute was the best performing method in 3 configurations and Soft-Impute was best in the remaining 2 configurations. ScaledGD produces the weakest average objective value across these experiments.

2. In terms of ℓ_2 reconstruction error, Algorithm 5 again produced higher quality solutions than all benchmark methods in 3 out of 8 of the tested parameter configurations where $k \leq 15$ (see Table 4.13). Fast-Impute produced solutions achieving the lowest error in the other 5 parameter configurations.

3. The fitted rank of the solutions returned by Algorithm 5, ScaledGD and Fast-Impute always matched the specified target rank, but the solutions returned by Soft-Impute and Iterative-SVD were always of full rank despite the fact that these methods were provided with the target rank explicitly.

4. The runtime of Algorithm 5 is competitive with that of the other methods. Table 4.14 illustrates that ScaledGD was the fastest performing method, however its solutions were of the lowest quality. The runtime of Algorithm 5 is most competitive with Soft-Impute and Iterative-SVD for small values of k. Though Fast-Impute is the best performing method in terms of objective in 3 out of 8 configurations and the best in terms of ℓ_2 error in 5 out of 8 configurations, it takes on average 3 times as long as Algorithm 5 to execute.

5. Figure 4.8 illustrates that the computation of the solution for (4.12) is the computational bottleneck in the execution of Algorithm 5 in this set of experiments, followed next by the computation of the solution to (4.14) and (4.18).

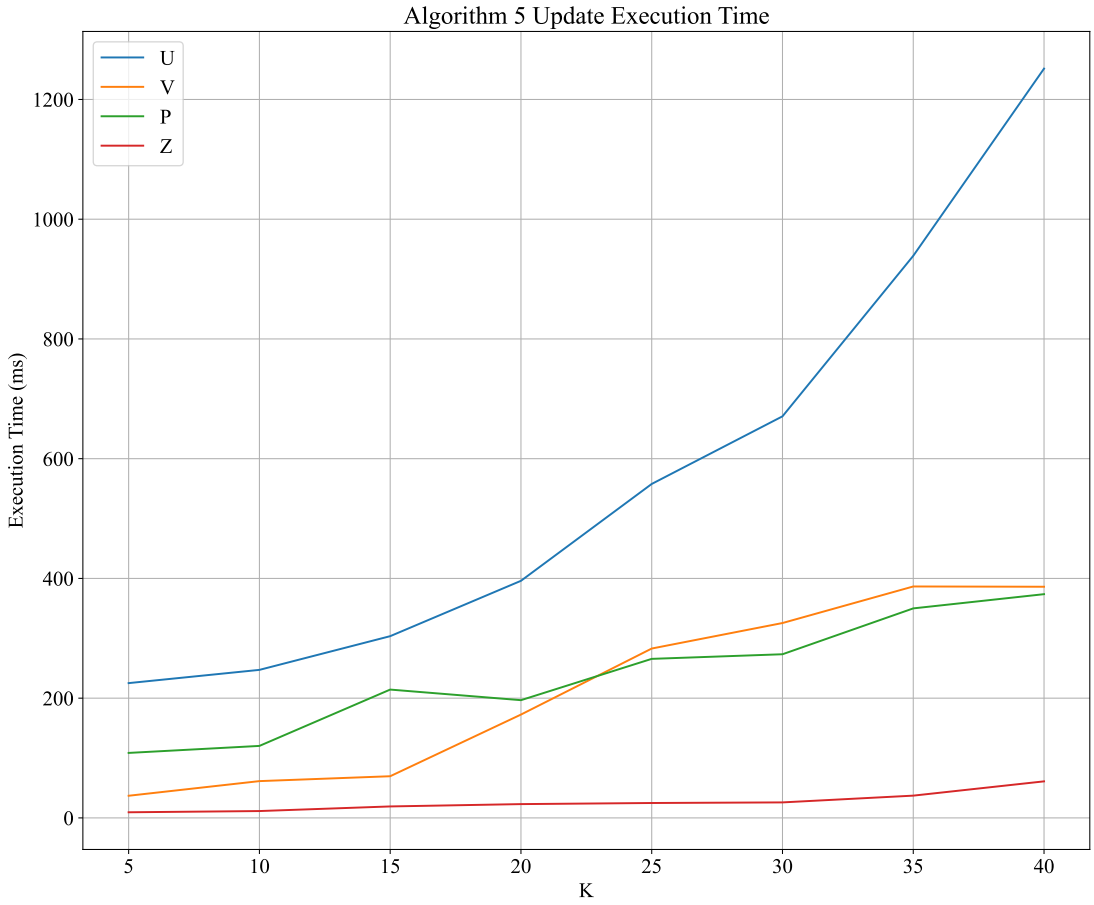

Figure 4.8: Cumulative time spent solving each subproblem of Algorithm 5 versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

4.6.6 Real World Data Experiments

We seek to answer the following question: how does the performance of Algorithm 5 compare to Fast-Impute on real world data? We consider the Netflix Prize Dataset augmented with features from the TMDB Database.

The Netflix Prize Dataset consists of greater than 10 million user ratings of movies spread across more than 450000 users and 17000 movies. To prepare data for our experiment, we first pull the following numerical features from the TMDB database:

1. Total Budget;

2. Revenue;

3. Popularity;

4. Average Vote;

5. Vote Count;

6. Total Runtime.

Note that many movies did not have all 6 features available from TMDB. We constructed two datasets for our experimentation. In Dataset 1, we restricted the dataset to movies that had all 6 features present and to users who had given at least 5 ratings across those movies. After performing this filtering, we were left with $n = 3430$ movies and $m = 467364$ users. In Dataset 2, we considered the 4 most frequent features (popularity, average vote, vote count, total runtime) and restricted the dataset to movies that had all 4 of these features present and to users who had given at least 5 ratings to any of these movies. After performing this filtering, we were left with $n = 10574$ movies and $m = 470706$ users. For each dataset, we conducted experiments for values of the target rank k in the set $k \in \{3, 4, 5, 6, 7, 8, 9, 10\}$. For each value of k, we conducted 5 trials where a given trial consisted of randomly withholding 20% of the data as test data, estimating a low rank matrix on the 80% training data and evaluating the out of sample ℓ_2 reconstruction error on the withheld data.

We report the in sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time for Algorithm 5 and Fast-Impute in addition to the average cumulative time spent solving subproblems (4.12), (4.14), (4.16), (4.18) during the execution of Algorithm 5 versus k on Dataset 1 and Dataset 2 in Figures 4.9 and 4.10 respectively. We additionally report the in sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time for Algorithm 5 and Fast-Impute on Dataset 1 and Dataset 2 in Tables 4.15 and 4.16 of Appendix 4.8 respectively. We report only results for Fast-Impute as a benchmark because Soft-Impute, Iterative-SVD and ScaledGD failed to terminate after a 20 hour

time limit across all experiments involving Dataset 1 and Dataset 2. Fast-Impute failed to terminate after a 20 hour time limit across all experiments involving Dataset 2 and across experiments involving Dataset 1 for which the target rank was greater than 6. Note that Fast-Impute was the best performing benchmark method across the synthetic data experiments so it consists of a reasonable method to compare against. In Figure 4.11, we report the coefficient of determination (R^2) achieved by Algorithm 5 on the side information both overall and on individual features in Dataset 1 and Dataset 2. Our main findings from this set of experiments are as follows:

