Sparse Learning using Discrete Optimization: Scalable Algorithms and Statistical Insights

by

Hussein Hazimeh

M.S., University of Illinois at Urbana-Champaign (2016)

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 2021

Author .....
Sloan School of Management
June 24, 2021

Certified by .....
Rahul Mazumder
Robert G. James Career Development Professor
Thesis Supervisor

Accepted by .....
Georgia Perakis
William F. Pounds Professor of Management
Co-director, Operations Research Center

Sparse Learning using Discrete Optimization: Scalable Algorithms and Statistical Insights

by

Hussein Hazimeh

Submitted to the Sloan School of Management
on June 24, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Sparsity is a central concept in interpretable machine learning and high-dimensional statistics. While sparse learning problems can be naturally modeled using discrete optimization, computational challenges have historically shifted the focus towards alternatives based on continuous optimization and heuristics. Recently, growing evidence suggests that discrete optimization methods can obtain more interpretable models than popular alternatives. However, scalability issues are limiting the adoption of discrete methods and our understanding of their statistical properties. This thesis develops scalable discrete optimization methods and presents new statistical insights for a fundamental class of sparse learning problems.

In the first chapter, we consider the ℓ_0-regularized linear regression problem, which aims to select a subset of features that best predict the outcome. We propose fast, approximate algorithms, based on coordinate descent and local combinatorial optimization, and establish convergence guarantees. Empirically, we identify important high-dimensional settings where ℓ_0-based estimators achieve better statistical performance than popular sparse learning methods (e.g., based on ℓ_1 regularization). Our open-source implementation (`L0Learn`) can handle instances with millions of features and run up to 3x faster than state-of-the-art sparse learning toolkits.

In the second chapter, we propose an exact, scalable approach for ℓ_0-regularized linear regression. In particular, we develop a specialized nonlinear branch-and-bound (BnB) framework that solves a mixed integer programming (MIP) formulation of the problem. In a radical shift from modern MIP solvers, we solve the BnB subproblems using a specialized first-order method that exploits sparsity. Our open-source solver `L0BnB` can scale to instances with $\sim 10^7$ features, over 1000x larger than what modern MIP solvers can handle.

In the third chapter, we focus on ℓ_0-regularized classification. We propose an exact and novel algorithm that solves the problem via a sequence of MIP subproblems, each involving a relatively small number of binary variables. The algorithm can scale to instances with 50,000 features. We also develop fast, approximate algorithms that generalize those of the

first chapter. We show theoretically and empirically that our proposals can outperform popular sparse classification methods.

In the last two chapters, we consider structured sparse learning problems, in which group or hierarchy constraints are imposed to enhance interpretability. We develop specialized convex and discrete optimization algorithms for these problems. Our experiments indicate that the proposed algorithms are more scalable and can achieve better statistical performance than existing methods.

Thesis Supervisor: Rahul Mazumder
Title: Robert G. James Career Development Professor

Acknowledgments

First, I would like to thank my advisor Rahul Mazumder. Over the past five years, Rahul has been a great mentor who taught me a lot about becoming a successful researcher. He gave me the freedom and independence to explore various topics, and at the same time, he was always there to help and guide me. On a personal level, Rahul is very humble, compassionate, and supportive of his students. His positive attitude and encouragement kept me enthusiastic about research and made my PhD experience enjoyable.

Thank you to my thesis committee members, Rahul Mazumder, Rob Freund, Devavrat Shah, and Natalia Ponomareva, for their valuable advice and comments. In addition, I want to thank Rob for serving on my general exam committee and nominating Rahul and me for the Young Researchers Award. I was also fortunate to have Natalia Ponomareva as my mentor at Google. Natalia taught me a lot about deep learning research and went above and beyond to make sure that I had an enjoyable experience at Google.

I have been very fortunate to interact with many faculty members at the ORC. I want to thank Dimitris Bertsimas for chairing my general exam committee and for his extensive support and advice. I am also very thankful to Juan Pablo Vielma for his support, many reference letters, and for his valuable classes on integer programming. Thank you to Amr Farahat, who I learned so much from about teaching. Also, thank you to Georgia Perakis, who always made sure that things are going smoothly for me and generally for the ORC students.

At Google, I am very grateful to my mentors and collaborators: Zhe Zhao, James Chen, Petros Mol, Zhenyu Tan, Aakanksha Chowdhery, Mahesh Sathiamoorthy, Lichan Hong, and Ed Chi. Also thank you to Jeff Dean, Sergei Vassilvitskii, Sean Lu, Badih Ghazi, Miles Lubin, and Ross Anderson for their support and valuable advice.

I owe gratitude to many people whom I met before my PhD. Thank you to ChengXiang Zhai, who was an exemplary advisor during my masters. ChengXiang cared a lot about my personal growth and helped me tremendously during my PhD applications. Also, thank you

to Charles Elkan for mentoring me at UCSD and Amazon and for his valuable advice and support. From the American University of Beirut, I want to thank Imad Elhajj, Ibrahim Abu Faycal, Louay Bazzi, and Fadi Zaraket, for their support and mentorship.

Thank you to all my friends and colleagues at MIT and the ORC and to my collaborators Peter Radchenko, Antoine Dedieu, Ali Saab, Shibal Ibrahim, Wenyu Chen, and Tim Nonet. I am also very grateful to my friends: Ibrahim, Hussein, Mounir, Sami, Michael, Jeff, Abbas, Ali, Rasha, Wael, Karim, Hadi, Leyla, Tarek, Lamis, Mohammad, Rima, Manuela, and Lily.

Finally, I want to thank my family for their unconditional love and support. I dedicate this thesis to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interpretable machine learning models are desirable in many scientific and industrial applications. In critical sectors, such as healthcare and finance, interpretability is a key consideration for decision makers when assessing whether to deploy a learning model [110, 137]. The notion of interpretability in machine learning is not well defined and usually depends on the application and audience. Nonetheless, one central and broadly applicable notion is that an interpretable model should use a small number of comprehensible parameters [68, 110]. For example, in healthcare, such compact models not only engender trust but are also valuable tools to identify hidden issues and biases in the data [50].

Motivated by the need for interpretability, this thesis focuses on a fundamental class of machine learning models that use a small number of comprehensible parameters. These models are obtained by imposing *sparsity* on the model parameters, i.e., forcing many of the parameters to be inactive [85]. For example, in linear regression, imposing sparsity leads to many regression coefficients being exactly set to zero, so only a small subset of the features will have a contribution. As we will demonstrate throughout the thesis, sparse learning problems can be naturally modeled using discrete optimization. State-of-the-art discrete optimization approaches for these problems rely on commercial mixed integer programming (MIP) solvers, such as Gurobi and CPLEX [76, 34]. While commercial MIP solvers are flexible and support a wide range of problems, they face serious scalability issues when handling sparse learning

problems. Moreover, these solvers lack transparency: they provide the user with a limited understanding of and access to their internal components. Unfortunately, these issues are hindering the adoption of discrete optimization approaches in sparse learning and limiting our understanding of how such approaches perform in practice.

Our goal in this thesis is twofold. First, we aim to develop discrete optimization algorithms for sparse learning that are both scalable and transparent, unlike commercial MIP solvers. Second, we aim to advance the understanding of how such discrete optimization approaches perform statistically. The problems we consider can be classified into two categories. The first category consists of sparse regression and classification problems, and this will be the focus of Chapters 2, 3, and 4. The second category consists of structured sparsity problems. These problems are similar in spirit to the first category but impose additional group or hierarchy constraints to improve interpretability. This category will be the focus of Chapters 5 and 6.

As a motivation, we next introduce the problem of sparse linear regression. This is a core problem in the thesis and serves as a basis for the more general class of sparse learning problems we consider.

Motivating Problem: Sparse Linear Regression. Let us consider the usual linear regression setup with a data matrix $X \in \mathbb{R}^{n\times p}$, consisting of n samples and p features (predictors), and an outcome vector $y \in \mathbb{R}^n$. A linear regression model assumes that $y = X\beta^* + \epsilon$, where β^* is an unknown coefficient vector and ϵ is a noise vector. Note that we ignore the intercept term to simplify the presentation. A standard technique to estimate β^* is the method of least squares: $\min_\beta \ \frac{1}{2}\|y - X\beta\|_2^2$. Typically, least-squares solutions are dense (they have p nonzero entries), making them difficult to interpret if p is large. Moreover, if $p > n$, there can be an infinite number of least-squares solutions, which not only hinders interpretation but can also lead to overfitting. Sparsity is one important regularization technique to mitigate the interpretability and overfitting issues [85].

A natural way to impose sparsity on the least-squares problem is by constraining the number of nonzero entries in the regression vector β. This can be achieved by solving the following

ℓ_0-regularized regression problem:

$$\min_{\beta} \ \frac{1}{2}\|y - X\beta\|_2^2 \quad \text{s.t. } \|\beta\|_0 \leq k, \tag{1.1}$$

where $\|\beta\|_0$ is the ℓ_0 (pseudo) norm of β, which counts the number of nonzero entries in β, and k is a user-specified parameter. The cardinality constraint in Problem (1.1) forces the number of nonzeros in β not to exceed k. Problem (1.1) is known as *best subset selection*, since it attempts to find a subset of features that leads to the best fit (in terms of squared error). This is a fundamental problem in statistics, dating back to over five decades [14, 94]. More recently, there has been growing interest in this problem in the wider optimization, computer science, and applied math communities. From a theoretical perspective, estimators from Problem (1.1) and its variants have competitive statistical properties, especially in high dimensional settings [73, 147, 187, 189, 70].

However, due to the discrete nature of the ℓ_0 norm, Problem (1.1) is computationally challenging and is indeed known to be NP-Hard [128]. Given this challenge, the major focus in the sparse learning literature has been on computationally friendlier alternatives. A popular class of alternatives uses continuous proxies to the ℓ_0 norm. These proxies include the ℓ_1 norm [162] and nonconvex (but continuous) penalties such as SCAD [65] and MCP [185]. For these proxies, specialized and highly scalable continuous optimization algorithms have matured over decades—see [85, Chapter 5] for an overview and [69, 41] for examples of scalable software implementations. In addition to these continuous proxies, greedy heuristics for Problem (1.1), such as stepwise regression [62] and iterative hard thresholding [35], are also commonly used.

Recently, an exciting line of work shows that Problem (1.1) (or variants) can be solved to optimality by using MIP solvers [24, 28]. These works demonstrate empirically that ℓ_0-based estimators can lead to significantly more compact models and achieve better variable selection than popular alternatives (such as those based on ℓ_1 regularization). However, the problem is still generally very challenging to solve beyond $p \sim 10^3$, as modern MIP solvers can take in the order of hours to days. On the other hand, the fast solvers for ℓ_1 regularization

(or nonconvex penalties like MCP) can handle problems with $p \sim 10^3$ in milliseconds and scale to p in the millions [69, 83]. Unfortunately, the stark computational gap between ℓ_0 and ℓ_1-based estimators has not only limited the adoption of ℓ_0-based estimators, but also our understanding of how these estimators perform in practice. The main focus of this thesis will be on bridging this gap by developing scalable discrete optimization algorithms for computing ℓ_0-based estimators (for regression, classification, and structured sparsity problems). Next, we give a more detailed description of our objectives and contributions.

Objectives and Contributions. Recall that our focus in this thesis is on a fundamental class of sparse (ℓ_0-regularized) learning problems, which consists of sparse regression and classification, in addition to structured sparsity problems. We have two main objectives for these problems:

1. Develop discrete optimization methods that exploit problem structure to achieve scalability. In addition to establishing important theoretical guarantees (e.g., on convergence) for these methods, we aim to provide transparent, open-source software implementations that facilitate practical adoption and future research. In contrast, state-of-the-art discrete optimization approaches are limited in scalability and rely on commercial MIP solvers whose internal mechanisms are hidden from the user.

2. Improve the understanding of the statistical properties of estimators obtained via discrete optimization, from both the theoretical and empirical perspectives. This can lead to better insights into the settings where discrete optimization can outperform computationally friendlier alternatives. We hope that these insights will help researchers and data scientists make more informed decisions when choosing a sparse learning algorithm.

The thesis addresses the objectives above in five chapters. The results of Chapters 2, 3, 4, 5, and 6 are based on papers [86, 90, 60, 89, 87], respectively. Below we highlight the contributions and discuss connections among each of these chapters.

Chapter 2: Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms

This chapter focuses on solving the best subset selection problem with additional shrinkage-inducing penalties, i.e., variants of Problem (1.1). Recent work has shown that commercial MIP solvers can be used to solve small to moderate instances of this problem. However, these solvers are generally slow, especially when compared to the highly specialized algorithms for continuous sparse learning (for example, based on ℓ_1 regularization). To address this computational challenge, we develop fast, approximate algorithms for a class of ℓ_0-regularized regression problems. The algorithms are based on a combination of cyclic coordinate descent (CD) and local combinatorial optimization. Our choice of CD is inspired by its success and scalability in continuous sparse learning. However, standard results on the convergence of CD do not apply to our problem (because of the discontinuities in the ℓ_0 norm). Thus, we establish rigorous convergence guarantees for CD. Given a local minimizer from CD, our local combinatorial optimization algorithm searches a local neighborhood for another solution with a better objective. Generally, the local search problem can be solved using a MIP with a relatively small search space. In a special case where the neighborhood is restricted to be small, we develop a specialized algorithm that reduces the time complexity of exhaustive local search by a factor of n (where n is the number of samples).

We carry out a series of experiments on high-dimensional synthetic and real datasets, with up to 10^6 features. Our experiments show that estimators based on a combination of ℓ_0 and ℓ_2 regularization can outperform the state-of-the-art sparse learning algorithms in many important high-dimensional settings and under various statistical metrics (prediction, estimation, and variable selection). We also observe that pure ℓ_0 regularization can achieve competitive performance when the signal strength is high, but it tends to overfit in low-signal settings. The combination of ℓ_0 and ℓ_2 regularization mitigates the overfitting problem and works well in both the low and high signal regimes. We open-source the proposed algorithms through our sparse learning toolkit `L0Learn`, which is available on CRAN and Github. L0Learn reaches up to a three-fold speed-up compared to popular competing toolkits such as `glmnet` and `ncvreg`.

Chapter 3: Sparse Regression at Scale: Branch-and-Bound rooted in First-Order Optimization

While the approximate algorithms we develop in Chapter 2 can be useful from a practical perspective, they cannot certify optimality. In sensitive applications where decisions are consequential, e.g., in healthcare, certifying optimality is desirable and can engender trust in the model. In this chapter, we focus on computing ℓ_0-based estimators exactly. We consider the linear regression problem penalized with a combination of ℓ_0 and squared ℓ_2 penalties. As discussed previously, state-of-the-art exact approaches rely on commercial MIP solvers and generally face difficulties beyond $p \sim 10^3$ features.

Motivated by the limited scalability and transparency in the commercial MIP solvers, we propose a specialized MIP framework, which we transparently design from scratch. Specifically, we model the problem as a Mixed Integer Second Order Cone Program (MISOCP) and propose a specialized, nonlinear branch-and-bound (BnB) framework. Modern nonlinear BnB solvers for MISOCPs rely on primal-dual methods, such as interior-point methods, to handle the continuous node subproblems [105]. However, such methods are known to face difficulties in exploiting warm starts and sparsity. In our BnB, we take a radically different approach to solve the node subproblems: we propose a primal algorithm based on coordinate descent, designed to exploit sparsity. Our CD-based method effectively leverages information across the BnB nodes, by using warm starts, active sets, and gradient screening. Although scalable, our primal algorithm cannot readily generate the dual bounds needed for search space pruning in BnB. Thus, we develop a new method to efficiently obtain dual bounds directly from primal solutions, and provide a theoretical analysis that shows that the method works well in high dimensions. We obtain warms starts, which are generally critical to the performance of BnB, using the approximate algorithms of Chapter 2. On instances where sparse solutions are desired (≤ 30 nonzeros), our BnB can solve to optimality problems with $\sim 10^7$ features in minutes to hours, which is over 1000x larger from what can be handled using the state-of-the-art MIP solvers such as Gurobi and MOSEK. We open-source the implementation through our toolkit `L0BnB`.

Chapter 4: Learning Sparse Classifiers: Continuous and Mixed Integer Optimization Perspectives

In this chapter, we consider the problem of learning sparse classifiers, where the outcome depends on a linear combination of a small subset of features. Inspired by the success of our algorithms for ℓ_0-regularized regression in Chapters 2 and 3, we develop a scalable computational framework for ℓ_0-regularized classification. We propose two classes of scalable algorithms: an exact algorithm that can handle $p \approx 50,000$ features in a few minutes, and approximate algorithms that can address instances with $p \approx 10^6$ in times comparable to the fast ℓ_1-based algorithms. Our exact algorithm is based on the novel idea of *integrality generation*, which solves the original problem (with p binary variables) via a sequence of mixed integer programs that involve a small number of binary variables. Our approximate algorithms, based on coordinate descent and local combinatorial search, generalize those in Chapter 2 to the classification setting. We establish convergence guarantees for these approximate algorithms. In addition, we present new estimation error bounds for a class of ℓ_0-regularized classification estimators. Experiments on real and synthetic data demonstrate that our approach can lead to models with considerably improved statistical performance (especially variable selection) compared to competing sparse classification methods.

Chapter 5: Grouped Variable Selection with Discrete Optimization: Computational and Statistical Perspectives

Many machine learning applications involve features that have a natural group structure. In these applications, imposing sparsity on groups of variables, as opposed to individual variables as in Chapters 2-4, can improve the interpretability of the resulting model. We present a new algorithmic framework for grouped variable selection that is based on discrete optimization. While there are several appealing approaches based on convex relaxations and nonconvex heuristics, we focus on optimal solutions for the ℓ_0-regularized formulation, a problem that is relatively unexplored due to computational challenges. Our methodology covers both high-dimensional linear regression and nonparametric sparse additive modeling with smooth components. Our algorithmic framework consists of approximate and exact

algorithms. The approximate algorithms are based on coordinate descent and local combinatorial optimization, and these are generalizations of the algorithms developed in Chapter 2. Our exact algorithm is based on a standalone branch-and-bound (BnB) framework, which can solve the associated mixed integer programming (MIP) problem to certified optimality. By exploiting problem structure, our custom BnB algorithm can solve instances with 5×10^6 features in minutes to hours—over 1000 times larger than what is currently possible using commercial MIP solvers. We also explore statistical properties of the ℓ_0-based estimators. We demonstrate, theoretically and empirically, that our proposed estimators have an edge over popular group-sparse estimators in terms of statistical performance in various regimes.

Chapter 6: Learning Hierarchical Interactions at Scale: Convex and Discrete Optimization Algorithms

In this chapter, we consider another important structured sparsity problem, which appears in the context of interactions models. In many regression applications, it is beneficial to augment the main features with pairwise interactions. Such interaction models can be often enhanced by performing variable selection under the so-called *strong hierarchy (SH)* constraint: an interaction is nonzero only if its associated main features are nonzero. Existing convex optimization-based algorithms for learning under SH face difficulties in handling problems where the number of main features $p \sim 10^3$ (with a total number of features $\sim p^2$). In this chapter, we propose scalable algorithms that solve both convex and discrete formulations of the problem. Our convex optimization algorithm is based on proximal gradient descent. We introduce novel screening rules that allow for decomposing the complicated proximal problem and solving it in parallel. In addition, we introduce a specialized active-set strategy with gradient screening for avoiding costly gradient computations. Motivated by the success of our specialized BnB in Chapter 3, we design a specialized nonlinear BnB that solves a discrete optimization formulation of the SH problem. The BnB uses our proposed convex optimization algorithm to solve the node subproblems efficiently.

Empirically, we demonstrate that our convex optimization algorithm can achieve over a 4900x speed-up compared to the state of the art and can handle problems with $p = 50,000$

($\sim 10^9$ interactions). We open-source this algorithm through our toolkit `hierScale`. Our specialized BnB can also scale to problems with up to 50 million interactions (when highly sparse solutions are desired), which is over 1000x larger than what commercial MIP solvers can handle. Moreover, experiments on real and synthetic datasets suggest that our proposals outperform existing SH methods in terms of prediction and variable selection.

Funding: Hussein Hazimeh acknowledges research funding from the Office of Naval Research [Grants ONR-N000141512342 and ONR-N000141812298], the National Science Foundation [Grant NSF-IIS-1718258], Liberty Mutual, and the Massachusetts Institute of Technology.

Chapter 2

Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms

A version of this chapter appeared in Operations Research [86].

2.1 Introduction

The ongoing surge in high-dimensional data has drawn a lot of attention to sparse learning across several scientific communities. Indeed, sparsity can be very effective in high-dimensional settings as it leads to compact models that can be easier to interpret [43, 85]. We consider the usual linear regression setup with $y = X\beta + \epsilon$, where $y \in \mathbb{R}^n$ is the response, $X \in \mathbb{R}^{n \times p}$ is the model matrix, $\beta \in \mathbb{R}^p$ is the vector of regression coefficients, and $\epsilon \in \mathbb{R}^n$ is a noise vector. We will assume that the columns of X are standardized to have unit L_2-norm, and we ignore the intercept term to simplify the presentation. Our goal is to estimate β under the assumption that it is sparse (i.e., has few nonzeros)—a common desiderata in the high-dimensional learning framework with $p \gg n$ [43, 85]. A natural and direct way

to obtain such a sparse estimator is by minimizing the least squares loss with an L_0-norm[1] penalty on β [122]. Statistical (optimality) properties of this estimator have been extensively studied [73, 147, 187, 189]. Many appealing alternative sparsity-inducing estimators have been proposed in the literature based on Lasso [162], stepwise regression, continuous non-convex regularization [85], etc—each with different operating characteristics. Our focus in this chapter is on the algorithmic aspects of L_0-based estimators. Recent work [83, 115] has brought to light an intriguing phenomenon: in low signal-to-noise-ratio (SNR) regimes, the vanilla version of L_0 penalization suffers from overfitting. One way to mitigate this problem is by considering a larger family of estimators that includes (in addition to the L_0-penalty) an L_1 or L_2 norm regularization [115]. In this chapter, we consider the following extended family of L_0-based estimators, i.e., L_0L_q regularized regression problems of the form[2]:

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0\|\beta\|_0 + \lambda_q\|\beta\|_q^q, \tag{2.1}$$

where $q \in \{1, 2\}$ determines the type of the additional regularization (i.e., L_1 or L_2). The regularization parameter λ_0 controls the number of nonzeros (i.e., selected variables) in $\hat{\beta}$, and λ_q controls the amount of shrinkage induced by L_q regularization. In many regimes (and under suitable choices of λ_0, λ_q), estimators from Problem (2.1) exhibit superior statistical properties (variable selection, prediction, and estimation) compared to computationally friendlier alternatives (e.g., based on Lasso or stepwise regression)—see for example, [24, 189, 28, 115, 147, 187]. In spite of its potential usefulness, Problem (2.1) is NP-hard [128] and poses computational challenges. Recent work by [24] has shown that high-quality solutions can be obtained for the cardinality-constrained least squares problem via mixed integer optimization (MIO), in the order of minutes when $p \sim 1000$. However, efficient solvers for the Lasso (e.g., `glmnet` [69]) can address much larger problems within a second. Our goal is to bridge this gap in computation time by developing fast solvers that can obtain high-quality (approximate) solutions to Problem (2.1) for large and challenging instances (e.g., $p \sim 10^6$ and small n). This will allow performing systematic large-scale ex-

[1]The L_0-(pseudo) norm of β, i.e., $\|\beta\|_0$ counts the number of nonzeros in β.

[2]For computational considerations, we use an L_0 penalty instead of an L_0 constraint (as in Problem (1.1)).

periments to gain a deeper understanding of the statistical properties of L_0-based estimators and their differences with the state of the art. Such an understanding is currently limited due to computational considerations.

Our approach is based on two complementary algorithms: **(i)** cyclic coordinate descent (CD) for quickly finding solutions to Problem (2.1), and **(ii)** novel combinatorial search algorithms, which help improve solutions from (i). Particularly, the solutions obtained by (ii) cannot be improved by making small changes to their support. We establish novel convergence guarantees for our algorithms. We also address delicate implementation aspects of our algorithms and provide `L0Learn`: an open-source and efficient R/C++ toolkit available on CRAN at `https://CRAN.R-project.org/package=L0Learn` and on Github at `https://github.com/hazimehh/L0Learn`.

Current Landscape and Related Work: Our main focus is on the computational aspects of Problem (2.1). We contextualize our contribution within the rather large and impressive literature on algorithms for sparse regression—see for example, [16, 24] for an overview. We broadly categorize the main existing algorithms into two categories:

- **Proxy Algorithms and Heuristics**: Proxy algorithms use a proxy/surrogate to the L_0 norm, e.g., L_1 norm or non-convex penalties such as MCP and SCAD [162, 185, 65]. Fast solvers have been devised for these proxies (e.g., [69, 41, 117])—they typically result in good solutions (though not optimal for non-convex problems). Another approach is to use heuristics to find approximate solutions to Problem (2.1) with $\lambda_q = 0$. Popular methods include: (greedy) stepwise regression [85], iterative hard thresholding (IHT) [36, 24], greedy CD [16] and randomized CD [140].

- **Exact Algorithms**: These approaches *exactly* solve an optimization problem involving the L_0 norm. [24] use MIO to compute near-optimal solutions for least squares with a cardinality constraint for $p \approx 1000$. [28] propose a cutting plane method for a similar problem, which works well with mild sample correlations and a sufficiently large n. [118] use mixed integer linear optimization for solving an L_0-variant of the Dantzig Selector.

In spite of their usefulness, exact algorithms are usually accompanied by a steep increase in computational cost, placing them at a disadvantage compared to faster alternatives [83]. To address this challenge, our approach borrows the computational strengths of the proxy algorithms while maintaining a notion of "local combinatorial exactness"—i.e., making small perturbations to the support of the solution cannot improve its objective. Similar to the proxy algorithms, we employ cyclic CD as one of our main workhorses. We note that standard results on the convergence of cyclic CD [165] do not apply for our problem, and one of our contributions is rigorously establishing its convergence. A novelty of our work is the use of local combinatorial search to obtain high quality solutions. Our attention to the delicate computational aspects make our proposed algorithms comparable (and at times faster) in speed to the fastest proxy algorithms (e.g., `glmnet` and `ncvreg`).

Contributions: We summarize our key contributions below:

1. We introduce a new family of necessary optimality conditions for Problem (2.1), leading to a hierarchy of classes of local minima. Classes higher up in the hierarchy are of better quality.

2. We propose new algorithms based on cyclic CD and local combinatorial search to obtain these local minima. We present a novel convergence analysis of the algorithms. We formulate the local combinatorial search problems as structured MIO problems and develop efficient solvers for special cases. Our local search algorithms can run in seconds to minutes when p is in the order of 10^3 to 10^6.

3. Our open-source R/C++ toolkit, `L0Learn`, often runs faster than state-of-the-art toolkits (e.g., `glmnet` and `ncvreg`). Typical speedups (of a version of our algorithm) range between 25% to 300% for p up to 10^6 and $n \approx 10^3$.

4. Experiments on real and synthetic datasets suggest that our algorithms do a good job in optimizing Problem (2.1), with solutions often found to be similar to that of exact MIO methods, but with significantly shorter run times. In terms of statistical performance, our algorithms are found to be superior in terms of a combination of metrics (estima-

tion, prediction, and variable selection), compared to state-of-the-art methods for sparse learning.

Notation: We use the following notation throughout this chapter. We denote the set $\{1, 2, \ldots, p\}$ by $[p]$, the canonical basis for $\mathbb{R}^p$ by $e_1, \ldots, e_p$, and the standard Euclidean norm by $\|\cdot\|$. Similarly, $\|\cdot\|_q$ denotes the standard L_q-norm with $q \in \{0, 1, 2, \infty\}$. For any $\theta \in \mathbb{R}^p$ and $i \in [p]$, we define $\widetilde{\theta}_i = \langle y - \sum_{j \neq i} X_j \theta_j, X_i \rangle$. For any vector $u \in \mathbb{R}^k$, we define $\text{sign}(u) \in \mathbb{R}^k$ as a vector whose ith component is given by $\text{sign}(u_i) = u_i/|u_i|$ if $u_i \neq 0$ and $\text{sign}(u_i) \in [-1, 1]$ if $u_i = 0$. We denote the support of $\beta \in \mathbb{R}^p$ by $\text{Supp}(\beta) = \{i : \beta_i \neq 0, i \in [p]\}$. For $S \subseteq [p]$, we let $\beta_S \in \mathbb{R}^{|S|}$ denote the subvector of β with indices in S. Similarly, X_S denotes the submatrix of X with column indices S. We use U^S to denote the $p \times p$ matrix whose ith column is e_i if $i \in S$ and zero otherwise. Thus, $(U^S\beta)_i = \beta_i$ if $i \in S$ and $(U^S\beta)_i = 0$ if $i \notin S$.

Proofs of lemmas and theorems are included in Appendix 2.A.

2.2 Necessary Optimality Conditions

We present a family of necessary optimality conditions for Problem (2.1), leading to different classes of local minima[3]. Our methodology is centered around the following problem:

$$\min_{\beta \in \mathbb{R}^p} \quad F(\beta) \stackrel{\text{def}}{=} f(\beta) + \lambda_0 \|\beta\|_0, \tag{2.2}$$

where $f(\beta)$ is the least squares term with additional convex regularizers:

$$f(\beta) \stackrel{\text{def}}{=} \frac{1}{2}\|y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2. \tag{2.3}$$

We will use the shorthands: (i) (L_0L_2) to denote Problem (2.2) with $\lambda_1 = 0$ and $\lambda_2 > 0$; (ii) (L_0L_1) to denote Problem (2.2) with $\lambda_1 > 0$ and $\lambda_2 = 0$; and (iii) (L_0) to denote Problem (2.2) with $\lambda_1 = \lambda_2 = 0$. Unless specified otherwise, we will assume that $\lambda_0 > 0$.

[3]As we argue in Section 2.2.1 these also satisfy the usual notion of a local minimizer in nonlinear optimization.

Next, we present an overview of the different classes of local minima (minima for short) that we study—this is then followed by a more formal treatment.