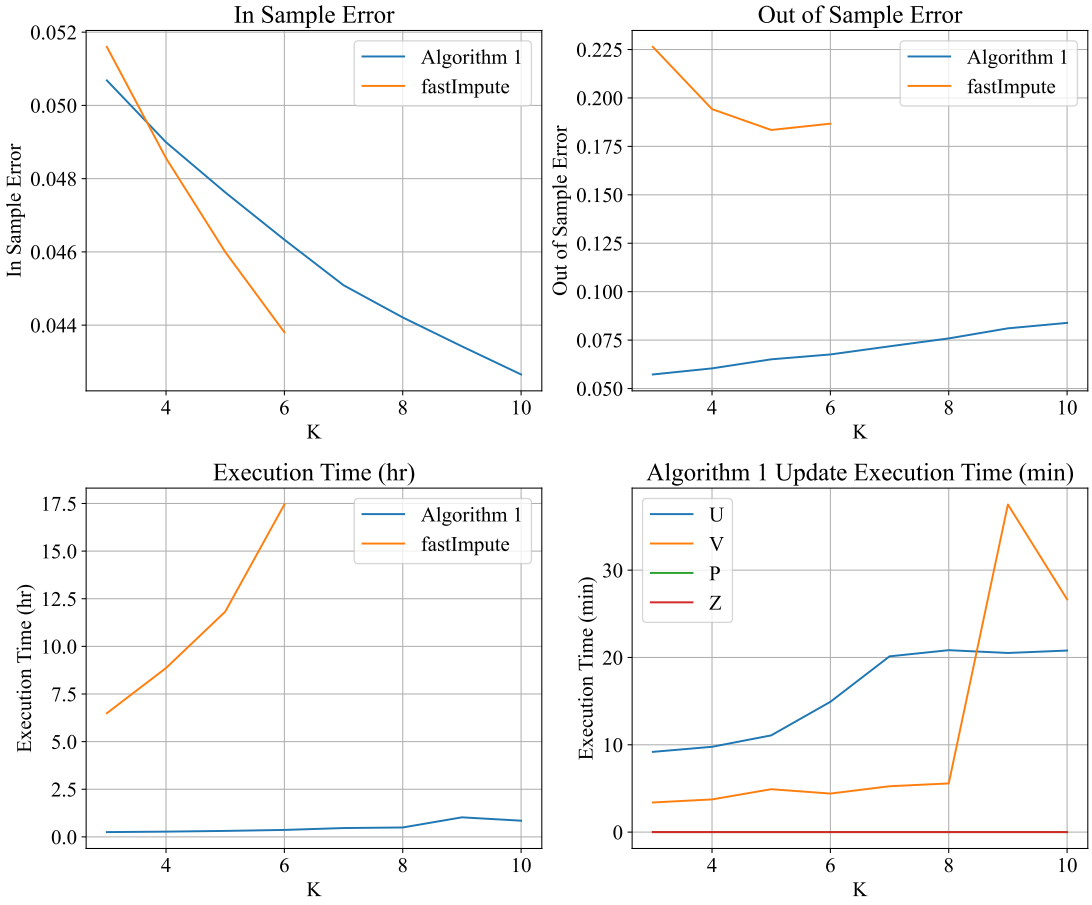

Figure 4.9: In sample ℓ_2 reconstruction error (top left), out of sample ℓ_2 reconstruction error (top right), execution time (bottom left) and subproblem execution time (bottom right) versus k on Netflix Prize Dataset 1. Averaged over 5 trials.

1. Fast-Impute in general produced solutions that achieved slightly lower in sample error

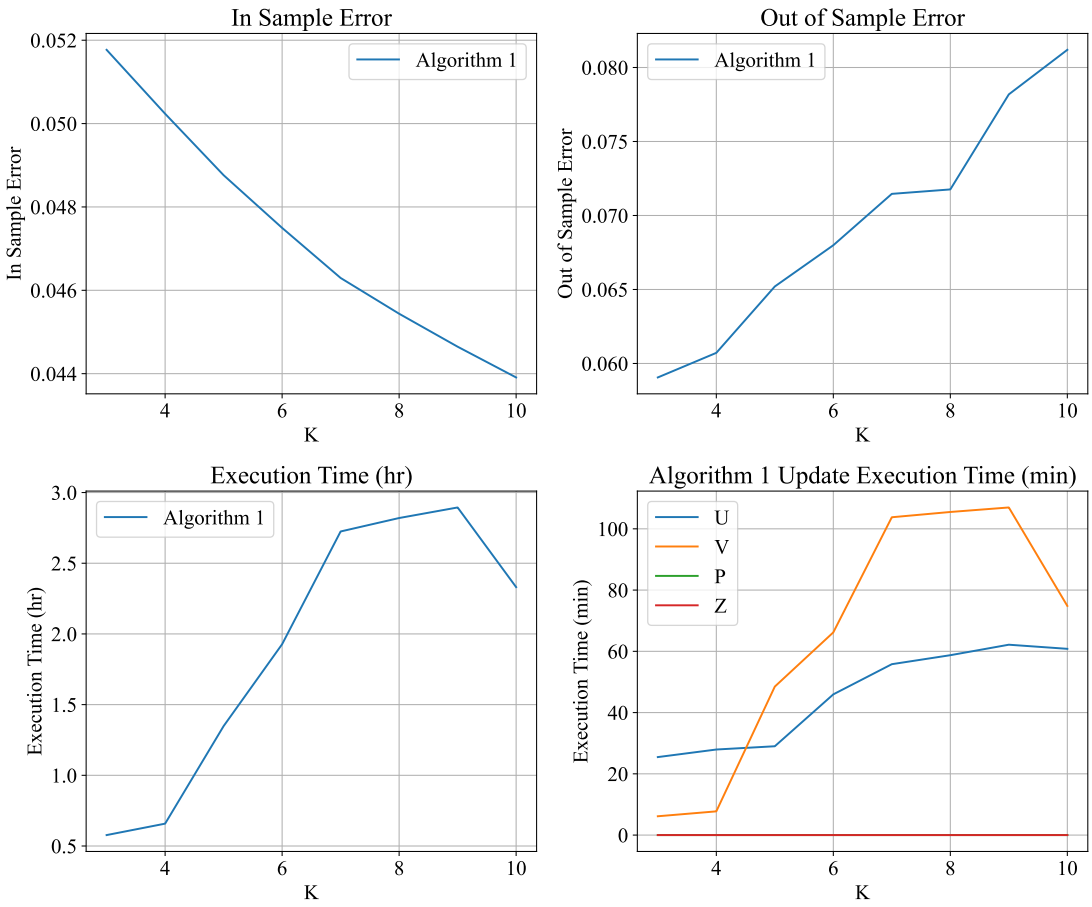

Figure 4.10: In sample ℓ_2 reconstruction error (top left), out of sample ℓ_2 reconstruction error (top right), execution time (bottom left) and subproblem execution time (bottom right) versus k on Netflix Prize Dataset 2. Averaged over 5 trials.

but significantly higher out of sample error than the solutions produced by Algorithm 5. Out of sample error is a much more important metric than in sample error as out of sample error captures the ability of a candidate solution to generalize to unseen data. Across the experiments in which Fast-Impute terminated within the specified time limit, Algorithm 5 produced solutions that on average achieved 68% lower out of sample error than Fast-Impute. The high out of sample error (relative to in sample error) of Fast-Impute suggests that this method is likely over-fitting the training data. As is expected, in sample error decreased as the rank of the reconstruction increased. In the case of Algorithm 5, out of sample error increased as the reconstruction rank

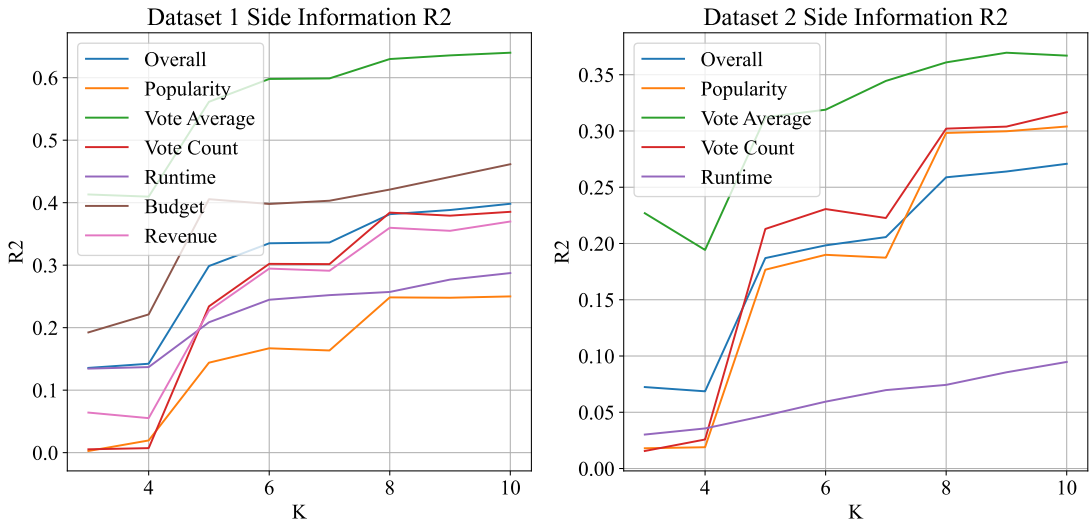

Figure 4.11: Algorithm 5 side information R^2 on Netflix Prize Dataset 1 (left) and Dataset 2 (right) versus k. Averaged over 5 trials.

increased, suggesting that Algorithm 5 was over-fitting the data as rank increased. The out of sample error of Fast-Impute initially decreased before increasing as the reconstruction rank increased.

2. Algorithm 5 exhibited significantly superior scalability than Fast-Impute. Across the experiments in which Fast-Impute terminated within the specified time limit, Algorithm 5 required on average 97% less time to execute than Fast-Impute. The execution time of Algorithm 5 on the largest tested instance (Dataset $2, k = 10$) was less than the execution time of Fast-Impute on the smallest tested instance (Dataset $1, k = 3$).

3. The computational bottleneck of Algorithm 5 is the solution time of subproblems (4.12) and (4.14). Solving these two subproblems requires an order of magnitude more time than solving subproblems (4.16) and (4.18)

4. Figure 4.11 illustrates that the reconstructed matrix produced by Algorithm 5 becomes a better predictor of the side information as the value of k increases. This is to be expected as increasing k increases model complexity. We see that popularity and runtime are the most difficult features to predict as a linear function of the reconstructed

matrix while vote average and budget are the easiest features to predict.

4.6.7 Summary of Findings

We now summarize our findings from our numerical experiments. In Sections 4.6.2-4.6.5, we see that across all experiments using synthetic data and target rank $k \leq 15$, Algorithm 5 produces solutions that achieve on average 79% lower objective value and 90.1% lower ℓ_2 reconstruction error than the solutions returned by the best performing benchmark method. In the regime where $k > 15$, we see in Section 4.6.5 that Fast-Impute outperforms Algorithm 5. We see that the execution time of Algorithm 5 is competitive with and often notably faster than the benchmark methods on synthetic data. Importantly, in the regime $k > 15$, although Fast-Impute returns higher quality solutions than Algorithm 5, the former has an execution time that is on average 3 times as long as our method. Our computational results are consistent with the complexity analysis performed in Section 4.5 for Problems (4.12), (4.14), (4.16) and (4.18). We observe that solution time for (4.16) becomes the bottleneck as the target rank k scales, otherwise the solution time for (4.12) is the bottleneck. On real world data comprised of the Netflix Prize Dataset augmented with features from the TMDB Database, Algorithm 5 produces solutions that achieve 67% lower out of sample error than Fast-Impute in 97% less execution time.

4.7 Concluding Remarks

In this chapter, we introduced Problem (4.1) which seeks to reconstruct a partially observed matrix that is predictive of fully observed side information. We illustrate that (4.1) has a natural interpretation as a robust optimization problem and can be reformulated as a mixed-projection optimization problem. We derive a semidefinite cone relaxation (4.9) to (4.1) and we present Algorithm 5, a mixed-projection alternating direction method of multipliers algorithm that obtains scalable, high quality solutions to (4.1). We rigorously benchmark the

performance of Algorithm 5 on synthetic and real world data against benchmark methods Fast-Impute, Soft-Impute, Iterative-SVD and ScaledGD. We find that across all synthetic data experiments with $k \leq 15$, Algorithm 5 outputs solutions that achieve on average 79% lower objective value in (4.1) and 90.1% lower ℓ_2 reconstruction error than the solutions returned by the best performing benchmark method. For the 5 synthetic data experiments with $k > 15$, Fast-Impute returns superior quality solutions than Algorithm 5, however the former takes on average 3 times as long as Algorithm 5 to execute. The runtime of Algorithm 5 is competitive with and often superior to that of the benchmark methods. Algorithm 5 is able to solve problems with $n = 10000$ rows and $m = 10000$ columns in less than a minute. On real world data from the Netflix Prize competition, Algorithm 5 produces solutions that achieve 67% lower out of sample error than benchmark methods in 97% less execution time. Future work could expand the mixed-projection ADMM framework introduced in this work to incorporate positive semidefinite constraints and general linear constraints. Additionally, future work could empirically investigate the strength of the semidefinite relaxation (4.9) and could explore how to leverage this lower bound to certify globally optimal solutions.

4.8 Appendix: Supplemental Computational Results

Table 4.1: Comparison of the objective value of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Objective				
N	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
100	249262.99	**655.46**	5530.48	19893.08	28677.73
200	306738.68	**1280.82**	9756.14	24251.93	44054.89
400	417643.27	**2483.49**	14321.46	28932.85	61112.62
800	421032.49	**4813.05**	31520.96	41179.38	93119.34
1000	522586.08	**6010.34**	33557.75	47701.68	107851.28
2000	563033.20	**11975.48**	83669.84	76566.52	167458.68
5000	1226489.68	**30060.61**	273747.04	172093.66	364065.64
10000	1973665.62	**60082.27**	521189.88	319915.98	642759.84

Table 4.2: Comparison of the reconstruction error of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error				
N	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
100	100.22460	**0.01520**	0.07540	0.21330	0.30090
200	58.37210	**0.00695**	0.12800	0.11770	0.21050
400	33.92500	**0.00412**	0.08980	0.07390	0.15980
800	14.97890	**0.00328**	0.23990	0.05160	0.13010
1000	12.66500	**0.00312**	0.18600	0.04950	0.12620
2000	5.54420	**0.00304**	0.07990	0.04410	0.11670
5000	2.47260	**0.00282**	0.15040	0.03720	0.10550
10000	1.32070	**0.00267**	0.10920	0.03510	0.10240

Table 4.3: Comparison of the side information R^2 of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Side Information R^2				
N	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
100	0.157	0.983	0.868	**1.000**	**1.000**
200	0.193	**0.984**	0.878	0.837	0.740
400	0.167	**0.985**	0.912	0.866	0.780
800	0.441	**0.985**	0.911	0.905	0.826
1000	0.328	**0.985**	0.896	0.906	0.829
2000	0.557	**0.985**	0.924	0.920	0.862
5000	0.525	**0.985**	0.889	0.928	0.879
10000	0.582	**0.985**	0.840	0.932	0.888

Table 4.4: Comparison of the execution time of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus n with $m = 100, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)				
N	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
100	**10.84**	53.47	99.95	141.42	115.26
200	**41.11**	73.84	121.05	187.26	163.11
400	**54.95**	113.89	152.16	262.95	246.42
800	**68.00**	184.11	195.05	389.63	334.16
1000	**44.11**	241.16	211.63	442.05	366.53
2000	**124.47**	340.84	311.32	759.58	575.79
5000	813.53	770.11	**484.58**	1795.58	1298.63
10000	18828.21	1730.00	**863.84**	3871.53	2713.26

Table 4.5: Comparison of the objective value of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Objective				
M	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
100	530097.31	**6014.93**	44337.08	47334.20	103403.04
200	2483913.82	**6131.52**	12159.08	29560.77	114448.43
400	14226534.31	**6361.90**	13875.31	21942.31	90652.40
800	99356634.18	**6800.63**	22152.43	37924.99	87895.33
1000	105451997.06	**7126.60**	24435.40	46327.69	128499.51
2000	591164404.77	**13964.62**	44333.51	95044.30	815807.16
5000	4002087935.12	**41679.23**	96985.27	308044.93	11294104.83
10000	9826251365.01	**117558.76**	197362.37	968255.00	60913874.17

Table 4.6: Comparison of the reconstruction error of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error				
M	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
100	13.68740	**0.00322**	0.14530	0.04960	0.12560
200	40.58900	**0.00154**	0.00590	0.01260	0.06640
400	127.71450	**0.00075**	0.00340	0.00340	0.02240
800	508.24550	**0.00036**	0.00340	0.00310	0.00460
1000	443.26620	**0.00029**	0.00310	0.00300	0.00350
2000	1292.61610	**0.00012**	0.00330	0.00290	0.00300
5000	3658.18810	**0.00004**	0.00310	0.00270	0.00540
10000	4559.27360	**0.00002**	0.00320	0.00270	0.00650

Table 4.7: Comparison of the execution time of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus m with $n = 1000, k = 5$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)				
M	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
100	**44.16**	222.21	193.16	444.89	365.74
200	**54.32**	237.37	223.32	953.32	760.11
400	**80.95**	311.32	315.47	1466.32	1511.79
800	**121.37**	637.53	360.53	2198.21	2564.00
1000	**154.89**	728.58	434.47	2611.58	3009.21
2000	4652.11	**1181.47**	5127.37	5308.16	6062.89
5000	28587.11	**12645.16**	40526.16	32824.79	35015.21
10000	108255.05	**39569.37**	156399.42	82361.37	86762.84