- **Stationary Solutions**: Solutions where the directional derivative is non-negative in any direction.
- **Coordinate-wise (CW) Minima**: Solutions where optimizing w.r.t. one coordinate at a time (while keeping others fixed) cannot improve the objective.
- **PSI(k) Minima**: These are stationary solutions where (i) removing any subset (of size at most k) from the support, (ii) adding any subset (of size at most k) to the support, and (iii) optimizing over the newly added subset, cannot improve the objective.
- **FSI(k) Minima**: These are similar to PSI(k) minima except that in step (iii), if we optimize over the whole new support, the objective does not improve.
- **IHT Minima**: These are fixed points arising from the popular IHT algorithm.

We also establish the following hierarchy among the different classes introduced above:

$$\textsc{Hierarchy:} \quad \underset{\text{Minima}}{\text{FSI}(k)} \subseteq \underset{\text{Minima}}{\text{PSI}(k)} \subseteq \underset{\text{Minima}}{\text{CW}} \subseteq \underset{\text{Minima}}{\text{IHT}} \subseteq \underset{\text{Solutions}}{\text{Stationary}} \tag{2.4}$$

In the above hierarchy, stationary solutions are the weakest. As we move from the right to left, the classes become smaller (i.e, satisfy more restrictive necessary optimality conditions) until reaching the most restrictive class: FSI(k) minima. Moreover, for sufficiently large k, FSI(k) and PSI(k) minima coincide with the class of global minimizers of Problem (2.2). We now present a formal treatment of the classes of minima introduced above.

2.2.1 Stationary Solutions

For a function $g : \mathbb{R}^p \to \mathbb{R}$ and a vector $d \in \mathbb{R}^p$, we denote the (lower) directional derivative [22] of g at β in the direction d by: $g'(\beta; d) \stackrel{\text{def}}{=} \liminf_{\alpha \downarrow 0} (g(\beta + \alpha d) - g(\beta))/\alpha$. Directional derivatives play an important role in describing necessary optimality conditions for contin-

uous optimization problems [22]. Although $F(\beta)$ is not continuous, it is insightful to use the notion of a directional derivative to arrive at a basic definition of stationarity for Problem (2.2).

Definition 2.1. *(Stationary Solution) A vector $\beta^* \in \mathbb{R}^p$ is a stationary solution for Problem (2.2) if for every direction vector $d \in \mathbb{R}^p$, the lower directional derivative satisfies: $F'(\beta^*; d) \geq 0$.*

Let $\nabla f(\beta) \in \mathbb{R}^p$ denote a subgradient of $f(\beta)$. If β has a support S, the notation $\nabla_S f(\beta)$ refers to the components of $\nabla f(\beta)$ restricted to S. Lemma 2.1 gives an alternate characterization of Definition 2.1.

Lemma 2.1. *Let $\beta^* \in \mathbb{R}^p$ with support S. β^* is a stationary solution for Problem* (2.2) *iff $\nabla_S f(\beta^*) = 0$.*

Note that $\nabla_S f(\beta^*) = 0$ can be explicitly written as:

$$\beta_i^* = \text{sign}(\widetilde{\beta}_i^*)\frac{|\widetilde{\beta}_i^*| - \lambda_1}{1 + 2\lambda_2} \quad \text{and} \quad |\widetilde{\beta}_i^*| > \lambda_1 \quad \text{for all } i \in \text{Supp}(\beta^*), \tag{2.5}$$

where we recall that $\widetilde{\beta}_i^* \stackrel{\text{def}}{=} \langle y - \sum_{j \neq i} X_j \beta_j^*, X_i \rangle$. Characterization (2.5) suggests that a stationary solution β^* does not depend on λ_0 and does not impose any restriction on the coordinates outside the support. Moreover, it can be readily verified that a stationary solution to Problem (2.2) satisfies the traditional definition of a local minimum in nonlinear optimization, i.e., if β^* is a stationary solution, then there exists a $\delta > 0$ such that $F(\beta^*) \leq F(\beta)$ for any β satisfying $\|\beta - \beta^*\| < \delta$.

2.2.2 Coordinate-wise (CW) Minima

We consider a class of stationary solutions inspired by coordinate-wise algorithms [16, 22, 165].

Definition 2.2. *(CW Minimum) A vector $\beta^* \in \mathbb{R}^p$ is a CW minimum for Problem* (2.2) *if for every $i \in [p]$, β_i^* is a minimizer of $F(\beta^*)$ w.r.t. the ith coordinate (with others held*

fixed), i.e.,

$$\beta_i^* \in \arg\min_{\beta_i \in \mathbb{R}} F(\beta_1^*, \ldots, \beta_{i-1}^*, \beta_i, \beta_{i+1}^*, \ldots, \beta_p^*). \tag{2.6}$$

As every column of X has unit L_2-norm, β_i^* is given by the following thresholding operator $\widetilde{T}$:

$$\widetilde{T}(\widetilde{\beta}_i^*, \lambda_0, \lambda_1, \lambda_2) \stackrel{\text{def}}{=} \arg\min_{\beta_i \in \mathbb{R}} \left\{ \frac{1+2\lambda_2}{2}\Big(\beta_i - \frac{\widetilde{\beta}_i^*}{1+2\lambda_2}\Big)^2 + \lambda_1|\beta_i| + \lambda_0 \mathbb{1}[\beta_i \neq 0] \right\}, \tag{2.7}$$

where $\{\lambda_i\}_0^2$ and $\widetilde{\beta}_i^*$ are fixed, and the set $\widetilde{T}(\widetilde{\beta}_i^*, \lambda_0, \lambda_1, \lambda_2)$ is described below.

Lemma 2.2. *Let $\widetilde{T}$ be the thresholding operator defined in (2.7). Then,*

$$\widetilde{T}(\widetilde{\beta}_i^*, \lambda_0, \lambda_1, \lambda_2) = \begin{cases} \Big\{ \operatorname{sign}(\widetilde{\beta}_i^*) \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \Big\} & \text{if } \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ \{0\} & \text{if } \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ \Big\{ 0, \operatorname{sign}(\widetilde{\beta}_i^*) \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \Big\} & \text{if } \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}. \end{cases}$$

Lemma 2.3 presents an alternative characterization of CW minima.

Lemma 2.3. *A vector $\beta^* \in \mathbb{R}^p$ is a CW minimum iff*

$$\begin{aligned} &\beta_i^* = \operatorname{sign}(\widetilde{\beta}_i^*) \tfrac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \quad \text{and} \quad |\beta_i^*| \geq \sqrt{\tfrac{2\lambda_0}{1+2\lambda_2}}, \quad \text{for every } i \in \mathit{Supp}(\beta^*) \\ \text{and} \quad &\tfrac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \leq \sqrt{\tfrac{2\lambda_0}{1+2\lambda_2}} \qquad\qquad \text{for every } i \notin \mathit{Supp}(\beta^*). \end{aligned} \tag{2.8}$$

Comparing (2.8) to (2.5), we see that the class of stationary solutions contains the class of CW minima, and the containment is strict (in general).

2.2.3 Swap Inescapable Minima

We now introduce stationary solutions that further refine the class of CW minima, using notions from local combinatorial optimization. Given a CW minimum β^*, one might obtain a better solution by the following "swapping" operation: we set some nonzeros in β^* to zero and allow some entries from outside the support of β^* to be nonzero. Then, we optimize over the new support using one of the following rules: (a) *Partial Optimization*: we optimize only w.r.t. the coordinates added from outside the support or (b) *Full Optimization*: we optimize w.r.t. all the coordinates in the new support. This may lead to a solution with a smaller objective value. If the current solution cannot be improved using the swapping operation, we call β^* a *Swap Inescapable* minimum. Our proposal is inspired by the work of [16] for the cardinality-constrained problem—where, the authors suggest a special case of partial swap optimization involving *one* coordinate. However, the problem studied here is different: we consider L_0-penalization (versus an L_0 constraint) and a non-smooth $f(\beta)$. Furthermore, we allow multiple coordinates to be swapped at once via partial or full optimization.

Partial Swap Inescapable (PSI) Minima: We formally define *Partial Swap Inescapable* (PSI) minima, arising from the partial optimization step outlined above. Recall that for any $L \subseteq [p]$, the ith coordinate of the vector $(U^L\beta)$ is β_i if $i \in L$ and zero otherwise.

Definition 2.3. *(PSI Minima) Let k be a positive integer. A vector β^* with support S is a PSI minimum of order k, denoted by PSI(k), if it is a stationary solution and for every $S_1 \subseteq S$, $S_2 \subseteq S^c$, with $|S_1| \leq k$, $|S_2| \leq k$, the following holds*

$$F(\beta^*) \leq \min_{\beta_{S_2}} F(\beta^* - U^{S_1}\beta^* + U^{S_2}\beta).$$

The following lemma characterizes PSI minima of order one, aka PSI(1).

Lemma 2.4. *A vector $\beta^* \in \mathbb{R}^p$ is a PSI(1) minimum iff*

$$\beta_i^* = \text{sign}(\widetilde{\beta}_i^*)\frac{|\widetilde{\beta}_i^*|-\lambda_1}{1+2\lambda_2} \quad \textit{and} \quad |\beta_i^*| \geq \max\left\{\sqrt{\tfrac{2\lambda_0}{1+2\lambda_2}}, \max_{j\notin Supp(\beta^*)} \frac{|\widetilde{\beta}_{ij}^*|-\lambda_1}{1+2\lambda_2}\right\}, \quad \textit{for } i \in Supp(\beta^*)$$

$$\textit{and} \qquad \frac{|\widetilde{\beta}_i^*|-\lambda_1}{1+2\lambda_2} \leq \sqrt{\tfrac{2\lambda_0}{1+2\lambda_2}}, \qquad \textit{for } i \notin Supp(\beta^*)$$

where $\widetilde{\beta}_{ij}^* = \langle y - \sum_{l\neq i,j} X_l\beta_l, X_j\rangle$.

Lemmas 2.3 and 2.4 suggest that PSI(1) minima impose additional restrictions on the magnitude of nonzero coefficients, when compared to CW minima. The class of CW minima contains PSI(k) minima for any k. Furthermore, as k increases, the class of PSI(k) minima becomes smaller—till it coincides with the class of global minimizers of Problem (2.2).

Full Swap Inescapable (FSI) Minima: We formally define *Full Swap Inescapable* (FSI) minima, arising from the full optimization step outlined above.

Definition 2.4. *(FSI Minima) Let k be a positive integer. A vector β^* with support S is a FSI minimum of order k, denoted by FSI(k), if for every $S_1 \subseteq S$ and $S_2 \subseteq S^c$, such that $|S_1| \leq k$ and $|S_2| \leq k$, the following holds*

$$F(\beta^*) \leq \min_{\beta_{(S\setminus S_1)\cup S_2}} F(\beta^* - U^{S_1}\beta^* + U^{(S\setminus S_1)\cup S_2}\beta).$$

We note that for a fixed k, the class of PSI(k) minima contains FSI(k) minima, justifying a part of the hierarchy displayed in (2.4). As k increases, the class of FSI(k) minima becomes smaller till it coincides with the set of global minimizers of Problem (2.2). Sections 2.3.2 and 2.3.2 introduce algorithms to obtain PSI(k) minima and FSI(k) minima, respectively.

2.2.4 Stationarity Motivated by Iterative Hard Thresholding (IHT)

Proximal gradient algorithms such as IHT are popularly used for L_0-penalized least squares problems [36]. It is insightful to consider the class of stationary solutions associated with IHT and study how they compare to CW minima. Let $f_d(\beta) := \frac{1}{2}\|y - X\beta\|^2 + \lambda_2\|\beta\|^2$. The

gradient of $f_d(\beta)$ is Lipschitz continuous with parameter L (say), i.e., $\|\nabla f_d(\beta) - \nabla f_d(\alpha)\| \leq L\|\beta - \alpha\|$ for all $\beta, \alpha \in \mathbb{R}^p$. IHT applied to Problem (2.2) performs the following updates:

$$\beta^{k+1} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2\tau} \|\beta - (\beta^k - \tau \nabla f_d(\beta^k))\|^2 + \lambda_1 \|\beta\|_1 + \lambda_0 \|\beta\|_0 \right\}, \tag{2.9}$$

where $\tau > 0$ is a constant step size. We say that $\alpha \in \mathbb{R}^p$ is a fixed point of update (2.9) if $\beta^k = \alpha$ leads to $\beta^{k+1} = \alpha$. This suggests another notion of stationarity (see Definition 2.5) for Problem (2.2). To this end, consider Theorem 2.1 establishing the convergence of β^k to a fixed point of update (2.9).

Theorem 2.1. *Let L be defined as above. The sequence $\{\beta^k\}$ defined in* (2.9) *converges to a fixed point β^* of update* (2.9) *for any $\tau < \frac{1}{L}$. Note that β^* is a fixed point iff*

$$\begin{aligned} &\beta_i^* = \text{sign}(\widetilde{\beta}_i^*) \tfrac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \quad \textit{and} \quad |\beta_i^*| \geq \sqrt{2\lambda_0 \tau} \quad \textit{for } i \in \textit{Supp}(\beta^*) \\ \textit{and} \qquad &\tfrac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \leq \sqrt{\tfrac{2\lambda_0}{(1+2\lambda_2)^2 \tau}} \qquad \textit{for } i \notin \textit{Supp}(\beta^*) \end{aligned} \tag{2.10}$$

Definition 2.5. *A vector β^* is an IHT minimum for Problem* (2.2) *if it satisfies* (2.10) *for $\tau < \frac{1}{L}$.*

The following remark shows that the class of IHT minima contains the family of CW minima.

Remark 2.1. *Let M be the largest eigenvalue of $X^T X$ and take $L = M + 2\lambda_2$. By Theorem 2.1, any $\tau < \frac{1}{M+2\lambda}$ ensures the convergence of updates* (2.9). *Since the columns of X are normalized, we have $M \geq 1$. Comparing* (2.10) *to* (2.8) *we see that the class of IHT minima includes CW minima. Usually, for high-dimensional problems, $M \gg 1$—making the class of IHT minima much larger than CW minima (see Section 2.5.2 for numerical examples).*

We have now explained the full hierarchy among the classes of local minima presented in (2.4). Section 2.3 discusses algorithms to obtain solutions that belong to these classes.

2.3 Algorithms

Section 2.3.1 presents a cyclic CD algorithm which converges to CW minima, and Section 2.3.2 discusses local combinatorial optimization to obtain PSI(k)/FSI(k) minima.