Table 4.8: Comparison of the objective value of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	Objective				
D	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
10	11691.78	**475.17**	4222.64	3367.41	7710.45
50	96771.27	**2067.20**	14565.83	16511.97	37203.79
100	229740.41	**4033.46**	28967.06	31000.81	68634.31
150	532018.26	**6057.23**	37244.64	45581.92	106504.32
200	734648.30	**7994.51**	42289.45	64771.47	141252.05
250	1195065.81	**9984.20**	71433.91	83752.90	183720.00
500	4165782.61	**20094.24**	120114.14	158458.64	361740.57
1000	13578263.54	**40191.29**	230467.93	294273.13	668550.27

Table 4.9: Comparison of the reconstruction error of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error				
D	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
10	0.60780	**0.00690**	0.15210	0.04840	0.12530
50	1.41600	**0.00471**	0.09780	0.05020	0.12730
100	5.03590	**0.00382**	0.24280	0.05130	0.12790
150	14.00330	**0.00326**	0.19790	0.05030	0.12650
200	22.26870	**0.00276**	0.08640	0.04840	0.12510
250	39.41630	**0.00245**	0.19210	0.04830	0.12580
500	177.68800	**0.00165**	0.15700	0.04860	0.12640
1000	679.11770	**0.00104**	0.09360	0.04990	0.12660

Table 4.10: Comparison of the side information R^2 of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	Side Information R^2				
D	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
10	0.912	**0.988**	0.907	0.903	0.821
50	0.528	**0.986**	0.899	0.903	0.825
100	0.533	**0.985**	0.914	0.911	0.840
150	0.303	**0.985**	0.925	0.905	0.831
200	0.411	**0.985**	0.909	0.907	0.836
250	0.291	**0.986**	0.913	0.908	0.834
500	0.151	**0.985**	0.906	0.905	0.826
1000	0.292	**0.984**	0.891	0.909	0.838

Table 4.11: Comparison of the execution time of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus d with $n = 1000, m = 100$ and $k = 5$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)				
D	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
10	**84.00**	199.89	222.68	465.11	386.53
50	**80.79**	193.05	226.00	459.53	385.00
100	**108.63**	245.11	226.53	457.84	380.11
150	**113.47**	246.16	218.47	452.21	380.79
200	**117.32**	318.26	243.05	455.32	387.05
250	**152.79**	362.63	229.63	487.21	412.79
500	**176.21**	365.37	274.26	439.05	358.95
1000	**138.74**	449.42	255.32	467.95	392.00

Table 4.12: Comparison of the objective value of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Objective				
K	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
5	514330.09	**6021.76**	41376.50	45858.20	106390.58
10	1892278.99	**7393.65**	255228.60	318398.98	805396.86
15	5213393.44	**14104.62**	115383.43	1112396.97	2495972.46
20	10196279.89	328671.00	**101812.52**	2628073.04	4910386.51
25	16816442.74	1069103.04	**116005.02**	4526388.13	7541300.54
30	-	34567679.83	10695127.17	**6864577.33**	10634436.81
35	39536651.09	187701091.79	**144464.25**	9424715.13	14192827.60
40	-	723504611.03	191276652.97	**12529277.05**	18290215.22

Table 4.13: Comparison of the reconstruction error of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	ℓ_2 Reconstruction Error				
K	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
5	12.95500	**0.00314**	0.10940	0.05090	0.12900
10	5.41720	**0.00288**	0.24270	0.08970	0.22520
15	3.45210	**0.00871**	0.01940	0.14010	0.30200
20	2.34590	0.04900	**0.00903**	0.18270	0.34220
25	1.73610	0.11890	**0.00678**	0.21480	0.36130
30	-	0.20030	**0.00562**	0.23810	0.37240
35	1.16780	0.17330	**0.00488**	0.25260	0.37740
40	-	0.22320	**0.00452**	0.26550	0.38020

Table 4.14: Comparison of the execution time of ScaledGD, Algorithm 5, Fast-Impute, Soft-Impute and SVD versus k with $n = 1000, m = 100$ and $d = 150$. Averaged over 20 trials for each parameter configuration.

	Execution Time (ms)				
K	ScaledGD	Algorithm 5	Fast-Impute	Soft-Impute	SVD
5	**55.79**	227.47	205.16	460.11	374.79
10	**80.79**	298.63	823.21	477.95	361.95
15	**107.89**	502.21	1817.16	509.68	346.05
20	**95.53**	713.42	2420.89	533.79	324.79
25	**111.53**	1110.89	3586.05	569.68	297.58
30	-	1353.95	4435.63	591.21	**280.37**
35	**107.21**	1822.16	6212.63	640.05	271.00
40	-	2281.95	8168.68	645.11	**263.37**

Table 4.15: Comparison of the in sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time of Algorithm 5 and Fast-Impute versus k on Netflix Prize Dataset 1. Averaged over 5 trials.

	In Sample Error		Out of Sample Error		Execution Time (hr)	
K	Algorithm 5	Fast-Impute	Algorithm 5	Fast-Impute	Algorithm 5	Fast-Impute
3	**0.0507**	0.0516	**0.0573**	0.2264	**0.2512**	6.4907
4	0.0490	**0.0486**	**0.0604**	0.1942	**0.2760**	8.8641
5	0.0476	**0.0460**	**0.0651**	0.1835	**0.3136**	11.8207
6	0.0463	**0.0438**	**0.0676**	0.1867	**0.3654**	17.4471
7	**0.0451**	-	**0.0718**	-	**0.4637**	-
8	**0.0442**	-	**0.0759**	-	**0.4941**	-
9	**0.0434**	-	**0.0811**	-	**1.0262**	-
10	**0.0427**	-	**0.0839**	-	**0.8503**	-

Table 4.16: In sample ℓ_2 reconstruction error, out of sample ℓ_2 reconstruction error and execution time of Algorithm 5 versus k on Netflix Prize Dataset 2. Averaged over 5 trials.

K	In Sample Error	Out of Sample Error	Execution Time (hr)
3	0.0518	0.0591	0.5773
4	0.0502	0.0607	0.6581
5	0.0488	0.0652	1.3482
6	0.0475	0.0680	1.9273
7	0.0463	0.0715	2.7246
8	0.0454	0.0718	2.8198
9	0.0446	0.0782	2.8945
10	0.0439	0.0812	2.3311

Table 4.17: Algorithm 5 side information R^2 on Netflix Prize Dataset 1. Averaged over 5 trials.

K	Overall	Popularity	Vote Average	Vote Count	Runtime	Budget	Revenue
3	0.136	0.003	0.413	0.005	0.134	0.192	0.064
4	0.142	0.02	0.41	0.007	0.137	0.221	0.055
5	0.299	0.144	0.561	0.234	0.209	0.406	0.227
6	0.335	0.167	0.598	0.302	0.245	0.398	0.295
7	0.336	0.163	0.599	0.302	0.252	0.403	0.291
8	0.382	0.248	0.63	0.384	0.257	0.421	0.36
9	0.388	0.248	0.636	0.379	0.277	0.441	0.355
10	0.398	0.25	0.64	0.385	0.287	0.462	0.37

Table 4.18: Algorithm 5 side information R^2 on Netflix Prize Dataset 2. Averaged over 5 trials.

K	Overall	Popularity	Vote Average	Vote Count	Runtime
3	0.072	0.018	0.227	0.016	0.03
4	0.069	0.019	0.194	0.026	0.036
5	0.187	0.177	0.312	0.213	0.047
6	0.198	0.19	0.319	0.231	0.059
7	0.206	0.187	0.344	0.223	0.07
8	0.259	0.298	0.361	0.302	0.074
9	0.264	0.3	0.37	0.304	0.086
10	0.271	0.304	0.367	0.317	0.095

Chapter 5

Conclusion

In this thesis, we considered a collection of fundamental statistics and machine learning problems that exhibit cardinality or rank constraints and designed algorithms that outperform existing convex relaxations and heuristics by leveraging techniques from mixed-integer and mixed-projection optimization. We make important theoretical and applied contributions.

Chapter 2 considered a novel formulation of the Sparse Plus Low Rank Matrix Decomposition problem that exploits discreteness and leverages regularization. We designed an alternating minimization heuristic that computes high quality feasible solutions and outperforms benchmark methods, scaling to dimension $n = 10000$ in minutes. We additionally designed a custom branch and bound algorithm that leverages a strong semidefinite relaxation to globally solve problem instances of dimension up to $n = 25$ in minutes.

In Chapter 3, we introduced a ℓ_2 regularized formulation of the Compressed Sensing problem which we reformulated exactly as a mixed-integer second order cone problem. For this problem, we presented a custom branch and bound algorithm that can compute globally optimal solutions. We found that our approach produced solutions that were on average 6.22% more sparse on synthetic data and 9.95% more sparse on real world ECG data when compared to state of the art benchmark approaches.

Chapter 4 studied an important generalization of the well known Matrix Completion

problem where we seek to reconstruct a partially observed matrix that is predictive of fully observed side information. We reformulated this problem as a mixed-projection optimization problem and presented an alternating direction method of multipliers algorithm that can solve problems with $n = 10000$ rows and $m = 10000$ columns in less than a minute. We found that in the low rank regime, this algorithm outputs solutions that achieve on average 79% lower objective value in and 90.1% lower ℓ_2 reconstruction error than the solutions returned by benchmark methods.

Together, our rigorous analysis of the algorithms presented across these chapters contributes to advancing the theoretical foundations of optimization theory over cardinality and rank constraints while our high performance open-source implementation of these algorithms consists of a practical tool set that can be used by practitioners to tackle these problems in the field.

Bibliography

[1] Aharon, Michal ; Elad, Michael ; Bruckstein, Alfred: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. In: *IEEE Transactions on signal processing* 54 (2006), Nr. 11, S. 4311–4322

[2] Arous, Gérard Ben ; Wein, Alexander S. ; Zadik, Ilias: Free energy wells and overlap gap property in sparse PCA. In: *Conference on Learning Theory* PMLR (Veranst.), 2020, S. 479–482

[3] Asif, M S. ; Romberg, Justin: Fast and Accurate Algorithms for Re-Weighted L1-Norm Minimization. In: *IEEE Transactions on Signal Processing* 61 (2013), Nr. 23, S. 5905–5916

[4] Askari, Armin ; d'Aspremont, Alexandre ; Ghaoui, Laurent E.: Approximation bounds for sparse programs. In: *SIAM Journal on Mathematics of Data Science* 4 (2022), Nr. 2, S. 514–530

[5] Bach, Francis R. ; Jordan, Michael I.: Predictive low-rank decomposition for kernel methods. In: *Proceedings of the 22nd international conference on Machine learning*, 2005, S. 33–40

[6] Baraniuk, Richard ; Davenport, Mark ; DeVore, Ronald ; Wakin, Michael: A simple proof of the restricted isometry property for random matrices. In: *Constructive Approximation* 28 (2008), Nr. 3, S. 253–263

[7] Basri, Ronen ; Jacobs, David W.: Lambertian reflectance and linear subspaces. In: *IEEE transactions on pattern analysis and machine intelligence* 25 (2003), Nr. 2, S. 218–233

[8] Basu, Sumanta ; Li, Xianqi ; Michailidis, George: Low Rank and Structured Modeling of High-Dimensional Vector Autoregressions. In: *IEEE Transactions on Signal Processing* 67 (2019), Nr. 5, S. 1207–1222

[9] Bell, Robert M. ; Koren, Yehuda: Lessons from the Netflix prize challenge. In: *Acm Sigkdd Explorations Newsletter* 9 (2007), Nr. 2, S. 75–79

[10] Ben-Tal, Aharon ; Den Hertog, Dick: Hidden conic quadratic representation of some nonconvex quadratic optimization problems. In: *Mathematical Programming* 143 (2014), Nr. 1, S. 1–29

[11] Berk, Lauren ; Bertsimas, Dimitris: Certifiably optimal sparse principal component analysis. In: *Mathematical Programming Computation* 11 (2019), Nr. 3, S. 381–420

[12] Bertsekas, Dimitri P.: *Nonlinear programming*. 3rd. Athena Scientific Belmont MA, 2016

[13] Bertsimas, Dimitris ; Copenhaver, Martin S.: Characterization of the equivalence of robustification and regularization in linear and matrix regression. In: *European Journal of Operational Research* 270 (2018), Nr. 3, S. 931–942

[14] Bertsimas, Dimitris ; Copenhaver, Martin S. ; Mazumder, Rahul: Certifiably optimal low rank factor analysis. In: *Journal of Machine Learning Research* 18 (2017), Nr. 1, S. 907–959

[15] Bertsimas, Dimitris ; Cory-Wright, Ryan ; Johnson, Nicholas A. G.: Sparse Plus Low Rank Matrix Decomposition: A Discrete Optimization Approach. In: *Journal of Machine Learning Research* 24 (2023), Nr. 267, S. 1–51

[16] Bertsimas, Dimitris ; Cory-Wright, Ryan ; Lo, Sean ; Pauphilet, Jean: *Optimal Low-Rank Matrix Completion: Semidefinite Relaxations and Eigenvector Disjunctions*. 2023

[17] Bertsimas, Dimitris ; Cory-Wright, Ryan ; Pauphilet, Jean: A unified approach to mixed-integer optimization problems with logical constraints. In: *SIAM Journal on Optimization* 31 (2021), Nr. 3, S. 2340–2367

[18] Bertsimas, Dimitris ; Cory-Wright, Ryan ; Pauphilet, Jean: A unified approach to mixed-integer optimization problems with logical constraints. In: *SIAM Journal on Optimization* 31 (2021), Nr. 3, S. 2340–2367

[19] Bertsimas, Dimitris ; Cory-Wright, Ryan ; Pauphilet, Jean: Mixed-projection conic optimization: A new paradigm for modeling rank constraints. In: *Operations Research* 70 (2022), Nr. 6, S. 3321–3344

[20] Bertsimas, Dimitris ; Cory-Wright, Ryan ; Pauphilet, Jean: A new perspective on low-rank optimization. In: *Mathematical Programming* 202 (2023), Nr. 1, S. 47–92

[21] Bertsimas, Dimitris ; Digalakis Jr, Vassilis: The backbone method for ultra-high dimensional sparse machine learning. In: *Machine Learning* 111 (2022), Nr. 6, S. 2161–2212

[22] Bertsimas, Dimitris ; Dunn, Jack: *Machine learning under a modern optimization lens*. Dynamic Ideas LLC Charlestown, MA, 2019

[23] Bertsimas, Dimitris ; Hertog, Dick den: *Robust and adaptive optimization*. Dynamic Ideas LLC, 2022

[24] Bertsimas, Dimitris ; Johnson, Nicholas A. G.: Compressed sensing: A discrete optimization approach. In: *Machine Learning* (2024), S. 1–40

[25] Bertsimas, Dimitris ; Johnson, Nicholas A. G.: *Predictive Low Rank Matrix Learning under Partial Observations: Mixed-Projection ADMM*. 2024. – URL https://arxiv.org/abs/2407.13731

[26] Bertsimas, Dimitris ; Li, Michael L.: Fast exact matrix completion: A unified optimization framework for matrix completion. In: *Journal of Machine Learning Research* 21 (2020), Nr. 231, S. 1–43

[27] Bertsimas, Dimitris ; Pauphilet, Jean ; Van Parys, Bart: Sparse regression. In: *Statistical Science* 35 (2020), Nr. 4, S. 555–578

[28] Bertsimas, Dimitris ; Pauphilet, Jean ; Van Parys, Bart: Sparse regression: Scalable algorithms and empirical performance. In: *Statistical Science* 35 (2020), Nr. 4, S. 555–578

[29] Bertsimas, Dimitris ; Van Parys, Bart: Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. In: *The Annals of Statistics* 48 (2020), Nr. 1, S. 300–323

[30] Bezanson, Jeff ; Edelman, Alan ; Karpinski, Stefan ; Shah, Viral B.: Julia: A fresh approach to numerical computing. In: *SIAM review* 59 (2017), Nr. 1, S. 65–98

[31] Bienstock, Daniel: Eigenvalue techniques for convex objective, nonconvex optimization problems. In: *International Conference on Integer Programming and Combinatorial Optimization* Springer (Veranst.), 2010, S. 29–42

[32] Billsus, Daniel ; Pazzani, Michael J. u. a.: Learning collaborative information filters. In: *Icml* Bd. 98, 1998, S. 46–54

[33] Bourguignon, Sébastien ; Ninin, Jordan ; Carfantan, Hervé ; Mongeau, Marcel: Exact sparse approximation problems via mixed-integer programming: Formulations and computational performance. In: *IEEE Transactions on Signal Processing* 64 (2015), Nr. 6, S. 1405–1419

[34] Bousquet, Olivier ; Elisseeff, André: Stability and generalization. In: *Journal of Machine Learning Research* 2 (2002), S. 499–526

[35] Boyd, Stephen ; Boyd, Stephen P. ; Vandenberghe, Lieven: *Convex optimization*. USA : Cambridge university press, 2004