2.3.1 Cyclic Coordinate Descent

Our main workhorse is cyclic CD [22] with full minimization in every coordinate—we also include some additional tweaks for reasons we discuss shortly. With initialization β^0, we update the first coordinate (with others fixed) to get β^1, and continue the updates as per a cyclic rule. Let β^k denote the solution obtained after performing k coordinate updates. Then, β^{k+1} is obtained by updating the ith coordinate (with others held fixed) via:

$$\beta_i^{k+1} \in \underset{\beta_i \in \mathbb{R}}{\arg\min}\, F(\beta_1^k, \ldots, \beta_{i-1}^k, \beta_i, \beta_{i+1}^k, \ldots, \beta_p^k), \tag{2.11}$$

where $i = (k+1) \bmod (p+1)$. Recall that the operator $\widetilde{T}(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2)$ (defined in (2.7)) describes solutions of Problem (2.11). Specifically, it returns two solutions when $\frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. In such a case, we consistently choose one of these solutions[4], namely the nonzero solution. Thus, we use the new operator (note the use of T instead of $\widetilde{T}$):

$$T(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2) \overset{\text{def}}{=} \begin{cases} \text{sign}(\widetilde{\beta}_i)\frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} & \text{if } \frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ 0 & \text{if } \frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \end{cases} \tag{2.12}$$

for update (2.11). In addition to the above modification, we introduce "spacer steps" that are occasionally performed during the course of the algorithm to *stabilize* its behavior[5]—spacer steps are commonly used in the context of continuous nonlinear optimization problems (e.g., see [22]). We perform a spacer step as follows. Let C be an a-priori fixed positive integer. We keep track of the supports encountered so far, and when a certain support S (say) appears for Cp-many times, we perform one pass of cyclic CD to minimize the continuous function

[4]This convention is used for a technical reason in the context of our proof of Theorem 2.2.

[5]The spacer steps are introduced for a technical reason, and our proof of convergence of CD relies on this to ensure the stationarity of the algorithm's limit points.

$\beta_S \mapsto f(\beta_S)$. This entails updating every coordinate in S via the operator: $T(\widetilde{\beta}_i, 0, \lambda_1, \lambda_2)$ (see Subroutine 2.1).

Algorithm 2.1 summarizes the above procedure. Count[S] is an associative array that stores the number of times a support S appears—it takes S as a key and returns the number of times S has appeared so far. Count[S] is initialized to zero for any S that appears for the first time during the course of the algorithm. Note that in the worst case, storing Count[] may require an exponential (in p) amount of memory. However, in practice, only one or few supports appear for Cp many times and need to be maintained in Count[].

Algorithm 2.1: Coordinate Descent with Spacer Steps (CDSS)

Input : Initial Solution β^0, Positive Integer C
$k \leftarrow 0$
while *Not Converged* **do**
  for i *in* 1 *to* p **do**
    $\beta^{k+1} \leftarrow \beta^k$
    $\beta_i^{k+1} \leftarrow \arg\min_{\beta_i \in \mathbb{R}} F(\beta_1^k, \ldots, \beta_i, \ldots, \beta_p^k)$ using (2.12) `// Non-spacer Step`
    $k \leftarrow k + 1$
    Count[Supp(β^k)] $\leftarrow$ Count[Supp(β^k)] + 1
    if *Count[Supp(β^k)]* $= Cp$ **then**
      $\beta^{k+1} \leftarrow$ SpacerStep(β^k) `// Spacer Step`
      Count[Supp(β^k)] $= 0$
      $k \leftarrow k + 1$

Subroutine 2.1: SpacerStep(β)

Input : β
for i *in* *Supp*(β) **do**
  $\beta_i \leftarrow \arg\min_{\beta_i \in \mathbb{R}} f(\beta_1, \ldots, \beta_i, \ldots, \beta_p)$ using (2.12) with $\lambda_0 = 0$
return (β)

Why Cyclic CD? Cyclic CD has been practically shown to be among the fastest algorithms for Lasso [69] and continuous non-convex regularizers (e.g., MCP, SCAD, etc) [117, 41]. Coordinate updates in cyclic CD have low cost and exploit sparsity via sparse residual updates and active set convergence [69]. This makes it well-suited for high-dimensional problems with $p \gg n$ and p of the order of tens-of-thousands to millions. On the other hand, methods requiring evaluation of the full gradient (e.g., proximal gradient descent, greedy coordinate descent, etc) can have difficulty in scaling with p [133]. For example, proximal gradient de-

scent methods do not exploit sparsity-based structure as well as CD-based methods [69, 133]. We also note that based on our experiments, random CD (proposed by [140] for a problem similar to ours) exhibits slower convergence in practice (see Section 2.5.2)—see also related discussions in [18] for convex problems. We have also observed empirically that cyclic CD has an edge over competing algorithms, both in terms of optimization objective (see Section 2.5.2) and statistical performance (see Sections 2.5.3, 2.5.4, and 2.5.6).

Convergence Analysis

We analyze the convergence behavior of Algorithm 2.1—in particular, we prove a new result establishing convergence to a CW minimum (the limit point depends upon the initialization). Moreover, we show that the linear rate of convergence of CD which holds for minimization of a smooth convex function [18] extends to our non-convex problem (in an asymptotic sense). We note that if we avoid full minimization and use a conservative step size, the proofs of convergence become straightforward by virtue of a sufficient decrease of the objective value after every coordinate update[6]. However, using CD with a conservative step size for Problem 2.2 can have a detrimental effect on the solution quality. By examining the fixed points, it can be shown that a conservative step size leads to a class of stationary solutions that contains CW minima. Cyclic CD has been studied in earlier work with non-convex but continuous regularizers [117, 41, 165] with a least-squares data fidelity term. However, to our knowledge, a convergence analysis of cyclic CD for Problem (2.2) is novel.

We present a few lemmas describing the behavior of Algorithm 2.1 (some additional technical lemmas are in the appendix). Theorem 2.2 establishes convergence and Theorem 2.3 presents an asymptotic linear rate of convergence of Algorithm 2.1.

The following lemma states that Algorithm 2.1 is a descent algorithm.

Lemma 2.5. *Algorithm 2.1 is a descent algorithm and* $F(\beta^k) \downarrow F^*$ *for some* $F^* \geq 0$.

For the remainder of the section, we will make the following minor assumptions to establish

[6]This observation also appears in establishing convergence of IHT-type algorithms—see for example [16, 112, 24].

the convergence of Algorithm 2.1 for the (L_0) and (L_0L_1) problems. These assumptions are not needed for problem (L_0L_2).

Assumption 2.1. *Let $m = \min\{n, p\}$. Every set of m columns of X are linearly independent.*

Assumption 2.2. *(Initialization) If $p > n$, we assume that the initial estimate β^0 satisfies*

- *In the (L_0) problem: $F(\beta^0) \leq \lambda_0 n$.*
- *In the (L_0L_1) problem: $F(\beta^0) \leq f(\beta^{\ell_1}) + \lambda n$ where $f(\beta^{\ell_1}) = \min_\beta \frac{1}{2}\|y - X\beta\|^2 + \lambda_1\|\beta\|_1$.*

The following remark demonstrates that Assumption 2 is rather minor.

Remark 2.2. *Suppose $p > n$ and Assumption 1 holds. For the (L_0) problem, let $S \subseteq [p]$ such that $|S| = n$. If β^0 is defined such that β^0_S is the least squares solution on the support S with $\beta^0_{S^c} = 0$; then $F(\beta^0) = \lambda_0 n$ (since the least squares loss is zero). This satisfies Assumption 2. For the (L_0L_1) problem, we note that there always exists an optimal lasso solution $\hat{\beta}$ such that $\|\hat{\beta}\|_0 \leq n$ (e.g., see [164]). Therefore, $\hat{\beta}$ satisfies Assumption 2.*

In what follows, we assume that Assumptions 2.1 and 2 hold for the (L_0) and (L_0L_1) problems. Lemma 2.6 shows that in the (L_0) and (L_0L_1) problems, the support size of any β^k obtained by Algorithm 2.1 cannot exceed $\min\{n, p\}$.

Lemma 2.6. *For the (L_0) and (L_0L_1) problems, $\{\beta^k\}$ satisfies $\|\beta^k\|_0 \leq \min\{n, p\}$ for all k.*

The following theorem establishes the convergence of Algorithm 2.1.

Theorem 2.2. *The following holds for Algorithm 2.1:*

1. *The support of $\{\beta^k\}$ stabilizes after a finite number of iterations, i.e., there exists an integer m and a support S such that $Supp(\beta^k) = S$ for all $k \geq m$.*
2. *The sequence $\{\beta^k\}$ converges to a CW minimum B with $Supp(B) = S$.*

Theorem 2.3 presents an asymptotic linear rate of convergence for Algorithm 2.1: we make

use of part 1 of Theorem 2.2 and the convergence rate of cyclic CD (for smooth and strongly convex functions) [18]. Note that in Theorem 2.3, *full cycle* refers to a single pass of vanilla CD over all the coordinates in S, and β^K refers to the iterate generated after performing K full cycles of CD.

Theorem 2.3. *[Adaptation of [18], Theorem 3.9] Let $\{\beta^K\}$ be the full-cycle iterates generated by Algorithm 2.1 and B be the limit with support S. Let m_S and M_S denote the smallest and largest eigenvalues of $X_S^T X_S$, respectively. Then, there is an integer N such that for all $K \geq N$ the following holds:*

$$F(\beta^{K+1}) - F(B) \leq \left(1 - \frac{m_S + 2\lambda_2}{2(1+2\lambda_2)\left(1 + |S|(\frac{M_S+2\lambda_2}{1+2\lambda_2})^2\right)}\right)\left(F(\beta^K) - F(B)\right). \tag{2.13}$$

2.3.2 Local Combinatorial Optimization Algorithms

We present algorithms to obtain solutions belonging to the classes of Swap Inescapable minima introduced in Section 2.2.3.

Algorithms for PSI minima

We introduce an iterative algorithm that leads to a PSI(k) minimum. In the ℓth iteration, the algorithm performs two steps: (1) runs Algorithm 2.1 to get a CW minimum β^ℓ, and (2) searches for a "descent move" by solving the following combinatorial optimization problem:

$$\min_{\beta, S_1, S_2} \quad F(\beta^\ell - U^{S_1}\beta^\ell + U^{S_2}\beta) \quad \text{s.t.} \quad S_1 \subseteq S,\ S_2 \subseteq S^c, |S_1| \leq k,\ |S_2| \leq k, \tag{2.14}$$

where $S = \text{Supp}(\beta^\ell)$. Note that if there is a feasible solution $\hat{\beta}$ to Problem (2.14) satisfying $F(\hat{\beta}) < F(\beta^\ell)$, then $\hat{\beta}$ may not be a CW minimum. In this case, Algorithm 2.1 can be initialized with $\hat{\beta}$ to obtain a better solution for Problem (2.2). Otherwise, if such a $\hat{\beta}$ does not exist, then β^ℓ is a PSI(k) minimum (by Definition 2.3). Algorithm 2.2 (aka CD-PSI(k)) summarizes the algorithm.

Algorithm 2.2: CD-PSI(k)

$\hat{\beta}^0 \leftarrow \beta^0$
for $\ell = 0, 1, \ldots$ **do**
  $\beta^{\ell+1} \leftarrow$ Output of Algorithm 2.1 initialized with $\hat{\beta}^\ell$
  if *Problem (2.14) has a feasible solution* $\hat{\beta}$ *satisfying* $F(\hat{\beta}) < F(\beta^{\ell+1})$ **then**
    $\hat{\beta}^{\ell+1} \leftarrow \hat{\beta}$
  else
    Terminate

Theorem 2.4. *Let* $\{\beta^\ell\}$ *be the sequence of iterates generated by Algorithm 2.2. For the* (L_0) *and* (L_0L_1) *problems, suppose that Assumptions 2.1 and 2 hold. Then, Algorithm 2.2 terminates in a finite number of iterations and the output is a PSI*(k) *minimum.*

As indicated in Theorem 2.4, Algorithm 2.2 terminates in a finite number of iterations (which depends upon the number of swaps that improve the objective value). In our experiments, Algorithm 2.2 typically terminates in less than 20 iterations. Each iteration involves a call to Algorithm 2.1 (which is quite fast) followed by solving a feasibility problem (i.e., finding $\hat{\beta}$ as described in Algorithm 2.2). As we discuss next, for moderate p (e.g., 10^3 to 10^4), this feasibility problem can be handled efficiently using MIO solvers. When $k = 1$, we propose a specialized algorithm for solving this feasibility problem that can easily scale to problems with $p = 10^6$.

MIO formulation for Problem (2.14)**:** Problem (2.14) admits an MIO formulation given

by:

$$\min_{\theta,\beta,z} \quad f(\theta) + \lambda_0 \sum_{i\in[p]} z_i \tag{2.15a}$$

$$\text{s.t.} \quad \theta = \beta^\ell - \sum_{i\in S} e_i\beta_i^\ell(1-z_i) + \sum_{i\in S^C} e_i\beta_i \tag{2.15b}$$

$$-\mathcal{M}z_i \le \beta_i \le \mathcal{M}z_i, \quad \forall i \in S^c \tag{2.15c}$$

$$\sum_{i\in S^c} z_i \le k \tag{2.15d}$$

$$\sum_{i\in S} z_i \ge |S| - k \tag{2.15e}$$

$$\beta_i \in \mathbb{R}, \quad \forall i \in S^c \tag{2.15f}$$

$$z_i \in \{0,1\}, \quad \forall i \in [p], \tag{2.15g}$$

where the optimization variables are $\theta \in \mathbb{R}^p$, $\beta_i, i \in S^c$, and $z \in \{0,1\}^p$. In formulation (2.15), $S = \text{Supp}(\beta^\ell)$, where β^ℓ is fixed, and $\mathcal{M}$ is a Big-M parameter (a-priori specified) controlling the L_∞-norm of β_{S^c}. Any sufficiently large value of $\mathcal{M}$ will lead to a solution for Problem (2.14); however, a tight choice for $\mathcal{M}$ affects the run time of the MIO solver—see [24] for additional details. We note that the $\|\theta\|_1$ term included in $f(\theta)$ can be expressed via linear inequalities using auxiliary variables. Thus, Problem (2.15) is a mixed integer quadratic optimization (MIQO) problem.

We now explain the constraints in Problem (2.15) and how they relate to Problem (2.14). To this end, let S_1 and S_2 be subsets defined in (2.14). Let $\theta = \beta^\ell - U^{S_1}\beta^\ell + U^{S_2}\beta$ and this relation is expressed in (2.15b). Let us consider any binary variable z_i where $i \in S$. If $z_i = 0$ then β_i^ℓ is removed from S, and we have $\theta_i = 0$ (see (2.15b)). If $z_i = 1$, then β_i^ℓ is not removed from θ, and we have $\theta_i = \beta_i^\ell \neq 0$ (see (2.15b)). Note that $|S_1| = \sum_{i\in S}(1-z_i) = |S| - \sum_{i\in S} z_i$. The condition $|S_1| \le k$, is thus encoded in the constraint $\sum_{i\in S} z_i \ge |S| - k$ in (2.15e). Thus we have that $\|\theta_S\|_0 = \sum_{i\in S} z_i$.

Now consider any binary variable z_i where $i \in S^c$. If $z_i = 1$, then by (2.15c) we observe that β_i is free to vary in $[-\mathcal{M}, \mathcal{M}]$. This implies that $\theta_i = \beta_i$. If $z_i = 0$ then $\theta_i = \beta_i = 0$.

Note $\sum_{i \in S^c} z_i = |S_2|$, and the constraint $|S_2| \leq k$ is expressed via $\sum_{i \in S^c} z_i \leq k$ in (2.15d). It also follows that $\|\theta_{S^c}\|_0 = \sum_{i \in S^c} z_i$. Finally, we note that the function appearing in the objective (2.15a) is $F(\theta)$, since $\lambda_0 \sum_{i \in [p]} z_i = \lambda_0 \|\theta\|_0$.