[36] Boyd, Stephen ; El Ghaoui, Laurent ; Feron, Eric ; Balakrishnan, Venkataramanan: *Linear matrix inequalities in system and control theory*. SIAM, 1994

[37] Boyd, Stephen ; Parikh, Neal ; Chu, Eric ; Peleato, Borja ; Eckstein, Jonathan u. a.: Distributed optimization and statistical learning via the alternating direction method of multipliers. In: *Foundations and Trends® in Machine learning* 3 (2011), Nr. 1, S. 1–122

[38] Brady, David J. ; Choi, Kerkil ; Marks, Daniel L. ; Horisaki, Ryoichi ; Lim, Sehoon: Compressive holography. In: *Optics express* 17 (2009), Nr. 15, S. 13040–13049

[39] Burer, Samuel ; Monteiro, Renato D.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. In: *Mathematical Programming* 95 (2003), Nr. 2, S. 329–357

[40] Burer, Samuel ; Monteiro, Renato D.: Local minima and convergence in low-rank semidefinite programming. In: *Mathematical Programming* 103 (2005), Nr. 3, S. 427–444

[41] Cai, HanQin ; Cai, Jian-Feng ; Wei, Ke: Accelerated alternating projections for robust principal component analysis. In: *Journal of Machine Learning Research* 20 (2019), Nr. 1, S. 685–717

[42] Cai, T T. ; Wang, Lie: Orthogonal matching pursuit for sparse signal recovery with noise. In: *IEEE Transactions on Information theory* 57 (2011), Nr. 7, S. 4680–4688

[43] Candes, Emmanuel ; Recht, Benjamin: Exact matrix completion via convex optimization. In: *Communications of the ACM* 55 (2012), Nr. 6, S. 111–119

[44] Candès, Emmanuel J. ; Li, Xiaodong ; Ma, Yi ; Wright, John: Robust principal component analysis? In: *Journal of the ACM* 58 (2011), Nr. 3, S. 1–37

[45] Candes, Emmanuel J. ; Plan, Yaniv: Matrix completion with noise. In: *Proceedings of the IEEE* 98 (2010), Nr. 6, S. 925–936

[46] Candes, Emmanuel J. ; Tao, Terence: Decoding by linear programming. In: *IEEE transactions on information theory* 51 (2005), Nr. 12, S. 4203–4215

[47] Candes, Emmanuel J. ; Tao, Terence: Near-optimal signal recovery from random projections: Universal encoding strategies? In: *IEEE transactions on information theory* 52 (2006), Nr. 12, S. 5406–5425

[48] Candès, Emmanuel J. ; Tao, Terence: The power of convex relaxation: Near-optimal matrix completion. In: *IEEE transactions on information theory* 56 (2010), Nr. 5, S. 2053–2080

[49] Candes, Emmanuel J. ; Wakin, Michael B. ; Boyd, Stephen P.: Enhancing sparsity by reweighted L1 minimization. In: *Journal of Fourier analysis and applications* 14 (2008), Nr. 5, S. 877–905

[50] Chandrasekaran, Venkat ; Sanghavi, Sujay ; Parrilo, Pablo A. ; Willsky, Alan S.: Rank-sparsity incoherence for matrix decomposition. In: *SIAM Journal on Optimization* 21 (2011), Nr. 2, S. 572–596

[51] Chen, Junbo ; Liu, Shouyin ; Huang, Min: Low-rank and sparse decomposition model for accelerating dynamic MRI reconstruction. In: *Journal of Healthcare Engineering* 2017 (2017)

[52] Chen, Junxin ; Xing, Jiazhu ; Zhang, Leo Y. ; Qi, Lin: Compressed sensing for electrocardiogram acquisition in wireless body sensor network: A comparative analysis. In: *International Journal of Distributed Sensor Networks* 15 (2019), Nr. 7, S. 1550147719864884

[53] Chen, Scott S. ; Donoho, David L. ; Saunders, Michael A.: Atomic decomposition by basis pursuit. In: *SIAM review* 43 (2001), Nr. 1, S. 129–159

[54] Chen, Shaobing ; Donoho, David: Basis pursuit. In: *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers* Bd. 1 IEEE (Veranst.), 1994, S. 41–44

[55] Chen, Xiaojun ; Zhou, Weijun: Convergence of reweighted l1 minimization algorithms and unique solution of truncated lp minimization. In: *Department of Applied Mathematics, The Hong Kong Polytechnic University* (2010)

[56] Chen, Yudong ; Wainwright, Martin J.: Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. In: *arXiv preprint arXiv:1509.03025* (2015)

[57] Chi, Yuejie ; Lu, Yue M. ; Chen, Yuxin: Nonconvex optimization meets low-rank matrix factorization: An overview. In: *IEEE Transactions on Signal Processing* 67 (2019), Nr. 20, S. 5239–5269

[58] Chiang, Kai-Yang ; Hsieh, Cho-Jui ; Dhillon, Inderjit S.: Matrix completion with noisy side information. In: *Advances in neural information processing systems* 28 (2015)

[59] Dai, Wei ; Milenkovic, Olgica: Subspace pursuit for compressive sensing signal reconstruction. In: *IEEE transactions on Information Theory* 55 (2009), Nr. 5, S. 2230–2249

[60] Dempster, Arthur P. ; Laird, Nan M. ; Rubin, Donald B.: Maximum likelihood from incomplete data via the EM algorithm. In: *Journal of the royal statistical society: series B (methodological)* 39 (1977), Nr. 1, S. 1–22

[61] Dong, Hongbo ; Chen, Kun ; Linderoth, Jeff: Regularization vs. relaxation: A conic optimization perspective of statistical variable selection. In: *arXiv preprint arXiv:1510.06083* (2015)

[62] Donoho, David L.: Compressed sensing. In: *IEEE Transactions on information theory* 52 (2006), Nr. 4, S. 1289–1306

[63] Donoho, David L. ; Elad, Michael: Optimally sparse representation in general (nonorthogonal) dictionaries via L1 minimization. In: *Proceedings of the National Academy of Sciences* 100 (2003), Nr. 5, S. 2197–2202

[64] Donoho, David L. ; Johnstone, Iain M. ; Kerkyacharian, Gérard ; Picard, Dominique: Wavelet shrinkage: asymptopia? In: *Journal of the Royal Statistical Society: Series B (Methodological)* 57 (1995), Nr. 2, S. 301–337

[65] Elad, Michael ; Bruckstein, Alfred M.: A generalized uncertainty principle and sparse representation in pairs of bases. In: *IEEE Transactions on Information Theory* 48 (2002), Nr. 9, S. 2558–2567

[66] Fazel, Maryam: *Matrix rank minimization with applications*, Stanford University, Dissertation, 2002

[67] Gamarnik, David: The overlap gap property: A topological barrier to optimizing over random structures. In: *Proceedings of the National Academy of Sciences* 118 (2021), Nr. 41, S. e2108492118

[68] Ge, Rong ; Huang, Furong ; Jin, Chi ; Yuan, Yang: Escaping from saddle points—online stochastic gradient for tensor decomposition. In: *Conference on learning theory* PMLR (Veranst.), 2015, S. 797–842

[69] Gill, Patrick R. ; Wang, Albert ; Molnar, Alyosha: The in-crowd algorithm for fast basis pursuit denoising. In: *IEEE Transactions on Signal Processing* 59 (2011), Nr. 10, S. 4595–4605

[70] Gillis, Nicolas ; Glineur, François: Low-rank matrix approximation with weights or missing data is NP-hard. In: *SIAM Journal on Matrix Analysis and Applications* 32 (2011), Nr. 4, S. 1149–1165

[71] Glover, Fred: Improved linear integer programming formulations of nonlinear integer problems. In: *Management Science* 22 (1975), Nr. 4, S. 455–460

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

[73] Goodfellow, Ian J. ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. Cambridge, MA, USA : MIT Press, 2016

[74] Gribonval, Rémi ; Nielsen, Morten: Sparse representations in unions of bases. In: *IEEE transactions on Information theory* 49 (2003), Nr. 12, S. 3320–3325

[75] Gu, Quanquan ; Wang, Zhaoran W. ; Liu, Han: Low-rank and sparse structure pursuit via alternating minimization. In: *Artificial Intelligence and Statistics* PMLR (Veranst.), 2016, S. 600–609

[76] Guédon, Olivier ; Litvak, Alexander E. ; Pajor, Alain ; Tomczak-Jaegermann, Nicole: Restricted isometry property for random matrices with heavy-tailed columns. In: *Comptes Rendus Mathematique* 352 (2014), Nr. 5, S. 431–434

[77] Günlük, Oktay ; Linderoth, Jeff: Perspective reformulation and applications. In: *Mixed Integer Nonlinear Programming*. Springer, 2012, S. 61–89

[78] Guo, Ke ; Han, Deren ; Wang, David Z. ; Wu, Tingting: Convergence of ADMM for multi-block nonconvex separable optimization models. In: *Frontiers of Mathematics in China* 12 (2017), S. 1139–1162

[79] Ha, Wooseok ; Liu, Haoyang ; Barber, Rina F.: An equivalence between critical points for rank constraints versus low-rank factorizations. In: *SIAM Journal on Optimization* 30 (2020), Nr. 4, S. 2927–2955

[80] Halko, Nathan ; Martinsson, Per-Gunnar ; Tropp, Joel A.: Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. In: *SIAM Review* 53 (2011), Nr. 2, S. 217–288

[81] Hashemi, Ali ; Rostami, Mohammad ; Cheung, Ngai-Man: Efficient environmental temperature monitoring using compressed sensing. In: *2016 Data Compression Conference (DCC)* IEEE Computer Society (Veranst.), 2016, S. 602–602

[82] Hsu, Daniel J. ; Kakade, Sham M. ; Langford, John ; Zhang, Tong: Multi-label prediction via compressed sensing. In: *Advances in neural information processing systems* 22 (2009)

[83] Jain, Prateek ; Netrapalli, Praneeth: Fast exact matrix completion with finite samples. In: *Conference on Learning Theory* PMLR (Veranst.), 2015, S. 1007–1034

[84] Jain, Prateek ; Netrapalli, Praneeth ; Sanghavi, Sujay: Low-rank matrix completion using alternating minimization. In: *Proceedings of the forty-fifth Annual ACM Symposium on Theory of Computing*, 2013, S. 665–674

[85] Ji, Hui ; Liu, Chaoqiang ; Shen, Zuowei ; Xu, Yuhong: Robust video denoising using low rank matrix completion. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* IEEE (Veranst.), 2010, S. 1791–1798

[86] Jin, Chi ; Kakade, Sham M. ; Netrapalli, Praneeth: Provable efficient online matrix completion via non-convex stochastic gradient descent. In: *Advances in Neural Information Processing Systems* 29 (2016)

[87] Kai, Wang ; Jingzhi, Liu ; Yanjun, Cai: Compressed Sensing based Multi-label Classification without Label Sparsity Level Prior. In: *Proceedings of the 2017 International Conference on Deep Learning Technologies*, 2017, S. 66–69

[88] Kapoor, Ashish ; Viswanathan, Raajay ; Jain, Prateek: Multilabel classification using bayesian compressed sensing. In: *Advances in neural information processing systems* 25 (2012)

[89] Karahanoğlu, N B. ; Erdoğan, Hakan ; Birbil, Ş İlker: A mixed integer linear programming formulation for the sparse recovery problem in compressed sensing. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* IEEE (Veranst.), 2013, S. 5870–5874

[90] Koren, Yehuda ; Bell, Robert ; Volinsky, Chris: Matrix factorization techniques for recommender systems. In: *Computer* 42 (2009), Nr. 8, S. 30–37

[91] Kyrillidis, Anastasios ; Cevher, Volkan: Matrix ALPS: Accelerated low rank and sparse matrix reconstruction. In: *2012 IEEE Statistical Signal Processing Workshop (SSP)* IEEE (Veranst.), 2012, S. 185–188

[92] Land, Ailsa H. ; Doig, Alison G.: *An Automatic Method for Solving Discrete Programming Problems*. S. 105–132. In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010

[93] Lasserre, Jean B.: An explicit exact SDP relaxation for nonlinear 0-1 programs. In: *Integer Programming and Combinatorial Optimization: 8th International IPCO Conference Utrecht, The Netherlands, June 13–15, 2001 Proceedings 8* Springer (Veranst.), 2001, S. 293–303

[94] Lasserre, Jean B.: Global optimization with polynomials and the problem of moments. In: *SIAM Journal on optimization* 11 (2001), Nr. 3, S. 796–817

[95] Lasserre, Jean B.: *Moments, positive polynomials and their applications*. Bd. 1. France : World Scientific, 2009

[96] Lavaei, Javad ; Low, Steven H.: Zero duality gap in optimal power flow problem. In: *IEEE Transactions on Power systems* 27 (2011), Nr. 1, S. 92–107

[97] Lee, Jon ; Zou, Bai: Optimal rank-sparsity decomposition. In: *Journal of Global Optimization* 60 (2014), Nr. 2, S. 307–315

[98] Little, John D.: *Branch and bound methods for combinatorial problems*, MIT, Dissertation, 1966

[99] Liu, Qi: Power Network System Identification and Recovery Based on the Matrix Completion. In: *Journal of Physics: Conference Series* Bd. 1237 IOP Publishing (Veranst.), 2019, S. 032059

[100] Lubin, Miles ; Zadik, Ilias ; Vielma, Juan P.: Mixed-integer convex representability. In: *International Conference on Integer Programming and Combinatorial Optimization* Springer (Veranst.), 2017, S. 392–404

[101] Lustig, Michael ; Donoho, David ; Pauly, John M.: Sparse MRI: The application of compressed sensing for rapid MR imaging. In: *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine* 58 (2007), Nr. 6, S. 1182–1195

[102] Ma, Cong ; Wang, Kaizheng ; Chi, Yuejie ; Chen, Yuxin: Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval and matrix completion. In: *International Conference on Machine Learning* PMLR (Veranst.), 2018, S. 3345–3354

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

[104] Mallat, Stéphane G ; Zhang, Zhifeng: Matching pursuits with time-frequency dictionaries. In: *IEEE Transactions on signal processing* 41 (1993), Nr. 12, S. 3397–3415

[105] Mazumder, Rahul ; Hastie, Trevor ; Tibshirani, Robert: Spectral Regularization Algorithms for Learning Large Incomplete Matrices. In: *Journal of Machine Learning Research* 11 (2010), Nr. 80, S. 2287–2322. – URL http://jmlr.org/papers/v11/mazumder10a.html

[106] Morrison, David R. ; Jacobson, Sheldon H. ; Sauppe, Jason J. ; Sewell, Edward C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. In: *Discrete Optimization* 19 (2016), S. 79–102

[107] Needell, Deanna: Noisy signal recovery via iterative reweighted l1-minimization. In: *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers* IEEE (Veranst.), 2009, S. 113–117

[108] Negahban, Sahand ; Wainwright, Martin J.: Estimation of (near) low-rank matrices with noise and high-dimensional scaling. In: *The Annals of Statistics* (2011), S. 1069–1097

[109] Nesterov, Yurii: A method for solving the convex programming problem with convergence rate O (1/k2). In: *Dokl akad nauk Sssr* Bd. 269, 1983, S. 543

[110] Netrapalli, Praneeth ; Niranjan, UN ; Sanghavi, Sujay ; Anandkumar, Animashree ; Jain, Prateek: Non-convex robust PCA. In: *arXiv preprint arXiv:1410.7660* (2014)

[111] Nguyen, Luong T. ; Kim, Junhan ; Shim, Byonghyo: Low-rank matrix completion: A contemporary survey. In: *IEEE Access* 7 (2019), S. 94215–94237

[112] Otazo, Ricardo ; Candes, Emmanuel ; Sodickson, Daniel K.: Low-rank plus sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic components. In: *Magnetic resonance in medicine* 73 (2015), Nr. 3, S. 1125–1136

[113] Overton, Michael L. ; Womersley, Robert S.: On the sum of the largest eigenvalues of a symmetric matrix. In: *SIAM Journal on Matrix Analysis and Applications* 13 (1992), Nr. 1, S. 41–45

[114] Owen, Art B. ; Perry, Patrick O.: Bi-cross-validation of the SVD and the nonnegative matrix factorization. In: *The Annals of Applied Statistics* 3 (2009), Nr. 2, S. 564 – 594

[115] Pati, Yagyensh C. ; Rezaiifar, Ramin ; Krishnaprasad, Perinkulam S.: Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: *Proceedings of 27th Asilomar conference on signals, systems and computers* IEEE (Veranst.), 1993, S. 40–44

[116] Pearson, Karl: On lines and planes of closest fit to systems of points in space. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2 (1901), Nr. 11, S. 559–572

[117] Petersen, Kaare B. ; Pedersen, Michael S. u. a.: The matrix cookbook. In: *Technical University of Denmark* 7 (2008), Nr. 15, S. 510