Remark 2.3. *Problem* (2.15) *has a smaller (combinatorial) search space compared to an MIO formulation for the full Problem* (2.2)*—solving* (2.15) *for small values of k is usually much faster than Problem* (2.2)*. Furthermore, an MIO framework can quickly deliver a feasible solution to Problem* (2.15) *with a smaller objective than the current solution—this is usually much faster than establishing optimality via dual bounds. Note that the MIO-framework can also certify (via dual bounds) if there is no feasible solution with a strictly smaller objective value.*

Section 2.5 presents examples where Problem (2.15) leads to higher quality solutions—from both the optimization and statistical performance viewpoints. We now present an efficient algorithm for solving the special case of Problem (2.14) with $k = 1$.

An efficient algorithm for computing PSI(1) minima. Subroutine 2.2 presents an algorithm for Problem (2.14) with $k = 1$. That is, we search for a feasible solution $\hat{\beta}$ of Problem (2.14) satisfying $F(\hat{\beta}) < F(\beta^\ell)$. The two for loops in Subroutine 2.2 can run

Subroutine 2.2: Vanilla Implementation of Problem (2.14) with $k = 1$.

$S \leftarrow \text{Supp}(\beta^\ell)$
for $i \in S$ **do**
  for $j \in S^c$ **do**
    $v_j^* \leftarrow \underset{v_j \in \mathbb{R}}{\arg\min}\, F(\beta^\ell - e_i \beta_i^\ell + e_j v_j)$ (2.16)
    $F_j^* \leftarrow F(\beta^\ell - e_i \beta_i^\ell + e_j v_j^*)$ (2.17)
  $\vartheta \leftarrow \underset{j \in S^c}{\arg\min}\, F_j^*$ (2.18)
  if $F_\vartheta^* < F(\beta^\ell)$ **then**
    $\hat{\beta} \leftarrow \beta^\ell - e_i \beta_i^\ell + e_\vartheta v_\vartheta^*$ (2.19)
    Terminate

for a total of $(p - \|\beta^\ell\|_0)\|\beta^\ell\|_0$ iterations, where every iteration requires $O(n)$ operations to perform the minimization in (2.16) and evaluate the new objective in (2.17). Therefore,

Subroutine 2.2 entails an overall cost of $O\Big(n(p - \|\beta^\ell\|_0)\|\beta^\ell\|_0\Big)$. However, we show below that a careful implementation can reduce the cost by a factor of n; leading to a cost of $O\Big((p - \|\beta^\ell\|_0)\|\beta^\ell\|_0\Big)$-many operations.

A solution v_j^* of Problem (2.16) is given by

$$v_j^* = \begin{cases} \text{sign}(\bar{\beta}_j)\frac{|\bar{\beta}_j| - \lambda_1}{1+2\lambda_2} & \text{if} \quad \frac{|\bar{\beta}_j| - \lambda_1}{1+2\lambda_2} \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

where

$$\bar{\beta}_j = \langle r + X_i\beta_i^\ell, X_j\rangle = \langle r, X_j\rangle + \langle X_i, X_j\rangle\beta_i^\ell, \tag{2.21}$$

and $r = y - X\beta^\ell$. We note that in Algorithm 2.2, solving (2.14) is directly preceded by a call to Algorithm 2.1. The quantities $\langle r, X_j\rangle$ and $\langle X_i, X_j\rangle$ appearing on the right-hand side of (2.21) can be stored during the call to Algorithm 2.1 (these two quantities are computed by CD as part of the 'covariance updates'—see Section 2.4 for details). By reusing these two stored quantities, we can compute every $\bar{\beta}_j$ (and consequently v_j^*) in $O(1)$ arithmetic operations.

Furthermore, the following equivalent representations hold:

$$\arg\min_{j\in S^c} F_j^* \iff \arg\max_{j\in S^c} |v_j^*| \tag{2.22}$$

$$F_\vartheta^* < F(\beta^l) \iff |v_\vartheta^*| > |\beta_i^l|. \tag{2.23}$$

Thus, we can avoid the computation of the objective F_j^* in (2.17) and replace (2.18) with $\vartheta \leftarrow \arg\max_{j\in S^c}|v_j^*|$. Furthermore, we can replace $F_\vartheta^* < F(\beta^\ell)$ (before equation (2.19)) with $|v_\vartheta^*| > |\beta_i^l|$. We summarize these changes in Subroutine 2.3, which is the efficient counterpart of Subroutine 2.2. Note that Subroutine 2.3 has a cost of $O\Big((p - \|\beta^l\|_0)\|\beta^l\|_0\Big)$ operations.

Remark 2.4. *Since CD-PSI(1) (Algorithm 2.2 with $k = 1$) is computationally efficient, in Algorithm 2.2 (with $k > 1$), CD-PSI(1) may be used to replace Algorithm 2.1. In our*

Subroutine 2.3: Efficient Implementation of Problem (2.14) with $k = 1$.

$S \leftarrow \text{Supp}(\beta^\ell)$
for $i \in S$ **do**
  for $j \in S^c$ **do**
    Compute v_j^* in $O(1)$ using (2.20)
  $\vartheta \leftarrow \arg\max_{j \in S^c} |v_j^*|$
  if $|v_\vartheta^*| > |\beta_i^\ell|$ **then**
    $\hat{\beta} \leftarrow \beta^\ell - e_i \beta_i^\ell + e_\vartheta v_\vartheta^*$
    Terminate

numerical experiments, this is found to work well in terms of lower run times and also in obtaining higher-quality solutions (in terms of objective values). This modification also guarantees convergence to a PSI(k) minimum (as the proof of Theorem 2.4 still applies to this modified version).

Algorithm for FSI minima

To obtain a FSI(k) minimum, Problem (2.14) needs to be modified—we replace optimization w.r.t. the variable $U^{S_2}\beta$ by that of $U^{(S\setminus S_1)\cup S_2}\beta$. This leads to the following problem:

$$\min_{\beta, S_1, S_2} \quad F(\beta^\ell - U^{S_1}\beta^\ell + U^{(S\setminus S_1)\cup S_2}\beta) \quad \text{s.t.} \qquad S_1 \subseteq S,\ S_2 \subseteq S^c, |S_1| \leq k,\ |S_2| \leq k, \quad (2.24)$$

where $S = \text{Supp}(\beta^l)$. Similarly, Algorithm 2.2 gets modified by considering Problem (2.24) instead of Problem (2.14). By the same argument used in the proof of Theorem 2.4, this modification guarantees that Algorithm 2.2 converges in a finite number of iterations to a FSI(k) minimum.

We present an MIO formulation for Problem (2.24). We write $\theta = \beta^\ell - U^{S_1}\beta^\ell + U^{(S\setminus S_1)\cup S_2}\beta$ and use a binary variable $w_i, i \in S$ to indicate whether $i \in S_1$ or not: we set $w_i = 0$ iff $i \in S_1$. We use another binary variable $z_i, i \in [p]$ to indicate the number of nonzeros in θ,

i.e., $z_i = 0 \Rightarrow \theta_i = 0$. This leads to the following MIO problem:

$$\min_{\theta,w,z} \quad f(\theta) + \lambda_0 \sum_{i\in[p]} z_i \tag{2.25a}$$

$$-\mathcal{M} z_i \leq \theta_i \leq \mathcal{M} z_i, \quad \forall i \in [p] \tag{2.25b}$$

$$z_i \leq w_i, \quad \forall i \in S \tag{2.25c}$$

$$\sum_{i\in S^c} z_i \leq k \tag{2.25d}$$

$$\sum_{i\in S} w_i \geq |S| - k \tag{2.25e}$$

$$\theta_i \in \mathbb{R}, \quad \forall i \in [p] \tag{2.25f}$$

$$z_i \in \{0,1\}, \quad \forall i \in [p] \tag{2.25g}$$

$$w_i \in \{0,1\} \quad \forall i \in S. \tag{2.25h}$$

In (2.25b), for every $i \in [p]$, the binary variable $z_i = 1$ if $i \in \text{Supp}(\beta)$, and $\mathcal{M}$ is a sufficiently large constant (similar to that in (2.15)). The second term in the objective (2.25a) stands for $\lambda_0 \sum_i z_i = \lambda_0 \|\theta\|_0$. In (2.25c) we enforce the condition that if $w_i = 0$ then $z_i = 0$ implying that $\theta_i = 0$; If $w_i = 1$ then $i \notin S_1$. Coordinates in $S \setminus S_1$ (i.e., those with $w_i = 1$) are free to be inside or outside of $\text{Supp}(\theta)$. We enforce $|S_1| \leq k$ and $|S_2| \leq k$ via (2.25e) and (2.25d), respectively.

Remark 2.5. *Formulation* (2.25) *has a larger search space compared to formulation* (2.15) *of PSI minima, due to the additional number of continuous variables. While this leads to increased run times compared to Problem* (2.15)*, it can still be solved faster than an MIO formulation for the full Problem* (2.2) *(for the same reasons as in Remark 2.3).*

In Section 2.5.5, we present experiments where we compare the quality of FSI(k) minima, for different values of k, to the other classes of minima.

2.4 Efficient Computation of the Regularization Path

We designed `L0Learn` [88][7]: an extensible C++ toolkit with an R interface that implements most of the algorithms discussed in this chapter. Our toolkit achieves lower running times[8] compared to other popular sparse learning toolkits (e.g., `glmnet` and `ncvreg`) by utilizing a series of computational tricks and heuristics. These include an adaptive grid of tuning parameters, continuation, active-set updates, greedy cyclic ordering of coordinates, correlation screening, and a careful accounting of floating point operations—some of these heuristics (as specified below) appear in prior work for deriving highly efficient algorithms for the Lasso (e.g., `glmnet`). Recently, [152] studied different exact and approximate approaches for L0-regularized regression, and their conclusion is: "Overall, the regularized `L0Learn` solutions exceed the performance of cardinality-constrained algorithms while being significantly faster and more convenient".

Below, we provide a detailed account of the computational strategies used in `L0Learn`.

Adaptive Selection of Tuning Parameters: We use continuation on a grid of λ_0 values: $\lambda_0^1 > \lambda_0^2 > \cdots > \lambda_0^m$ and use the solution obtained from λ_0^k as a warm start for λ_0^{k+1}. The choice of λ_0^i's requires care so we present a new method to select this sequence. If two successive values of the λ_0 sequence are far apart, one might miss good solutions. However, if these successive values are too close, the corresponding solutions will be identical. To avoid this problem, we derive conditions on the choice of λ_0 values which ensure that Algorithm 2.1 leads to different solutions. To this end, we present the following lemma, wherein we assume that λ_1, λ_2 are a-priori fixed.

Lemma 2.7. *Suppose $\beta^{(i)}$ is the output of Algorithm 2.1 with $\lambda_0 = \lambda_0^i$. Let $S = Supp(\beta^{(i)})$,*

[7] Available on CRAN at `https://CRAN.R-project.org/package=L0Learn` and on github at `https://github.com/hazimehh/L0Learn`

[8] Problem (2.2) usually leads to solutions with fewer nonzeros compared to Lasso and MCP penalized regression. This also contributes to reduced run times.

$r = y - X\beta^{(i)}$ denote the residual, and

$$M^i = \frac{1}{2(1+2\lambda_2)} \max_{j \in S^c} \Big(|\langle r, X_j \rangle| - \lambda_1 \Big)^2. \tag{2.26}$$

Then, running Algorithm 2.1 for $\lambda_0^{i+1} < \lambda_0^i$ initialized at $\beta^{(i)}$ leads to a solution $\beta^{(i+1)}$ satisfying: $\beta^{(i+1)} \neq \beta^{(i)}$ if $\lambda_0^{i+1} < M^i$, and $\beta^{(i+1)} = \beta^{(i)}$ if $\lambda_0^{i+1} \in (M^i, \lambda_0^i]$.

Lemma 2.7 suggests a simple scheme to compute a sequence $\{\lambda_0^i\}$ that avoids duplicate solutions. Suppose we have computed the regularization path up to $\lambda_0 = \lambda_0^i$, then λ_0^{i+1} can be computed as $\lambda_0^{i+1} = \alpha M^i$, where α is a fixed scalar in $(0, 1)$. Moreover, we note that M^i (defined in (2.26)) can be computed without explicitly calculating $\langle r, X_i \rangle$ for every $i \in S^c$, as these dot products can be maintained in memory while running Algorithm 2.1 with $\lambda_0 = \lambda_0^i$. Therefore, computing M^i, and consequently λ_0^{i+1}, requires only $O(|S^c|)$ operations.

(Partially) Greedy Cyclic Order: Suppose Algorithm 2.1 is initialized with a solution β^0 and let $r^0 = y - X\beta^0$. Before running Algorithm 2.1, we reorder the coordinates based on sorting the quantities $|\langle r^0, X_i \rangle|$ for $i \in [p]$ in descending order[9]. In practice, we perform *partial sorting*, in which only the top t (e.g., $t = 5000$ and p is much larger) coordinates are sorted, while the rest maintain their initial order. This is typically faster and equally effective compared to sorting all coordinates. Note that this ordering is performed *once* before the start of Algorithm 2.1—this is different from greedy CD that finds the maximal correlation at every coordinate update. Our experiments in Section 2.5.2 indicate that (partially) greedy cyclic order performs significantly better than a vanilla cyclic order or random order.

Correlation Screening: When using continuation, we perform a screening method inspired by [163][10]. We restrict the updates of Algorithm 2.1 to the support of the warm start in addition to a small portion (e.g., top 10^3) of other coordinates that are highly correlated with the current residuals—these coordinates are readily available as a byproduct of the (partially) greedy cyclic ordering rule, described above. After convergence on the screened support, we check if any coordinate from outside the support violates the conditions of a

[9]Since the columns of X have unit L_2-norm, updating index $\arg\max_i |\langle r^0, X_i \rangle|$ will lead to the maximal decrease in the objective function.

[10]Our approach differs from [163] who derive screening rules for convex problems.

CW minimum and rerun the algorithm if needed. Typically, the solution obtained from the screened set turns out to be a CW minimum, and only one pass is done over all the coordinates.

Active Set Updates: As in prior work [69], active set methods are found to be very useful in our context as well. Empirically, the iterates generated by Algorithm 2.1 can typically achieve support stabilization in less than 10 full cycles[11]. This is further supported by Theorem 2.2, which establishes finite-time stabilization of the support. If the support does not change across multiple consecutive full cycles, we restrict the updates of Algorithm 2.1 to the current support. After convergence on this support, we check whether any coordinate from outside the support violates the conditions of a CW minimum and rerun the algorithm if needed.

Fast Coordinate Updates: Following [69], we present techniques for efficiently computing the coordinate updates—these are also found to be useful for our implementation of PSI(1).

Let β^k be the current iterate in Algorithm 2.1 and r^k be the residuals. To compute $\widetilde{\beta}_i = \langle r^k, X_i \rangle + \beta_i^k$, one can use one of the following rules that exploit sparsity [69]: (i) *Residual Updates:* We maintain the residuals r^k throughout the algorithm and compute $\widetilde{\beta}_i$ using $O(n)$ operations. Once β_i^{k+1} is computed, we update r^{k+1} with cost $O(n)$ operations. Since β^k is sparse, many of the coordinates remain at zero during the algorithm implying that $r^{k+1} = r^k$. (ii) *Covariance Updates:* This appears in [69] for updating $\widetilde{\beta}_i$ without using the precomputed r^k. Note that Algorithm 2.1 precomputes $\langle y, X_i \rangle$ for all $i \in [p]$. If a coordinate ℓ enters the support for the first time, we compute and store the covariance terms: $\langle X_\ell, X_j \rangle$ for all $j \in [p]$ with cost $O(np)$. In iteration $k+1$, we compute $\widetilde{\beta}_i$ using these covariances and exploiting the sparsity of β^k—this costs $O(\|\beta^k\|_0)$ operations. This costs less than computing $\widetilde{\beta}_i$ using rule (i) if $\|\beta^k\|_0 < n$.

Scheme (ii) is useful when the supports encountered by Algorithm 2.1 are small w.r.t. n. It is also useful for an efficient implementation of the PSI(1) algorithm as it stores the dot

[11] Recall, one full cycle refers to updating all the p coordinates in a cyclic order.

products required in (2.21). However, when the supports encountered by CD are relatively large compared to n (e.g., 10% of n), then Scheme (i) can become significantly faster since the dot product computations can be accelerated using calls to the Basic Linear Algebra Subprograms (BLAS).

2.5 Computational Experiments

In this section, we investigate both the optimization and statistical performances of our proposed algorithms and compare them to other popular sparse learning algorithms. For convenience, we provide a road map of this section. Section 2.5.2 compares the optimization performance of our proposed algorithms and other variants of CD and IHT. Section 2.5.3 empirically studies the statistical properties of estimators available from our proposed algorithms versus others for varying sample sizes. Section 2.5.4 provides a similar study for varying SNR. Section 2.5.5 performs an in-depth investigation among the PSI(k)/FSI(k) algorithms, for different values of k. Section 2.5.6 presents timing and statistical performance comparisons on some large-scale instances, including real datasets.

2.5.1 Experimental Setup

Data Generation: We consider a series of experiments on synthetic datasets for a wide range of problem sizes and designs. We generate a multivariate Gaussian data matrix $X_{n\times p} \sim \mathrm{MVN}(0, \Sigma)$. We use a sparse coefficient vector $\beta^\dagger$ with $k^\dagger$ equi-spaced nonzero entries, each set to 1. We then generate the response vector $y = X\beta^\dagger + \epsilon$, where $\epsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$ is independent of X. We define the signal-to-noise ratio (SNR) by $\mathrm{SNR} = \frac{\mathrm{Var}(X\beta^\dagger)}{\mathrm{Var}(\epsilon)} = \frac{\beta^{\dagger T}\Sigma\beta^\dagger}{\sigma^2}$. An alternative to the SNR is the "proportion of variance explained" or R^2 for the *true model*: $R^2 = 1 - \frac{\mathrm{Var}(y - X\beta^\dagger)}{\mathrm{Var}(y)} = \frac{\mathrm{SNR}}{\mathrm{SNR}+1}$.

We consider the following instances of $\Sigma := ((\sigma_{ij}))$:

- **Constant Correlation**: We set $\sigma_{ij} = \rho$ for every $i \neq j$ and $\sigma_{ii} = 1$ for all $i \in [p]$.
- **Exponential Correlation**: We set $\sigma_{ij} = \rho^{|i-j|}$ for all i, j, with the convention $0^0 = 1$.

We select the tuning parameters by minimizing the prediction error on a separate validation set, which is generated under the fixed design setting as $y' = X\beta^\dagger + \epsilon'$, where $\epsilon'_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$. In Appendix 2.B, we also include alternative validation strategies. Particularly, we include the results of the experiments in Sections 2.5.3 and 2.5.4 based on both oracle and random design tuning (following [83]). The results are found to be quite similar.

Competing Algorithms and Parameter Tuning: In addition to our proposed algorithms, we compare the following state-of-the-art methods in the experiments:

- **Lasso**: We use our own implementation and in the figures we denote it by "L_1".
- **Relaxed Lasso**: We use the Relaxed Lasso version suggested in [83], defined as $\beta^{\text{relaxed}} = \gamma\beta^{\text{lasso}} + (1-\gamma)\beta^{\text{LS}}$, where β^{lasso} is the Lasso estimate, the nonzero components of β^{LS} are given by the least squares solution on the support of β^{lasso}, and $\gamma \in [0, 1]$ is a second tuning parameter (in addition to the tuning parameter for the Lasso). We use our own implementation for relaxed lasso and denote it by "L1Relaxed".
- **Elastic Net**: This uses a combination of the L_1 and L_2 regularization [193]. We use the implementation of `glmnet` and refer to it as "L_1L_2".
- **MCP**: This is the MCP penalty of [185]. We use the implementation provided in `ncvreg` [41].
- **Forward Stepwise**: We use the implementation of [83], and denote it by "FStepwise".
- **IHT**: We use our implementation for IHT for the (L_0) problem with a step size of M^{-1}, where M is defined in Remark 2.1.

For all the methods involving one tuning parameter, we tune over a grid of 100 parameter values, except for forward stepwise selection which we allow to run for up to 250 steps. For the methods with two parameters, we tune over a 2-dimensional grid of 100×100 values. For our algorithms, the tuning parameter λ_0 is generated as per Section 2.4. For the (L_0L_2) problem, we sweep λ_2 between $0.1\lambda_2^*$ and $10\lambda_2^*$, where λ_2^* is the optimal regularization parameter for ridge regression (based on validation over 100 grid points between 0.0001 and

1000) . For (L_0L_1), Lasso, Relaxed Lasso, and Elastic Net, we sweep λ_1 from $\|X^T y\|_\infty$ down to $0.0001 \times \|X^T y\|_\infty$. For Relaxed Lasso and Elastic Net, we sweep their second parameter between 0 and 1. For MCP, the range of the first parameter λ is chosen by `ncvreg`, and we sweep the second parameter γ between 1.5 and 25.

Performance Measures: We use several metrics to evaluate the quality of an estimator $\hat{\beta}$ (say). In addition to the objective value, number of true positives (TP), false positives (FP), and support size, we consider the following measures:

- **Prediction Error**: This is the same measure used in [24] and is defined by $\|X\hat{\beta} - X\beta^\dagger\|^2/\|X\beta^\dagger\|^2$. The prediction error of a perfect model is 0 and that of the null model ($\hat{\beta} = 0$) is 1.

- L_∞ **Norm**: This is the estimation error measured in terms of the L_∞-norm: $\|\hat{\beta} - \beta^\dagger\|_\infty$.

- **Full support recovery**: We study if the support of $\beta^\dagger$ is completely recovered by $\hat{\beta}$, i.e., $\mathbf{1}[\text{Supp}(\beta^\dagger) = \text{Supp}(\hat{\beta})]$ — we look at the average of this quantity across multiple replications, leading to an estimate for the probability of full support recovery.

We would like to point out that SNR alone does not dictate how difficult the underlying statistical problem is (e.g., in terms of variable selection, estimation, or prediction error). The situation is rather subtle: in addition to SNR, the matrix X and choices of $n, p, k^\dagger$ influence the statistical performance. For example, consider two instances with $p = 100$ and $p = 10^5$ where we set $k^\dagger = 10, \Sigma = I, n = 100, \text{SNR} = 1$. For $p = 100$, it is possible to obtain a model with estimation error smaller than the null model (with high probability), but this may not be possible for $p = 10^5$. Similarly, for the case of constant correlation, a problem with $\rho = 0, \text{SNR} = 1$ may be statistically easier than one with $\rho = 0.5, \text{SNR} = 100$ (with suitable choices of $n, p, k^\dagger$). The experiments that follow are intended to provide a solid understanding of how support recovery, sparsity, estimation error, and prediction error vary under different problem settings. We also seek to understand (i) when our algorithms can achieve full support recovery—a topic of considerable importance in high-dimensional statistics [70, 172]; and (ii) when pure L_0 starts to overfit (a deeper statistical understanding

of this seems to be in a nascent stage).

2.5.2 Comparison between CD variants and IHT: Optimization Performance

We investigate the optimization performance of the different algorithms for the (L_0) problem. Particularly, we study the objective values and the number of iterations till convergence for IHT and the following variants of CD:

- **Cyclic CD**: This is Algorithm 2.1 with default cyclic order.
- **Random CD**: This is a randomized version of CD, where the coordinates to be updated are chosen uniformly at random from $[p]$. This has been considered in [140].
- **Greedy Cyclic CD**: This is our proposed Algorithm 2.1 with a partially greedy cyclic ordering of coordinates, described in Section 2.4.

We generated a dataset with Exponential Correlation, $\rho = 0.5$, $n = 500$, $p = 2000$, SNR $= 10$, and a support size $k^\dagger = 100$. We generated 50 random initializations each with a support size of 100, where the nonzero indices are selected uniformly at random in 1 to p and assigned values that are drawn from Uniform$(0, 1)$. For every initialization, we ran Cyclic CD, Greedy Cyclic CD, and IHT and recorded the value of the objective function of the solution along with the number of iterations (here, one full pass over all p coordinates is defined as one iteration) till convergence. For Random CD, we ran the algorithm 10 times for every initialization and averaged the objective values and number of iterations. For all the algorithms above, we declare convergence when the relative change in the objective is $< 10^{-7}$. Figure 2-1 presents the results: the objective values resulting from Greedy Cyclic CD are significantly lower than the other methods; on average we have roughly a 12% improvement in the objective from Random CD and 55% improvement over IHT. This finding can be partially explained in the light of our discussion in Section 2.2.4, where we observed that the Lipschitz constant L controls the quality of the solutions returned by IHT. In this high-dimensional setting, $L \approx 11$ which is far from 1, and thus IHT can get stuck in relatively weak local minima. The number of iterations till convergence is also in favor of Greedy Cyclic

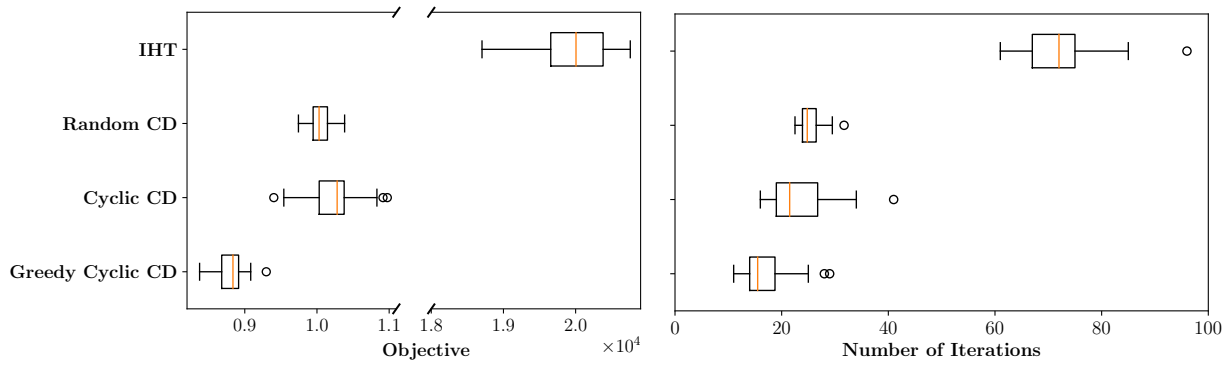


Figure 2-1: Box plots showing the distribution of the objective values and the number of iterations till convergence for different variants of CD and IHT. The ticks of the box plots represent 1.5 times the inter-quartile range.

CD, which requires roughly 28% less iterations than Random CD and 75% less iterations than IHT.

2.5.3 Statistical Performance for Varying Sample Sizes

We study how the different statistical metrics change with the number of samples, while the other factors $(p, k^\dagger, \text{SNR}, \Sigma)$ are fixed. We seek to empirically validate our hypothesis that: *Under difficult statistical settings (e.g., high correlation or a small value of n), advanced optimization techniques such as combinatorial search, can lead to significantly improved statistical performance.*

We consider Algorithms 2.1 and 2.2 (with $k = 1$) for the (L_0), (L_0L_1), and (L_0L_2) problems; in addition to the competing penalties discussed in Section 2.5.1. In Experiments 1 and 2 (below), we swept n between 100 and 1000, and for every value of n, we generated 20 random training and validation datasets having Exponential Correlation, $p = 1000$, $k^\dagger = 20$, and SNR=5. All the results we report here are based on validation-set tuning. In Appendix 2.B, we include the results for oracle and random design tuning.

Experiment 1: High Correlation

Here we choose $\rho = 0.9$ (exponential correlation)—this is a difficult problem due to the high correlations among features in the sample. Figure 2-2 summarizes the results. In the

top panel of Figure 2 we show the results of Algorithm 2.2 applied to the (L_0) and (L_0L_2) problems versus the other competing algorithms. In the bottom panel, we present a detailed comparison among Algorithms 2.1 and 2.2 for all the three problems: (L_0), (L_0L_1) and (L_0L_2).

From the top panel (Figure 2-2), we can see that Algorithm 2.2 applied to the (L_0L_2) problem overall achieves the best performance in terms of different measures across all values of n. Algorithm 2.2 (L_0) and Algorithm 2.2 (L_0L_2) perform similarly for $n \geq 300$. The probability of full recovery for Algorithm 2.2 increases with n and becomes 1 at around $n = 900$—note that the slight variation in the recovery probability values for our methods are solely due to the validation procedure (the oracle tuning presented in Appendix 2.B eliminates this wiggly behavior). Lasso, Relaxed Lasso, and Elastic Net do not achieve full support recovery for any n—the corresponding lines are aligned with the horizontal axis. The L_1-based methods also lead to large support sizes—as expected, L_1L_2 leads to supports that are denser than Lasso [193]. Due to shrinkage, tuning parameter selection based on prediction error makes the Lasso select models with many nonzero coefficients—this leads to sub-optimal variable selection. The Relaxed Lasso attempts to undo the shrinkage effect of the Lasso leading to models with fewer nonzeros. In addition, we note that shrinkage of Lasso also interferes with variable selection—a shortcoming that is inherited by the Relaxed Lasso—as seen in the panel displaying recovery probability.

Moreover, MCP, FStepwise, and IHT have a probability of recovery around 0.3 even when $n = p = 1000$—suggesting that they fail to do correct support recovery in this regime. A similar phenomenon occurs for the prediction error and the L_∞ norm, where Algorithm 2.2 is seen to dominate.

The bottom figure shows that Algorithm 2.2 can significantly outperform Algorithm 2.1 (which performs no swaps). It seems that in this highly correlated setting, our local combinatorial optimization procedures have an edge in performance.