[118] Pilanci, Mert ; Wainwright, Martin J. ; El Ghaoui, Laurent: Sparse learning via Boolean relaxations. In: *Mathematical Programming* 151 (2015), Nr. 1, S. 63–87

[119] Ramlatchan, Andy ; Yang, Mengyun ; Liu, Quan ; Li, Min ; Wang, Jianxin ; Li, Yaohang: A survey of matrix completion methods for recommendation systems. In: *Big Data Mining and Analytics* 1 (2018), Nr. 4, S. 308–323

[120] Rani, Meenu ; Dhok, Sanjay B. ; Deshmukh, Raghavendra B.: A systematic review of compressive sensing: Concepts, implementations and applications. In: *IEEE access* 6 (2018), S. 4875–4894

[121] Recht, Benjamin: Projected gradient methods. In: *Course Notes* (2012)

[122] Recht, Benjamin ; Fazel, Maryam ; Parrilo, Pablo A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. In: *SIAM Review* 52 (2010), Nr. 3, S. 471–501

[123] Reuther, Albert ; Kepner, Jeremy ; Byun, Chansup ; Samsi, Siddharth ; Arcand, William ; Bestor, David ; Bergeron, Bill ; Gadepally, Vijay ; Houle, Michael ; Hubbell, Matthew ; Jones, Michael ; Klein, Anna ; Milechin, Lauren ; Mullen, Julia ; Prout, Andrew ; Rosa, Antonio ; Yee, Charles ; Michaleas, Peter: Interactive supercomputing on 40,000 cores for machine learning and data analysis. In: *2018 IEEE High Performance extreme Computing Conference (HPEC)* IEEE (Veranst.), 2018, S. 1–6

[124] Roos, Kees ; Balvert, Marleen ; Gorissen, Bram L. ; Hertog, Dick den: A universal and structured way to derive dual optimization problem formulations. In: *INFORMS Journal on Optimization* 2 (2020), Nr. 4, S. 229–255

[125] Rubinsteyn, Alex ; Feldman, Sergey: *fancyimpute: An Imputation Library for Python*. 2016. – URL https://github.com/iskandr/fancyimpute

[126] Sarwar, Badrul ; Karypis, George ; Konstan, Joseph ; Riedl, John T.: Application of dimensionality reduction in recommender system-a case study. (2000)

[127] Saxena, Anureet ; Bonami, Pierre ; Lee, Jon: Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. In: *Mathematical programming* 124 (2010), Nr. 1, S. 383–411

[128] Skajaa, Anders ; Ye, Yinyu: A homogeneous interior-point algorithm for nonsymmetric convex conic optimization. In: *Mathematical Programming* 150 (2015), Nr. 2, S. 391–422

[129] Som, Subhojit: Learning label structure for compressed sensing based multilabel classification. In: *2016 SAI Computing Conference (SAI)* IEEE (Veranst.), 2016, S. 54–60

[130] Sun, Ruoyu ; Luo, Zhi-Quan: Guaranteed matrix completion via non-convex factorization. In: *IEEE Transactions on Information Theory* 62 (2016), Nr. 11, S. 6535–6579

[131] Tibshirani, Robert: Regression shrinkage and selection via the lasso. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58 (1996), Nr. 1, S. 267–288

[132] Tillmann, Andreas M. ; Pfetsch, Marc E.: The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing. In: *IEEE Transactions on Information Theory* 60 (2013), Nr. 2, S. 1248–1259

[133] Tong, Tian ; Ma, Cong ; Chi, Yuejie: Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. In: *Journal of Machine Learning Research* 22 (2021), Nr. 1, S. 6639–6701

[134] Tropp, J.A.: Greed is good: algorithmic results for sparse approximation. In: *IEEE Transactions on Information Theory* 50 (2004), Nr. 10, S. 2231–2242

[135] Tropp, Joel A. ; Wright, Stephen J.: Computational methods for sparse solution of linear inverse problems. In: *Proceedings of the IEEE* 98 (2010), Nr. 6, S. 948–958

[136] Troyanskaya, Olga ; Cantor, Michael ; Sherlock, Gavin ; Brown, Pat ; Hastie, Trevor ; Tibshirani, Robert ; Botstein, David ; Altman, Russ B.: Missing value estimation methods for DNA microarrays. In: *Bioinformatics* 17 (2001), Nr. 6, S. 520–525

[137] Tsoumakas, Grigorios ; Katakis, Ioannis ; Vlahavas, Ioannis: Effective and efficient multilabel classification in domains with large number of labels. In: *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)* Bd. 21, 2008, S. 53–59

[138] Udell, Madeleine ; Townsend, Alex: Why are big data matrices approximately low rank? In: *SIAM Journal on Mathematics of Data Science* 1 (2019), Nr. 1, S. 144–160

[139] Wang, Gang ; Zhao, Zhikai ; Ning, Yongjie: Design of compressed sensing algorithm for coal mine IoT moving measurement data based on a multi-hop network and total variation. In: *Sensors* 18 (2018), Nr. 6, S. 1732

[140] Wang, Hao ; Zeng, Hao ; Wang, Jiashan: An extrapolated iteratively reweighted L1 method with complexity analysis. In: *Computational Optimization and Applications* (2022), S. 1–31

[141] Wang, Hao ; Zhang, Fan ; Shi, Yuanming ; Hu, Yaohua: Nonconvex and nonsmooth sparse optimization via adaptively iterative reweighted methods. In: *Journal of Global Optimization* 81 (2021), Nr. 3, S. 717–748

[142] Wang, Jian: Support recovery with orthogonal matching pursuit in the presence of noise. In: *IEEE Transactions on Signal processing* 63 (2015), Nr. 21, S. 5868–5877

[143] Wang, Yu ; Yin, Wotao ; Zeng, Jinshan: Global convergence of ADMM in nonconvex nonsmooth optimization. In: *Journal of Scientific Computing* 78 (2019), S. 29–63

[144] Wold, Svante ; Esbensen, Kim ; Geladi, Paul: Principal component analysis. In: *Chemometrics and Intelligent Laboratory Systems* 2 (1987), Nr. 1, S. 37–52. – Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists

[145] Xu, Huan ; Caramanis, Constantine ; Mannor, Shie: Robustness and Regularization of Support Vector Machines. In: *Journal of Machine Learning Research* 10 (2009), Nr. 7

[146] Xu, Miao ; Jin, Rong ; Zhou, Zhi-Hua: Speedup matrix completion with side information: Application to multi-label learning. In: *Advances in neural information processing systems* 26 (2013)

[147] Xu, Yangyang ; Yin, Wotao ; Wen, Zaiwen ; Zhang, Yin: An alternating direction algorithm for matrix completion with nonnegative factors. In: *Frontiers of Mathematics in China* 7 (2012), S. 365–384

[148] Xu, Zheng ; De, Soham ; Figueiredo, Mario ; Studer, Christoph ; Goldstein, Tom: An empirical study of ADMM for nonconvex problems. In: *arXiv preprint arXiv:1612.03349* (2016)

[149] Yan, Qi ; Ye, Jieping ; Shen, Xiaotong: Simultaneous pursuit of sparseness and rank structures for matrix decomposition. In: *Journal of Machine Learning Research* 16 (2015), Nr. 1, S. 47–75

[150] Yi, Xinyang ; Park, Dohyung ; Chen, Yudong ; Caramanis, Constantine: Fast algorithms for robust PCA via gradient descent. In: *Advances in Neural Information Processing Systems* 29 (2016)

[151] Yuan, Xiaoming ; Yang, Junfeng: Sparse and low-rank matrix decomposition via alternating direction methods. In: *Pacific Journal of Optimization* 9 (2013), Nr. 1, S. 167–180

[152] Zhang, Teng ; Yang, Yi: Robust PCA by manifold optimization. In: *The Journal of Machine Learning Research* 19 (2018), Nr. 1, S. 3101–3139

[153] Zheng, Qinqing ; Lafferty, John: Convergence analysis for rectangular matrix completion using Burer-Monteiro factorization and gradient descent. In: *arXiv preprint arXiv:1605.07051* (2016)

[154] Zhou, Tianyi ; Tao, Dacheng: GoDec: Randomized Low-rank & Sparse Matrix Decomposition in Noisy Case. In: *Proceedings of the 28th International Conference on Machine Learning* 35 (2011), S. 33–40

[155] Zhou, Zihan ; Li, Xiaodong ; Wright, John ; Candès, Emmanuel ; Ma, Yi: Stable Principal Component Pursuit. In: *2010 IEEE International Symposium on Information Theory*, 2010, S. 1518–1522