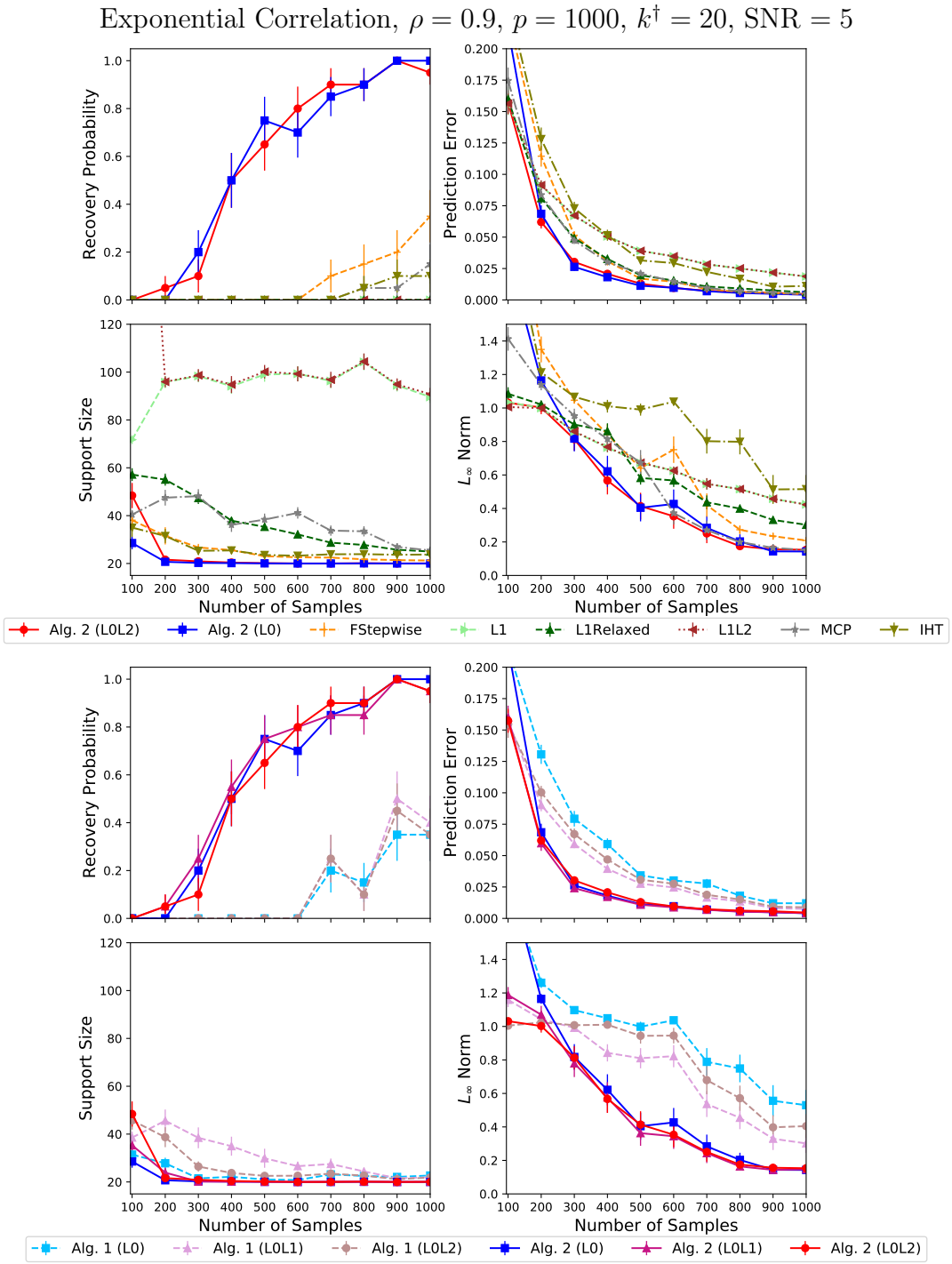


Figure 2-2: Performance measures as the number of samples n varies between 100 and 1000. The top figure compares two of our methods (Algorithm 2.2 for the (L_0) and (L_0L_2) problems) with other state-of-the-art algorithms. The bottom figure compares Algorithms 2.1 and 2.2 for all three problems. Algorithm 2.2 performs significantly better than Algorithm 2.1 and the other methods, since it does a better job at optimization.

Experiment 2: Mild Correlation

In this experiment, we keep the same setup as the previous experiment, but we reduce the correlation parameter ρ to 0.5. In Figure 2-3, we show the results for Algorithm 2.1 applied to (L_0) and Algorithm 2.2 applied to (L_0L_2) versus the other competing methods. We note that our other algorithms have a similar profile (we do not include their plots for space constraints). This setup is easier from a statistical perspective, when compared to

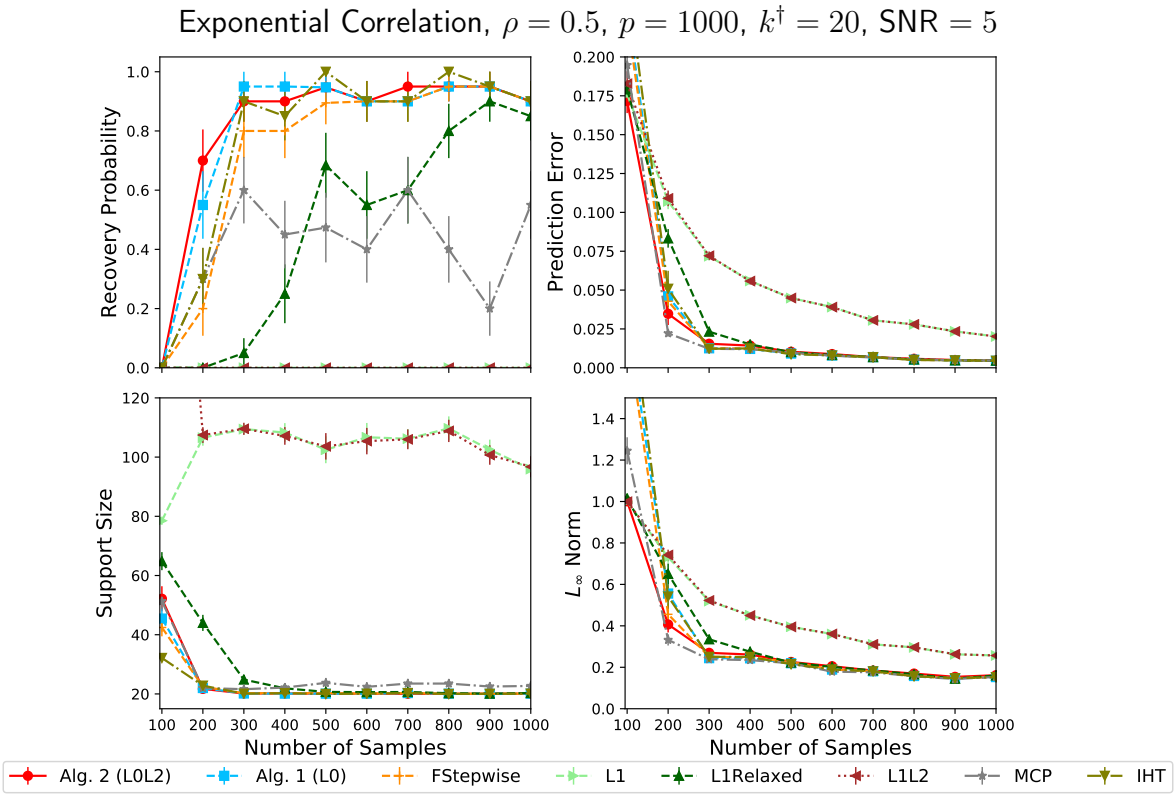


Figure 2-3: Performance measures as the number of samples n varies between 100 and 1000. The figure compares two of our methods (Algorithm 2.2 (L_0L_2) and Algorithm 2.1 for (L_0)) with other state-of-the-art algorithms. Algorithms 2.1 and 2.2 perform similarly in this case (in contrast to the highly correlated setting in Figure 2-2). Adding L_1 or L_2 regularization to (L_0) does not help in this case.

Experiment 1 where $\rho = 0.9$. Thus, we expect all the methods to perform better (overall) and display a phase-transition (for recovery probability) at a smaller sample size (compared to Experiment 1). Indeed, as shown in Figure 2-3, Algorithm 2.1 (L_0) and Algorithm 2.2 (L_0L_2) have roughly the same profiles, and they outperform the other methods; they fully recover the true support using roughly 300 samples. The swap variants of our methods in this case

do not seem to lead to significant improvements over the non-swap variants, and this further supports our hypothesis: when the statistical problem is relatively easy, Algorithm 2.1 works quite well—more advanced optimization (e.g., using swaps) do not seem necessary. MCP and FStepwise also exhibit good performance, but they start doing full support recovery for much larger values of n; and MCP does not seem to be robust. Lasso in this case never recovers the true support, and this property is inherited by Relaxed Lasso which requires at least 900 samples to match our methods in terms of support recovery.

2.5.4 Statistical Performance for Varying SNR

We present two experiments studying the role of varying SNR values on the different performance measures. In each experiment, we vary the SNR between 0.01 and 100. For every SNR value, we generated 20 random datasets over which we averaged the results. We observe that for low SNR values, ridge regression (L_2) works very well in terms of prediction performance. Hence we include L_2 in our results. We do not include the results for IHT in this case as its run times are substantially longer compared to competing algorithms. The results are based on validation-set tuning, and those for oracle and random design tuning are included in Appendix 2.B.

Experiment 1: Constant Correlation

We generated datasets with constant correlation, $\rho = 0.4$, $n = 1000$, $p = 2000$, and $k^\dagger = 50$. We report the results for Algorithm 2.2 applied to the (L_0), and (L_0L_2) problems along with all the other state-of-the-art algorithms in Figure 2-4.

Figure 2-4 suggests that full support recovery is difficult for all methods (suggesting that constant correlation leads to a difficult problem). At SNR = 100, (L_0L_2) fully recovers the support while (L_0) has a recovery probability ~ 0.3—this suggests that the additional L_2-regularization aids the optimization performance of Algorithm 2.2. However, none of the other considered methods does full recovery, even for high SNR. Algorithm 2.2 (L_0L_2) generally exhibits excellent performance in terms of all measures across the whole SNR range. Pure (L_0) tends to overfit quickly (as SNR becomes smaller) due to the lack of shrinkage [115]

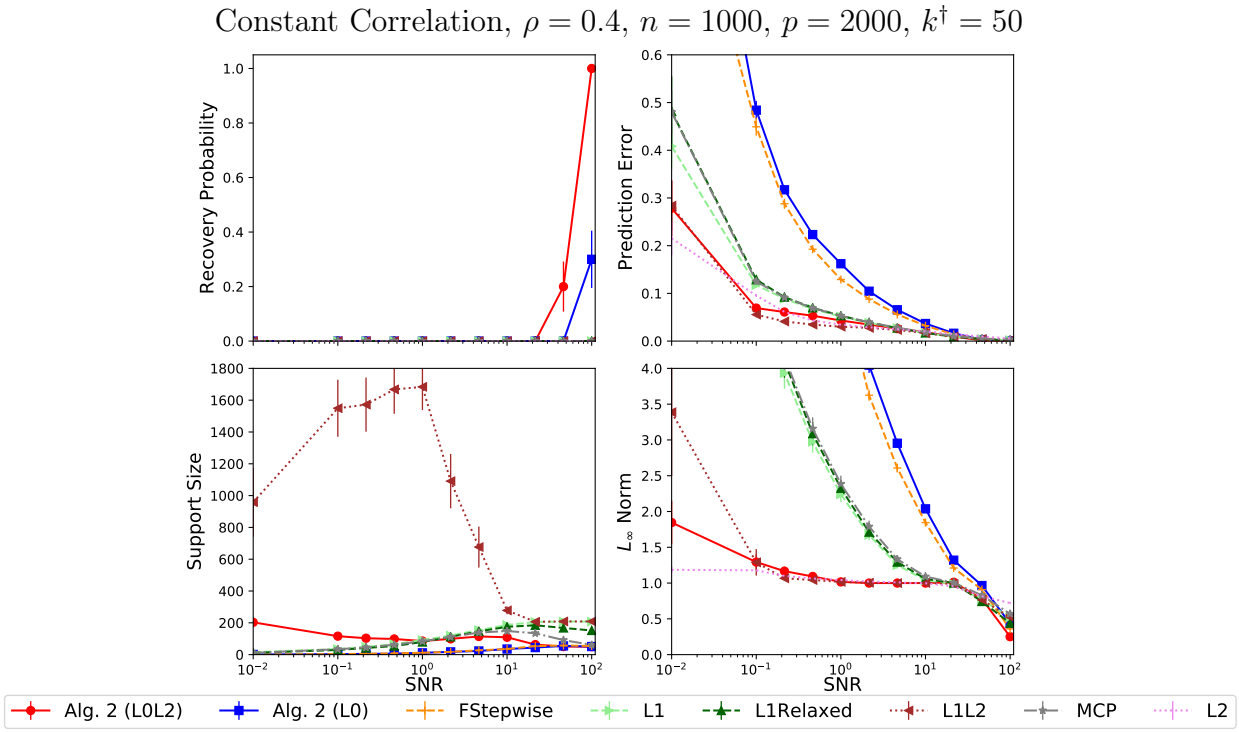


Figure 2-4: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.01 and 100. The figure compares two of our methods (Algorithm 2.2 applied to the (L_0) and (L_0L_2) problems) with other state-of-the-art algorithms. For low SNR levels, (L_0L_2) performs much better than (L_0) (as the latter overfits in these settings).

and it selects small support sizes (like "FStepwise"). Additional shrinkage (L_0L_2) seems to help alleviate this problem. The Elastic Net (L_1L_2) performs similarly to (L_0L_2) in terms of prediction error, but at the cost of very dense supports—in fact its support size can reach up to 90% of p for SNR ~ 1, which is undesirable from the viewpoint of having a parsimonious model. We note that for low SNR values, (L_0L_2)'s prediction error is comparable to that of L_2 (for SNR=0.01, L_2 has the best predictive performance)—however, (L_0L_2) leads to much sparser models and hence has an advantage.

Experiment 2: Exponential Correlation

We generated datasets having exponential correlation with $\rho = 0.5$, $n = 1000$, $p = 5000$, and $k^\dagger = 50$. We report the results in Figure 2-5. We observe that this setup is relatively easier (from a statistical viewpoint) than the constant correlation experiment in Section 2.5.4. Thus, we observe less significant differences among the algorithms, when compared to the first

experiment (see Figures 2-4 and 2-5). Algorithm 2.2 (L_0L_2) again seems to dominate across different measures and for all SNR values. Algorithm 2.2 (L_0) and Algorithm 2.2 (L_0L_2) exhibit similar performance for high SNR. For low SNR, Algorithm 2.2 (L_0L_2), L_2, and L_1L_2 have the best predictive performance, though (L_0L_2) leads to the most compact models. We note that even in this relatively easy case, Lasso and Elastic Net never fully recover the support—MCP and Relaxed Lasso also suffer in terms of full support recovery.

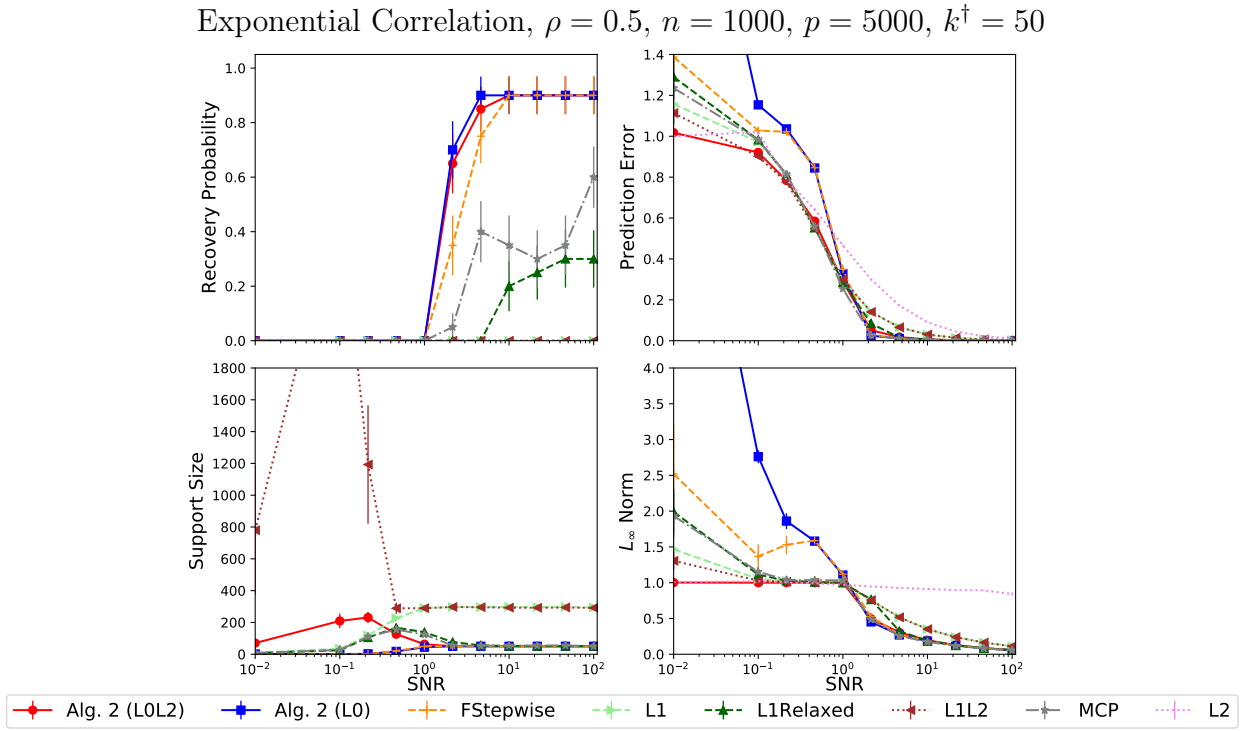


Figure 2-5: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.01 and 100. The figure compares two of our methods (Algorithm 2.2 applied to the (L_0) and (L_0L_2) problems) with other state-of-the-art algorithms. For SNR$\leq$1, (L_0L_2) performs significantly better than (L_0)—this performance improvement vanishes for larger SNR values.

2.5.5 Comparing PSI(k) versus FSI(k)

Here, we examine the differences among the various classes of minima introduced in this chapter, i.e., CW, PSI(k) and FSI(k) minima, for the (L_0) problem. To understand the differences, we consider a relatively difficult setting with Constant Correlation where $\rho = 0.9$, $n = 250$, $p = 1000$, and $k^\dagger = 25$. We set SNR$=$ 300 to allow for full support recovery. We generated 10 random training datasets under this setting and ran: Algorithm 2.1; and the

PSI and FSI variants of Algorithm 2.2 for $k \in \{1, 2, 5\}$. All algorithms were initialized with a vector of zeros. For Algorithm 2.2 we used Gurobi (v7.5) to solve the MIQO subproblems (2.15) and (2.25) when $k > 1$.

Figure 2-6 presents box plots showing the distribution of objective values, true positives, and false positives recorded for each of the algorithms and 10 datasets. PSI(1) and FSI(1) minima lead to a significant reduction in the objective, when compared to Algorithm 2.1 (which results in CW minima). We do observe further reductions as k increases, but the gains are less pronounced. In this case, CW minima contains on average a large number of false positives (> 35) and few true positives—this is perhaps due to high correlations among all features, which makes the optimization task arguably very challenging. Both PSI and FSI minima increase the number of true positives significantly. A closer inspection shows that FSI minima do a better job in having fewer false positives, when compared to PSI minima. This comes at the cost of solving relatively more difficult optimization problems, but within reasonable computation times.

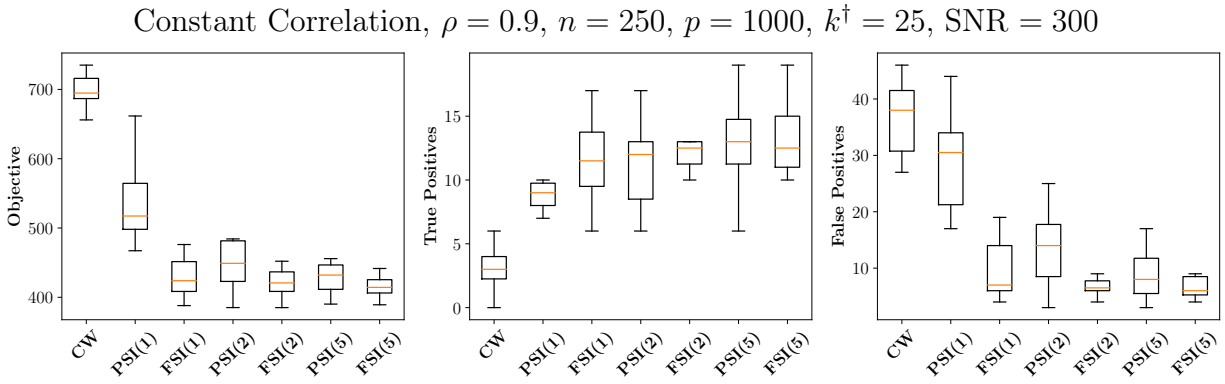


Figure 2-6: Box plots showing the distribution of objective values, number of true positives, and number of false positives for the different classes of local minima.

In Figure 2-7, we show the evolution of solutions when running the FSI(5) variant of Algorithm 2.2. The algorithm starts from a CW minimum and iterates between running CD-PSI(1) and finding a swap that improves the objective by solving optimization problem (2.25) using MIO. The PSI(1) minima obtained from running CD-PSI(1) are marked by red circles and the results obtained by solving problem (2.25) using MIO are denoted by blue squares.

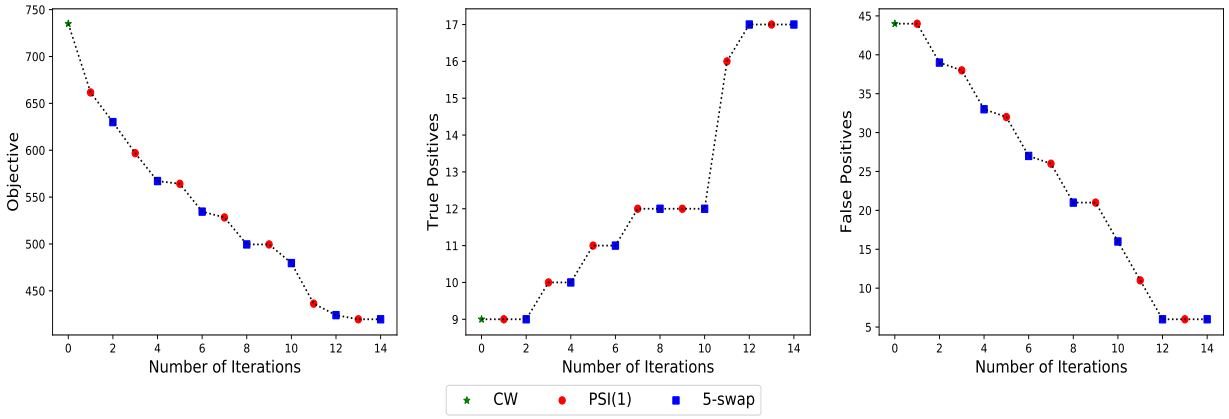


Figure 2-7: Evolution of solutions during the course of a variant of Algorithm 2.2 where we successively run CD-PSI(1) and solve combinatorial problem (2.25) to generate an FSI(5) minimum.

Figure 2-7 shows that running CD-PSI(1) on top of the solutions obtained by MIO leads to important gains in terms of better objective values, for most of the cases. This also confirms our intuition that MIO can lead to solutions that are not available via PSI(1). From the plot of true positives and false positives, we can see that CD-PSI(1) improves the solution by increasing the true positives whereas MIO improves the solution by removing false positives. This observation confirms the behavior we noticed in Figure 2-6, where PSI(1) minima were successful in obtaining a good number of true positives, but suffered in terms of false positives.

2.5.6 Large High-dimensional Experiments

Synthetic Experiments: Here, we investigate the performance of the different algorithms when $p \gg n$. We ran two experiments with a large number of features under the following settings:

- **Setting 1**: Exponential Correlation, $\rho = 0.5$, $n = 1000$, $p = 10^5/2$, $k^\dagger = 100$, and SNR $= 10$
- **Setting 2**: Constant Correlation, $\rho = 0.3$, $n = 1000$, $p = 10^5$, $k^\dagger = 50$, and SNR $= 100$

Every experiment is performed with 10 replications, and the results are averaged. We report

the results for Settings 1 and 2 in Table 2.1.

	Setting 1 ($n = 1000, p = 10^5/2, \rho = 0.5$)				**Setting 2** ($n = 1000, p = 10^5, \rho = 0.3$)			
Method	$\|\beta\|_0$	**TP**	**FP**	**PE**$\times 10^2$	$\|\beta\|_0$	**TP**	**FP**	**PE**$\times 10^3$
Alg 2 (L_0)	160 ± 24	79 ± 9	81 ± 33	5 ± 1.6	69 ± 18	47 ± 3	22 ± 22	1.6 ± 1
Alg 2.1 (L_0L_2)	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	$\mathbf{0 \pm 0}$	$\mathbf{0.97 \pm 0.05}$	$\mathbf{50 \pm 0}$	$\mathbf{50 \pm 0}$	$\mathbf{0 \pm 0}$	$\mathbf{0.5 \pm 0.02}$
Alg 2.1 (L_0L_1)	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	$\mathbf{0 \pm 0}$	1 ± 0.05	$\mathbf{50 \pm 0}$	$\mathbf{50 \pm 0}$	$\mathbf{0 \pm 0}$	$\mathbf{0.5 \pm 0.02}$
L1	808 ± 7	95 ± 1	712 ± 7	7.9 ± 0.17	478 ± 11	$\mathbf{50 \pm 0}$	428 ± 11	4.7 ± 0.1
L1Relaxed	602 ± 40	95 ± 1	508 ± 41	7.9 ± 0.19	385 ± 12	50 ± 0.2	335 ± 13	4.4 ± 0.2
MCP	102 ± 1	$\mathbf{100 \pm 0}$	2.3 ± 1	$\mathbf{0.97 \pm 0.05}$	65 ± 3	$\mathbf{50 \pm 0}$	15 ± 3	3.5 ± 0.13
FStepwise	216 ± 17	64 ± 7	152 ± 23	8.9 ± 1.3	75 ± 2	$\mathbf{50 \pm 0}$	25 ± 2	1.1 ± 0.07

Table 2.1: Performance measures for the different algorithms under Settings 1 and 2. TP, FP, and PE denote the True Positives, False Positives, and Prediction Error, respectively. The standard error of the mean is reported next to every value.

In Table 2.1, Algorithm 2.1 for the (L_0L_1) and (L_0L_2) problems fully recovers the true support and attains the lowest prediction error. None of the other methods was able to do full support recovery; Lasso and Relaxed Lasso capture most of the true positives but include a very large number of false positives. MCP comes in the middle between $(L_0L_1)/(L_0L_2)$ and Lasso—it captures all the true positives and includes few false positives. We also note that in such high SNR settings, we do not expect shrinkage (arising from the L_1/L_2 penalties) to lead to major statistical improvements. Thus, the difference in performance between (L_0) and $(L_0L_1)/(L_0L_2)$ seems to be due to the continuous regularizers that help in optimization.

Timings and Out-of-sample Performance: We ran Algorithm 2.1 using our toolkit `L0Learn` and compared the running time and predictive performance versus `glmnet` and `ncvreg`, on a variety of real and synthetic datasets. For the real datasets, there is no ground truth—we study predictive performance vis-a-vis model sparsity. We note that `L0Learn`, `glmnet`, and `ncvreg` are solving different optimization problems—the run times provided herein are meant to demonstrate that a main workhorse for our proposed framework is competitive when compared to efficient state-of-the-art implementations for sparse learning. Below we provide some details about the datasets:

- **House Prices**: $p = 104,000$ and $n = 200$. We added pairwise interactions to the popular Boston House Prices dataset [80] to get 104 features. Then, we added random "probes" (aka noisy features) by appending to the data matrix 1000 random permutations of every

column. The validation and testing sets have 100 and 206 samples, respectively.

- **Amazon Reviews**: $p = 17,580$ and $n = 2500$. We used the Amazon Grocery and Gourmet Food dataset [93] to predict the helpfulness of every review (based on its text). Specifically, we calculated the helpfulness of every review as the ratio of the number of up votes to that of down votes, and we obtained X by using Scikit-learn's TF-IDF transformer (while removing stopwords). The validation and testing sets have 500 and 1868 samples, respectively. We also created an augmented version of this dataset where we added random probes by appending to the data matrix 9 random permutations of every column to get $p = 174,755$.

- **US Census**: $p = 55,537$ and $n = 5000$. We used 37 features extracted from the 2016 US Census Planning Database to predict the mail-return rate[12] [64]. We appended the data matrix with 1500 random permutations of every column, and we randomly sampled 15,000 rows, evenly distributed between the training, testing, and validation sets.

- **Gaussian 1M**: $p = 10^6$ and $n = 200$. We generated a synthetic dataset with independent standard normal entries. We set $k^\dagger = 20$, SNR=10, and performed validation and testing as described in Section 2.5.1.

For all real datasets, we tuned and tested on separate validation and testing sets. The timings were performed on a machine with an i7-4800MQ CPU and 16GB RAM running Ubuntu 16.04 and OpenBLAS 0.2.20. For all methods, we report the training time required to obtain a grid of 100 solutions. For (L_0L_2), (L_0L_1), and MCP, we provide the time for a fixed λ_2, λ_1, and γ, respectively (these parameters have been set to the optimal values obtained via validation set tuning over 10 values of the tuning parameter). Table 2.2 presents run times for all the four methods.

The results presented in Table 2.2 show the following: `L0Learn` is faster than `glmnet` and `ncvreg` on all the considered datasets, e.g., more than twice as fast on the Amazon Reviews dataset. The speed-ups can be attributed to the careful design of `L0Learn` (as described

[12]We thank Dr. Emanuel Ben David, US Census Bureau for help on preparing this dataset.

in Section 2.4) and due to the nature of L_0 regularization which generally selects sparser supports than those obtained by L_1 or MCP regularization. Moreover, `L0Learn`, for both the (L_0L_2) and (L_0L_1) problems, provides much sparser supports and competitive testing MSE compared to the other toolkits. Finally, we note that prediction errors for our methods can be potentially improved by using Algorithm 2.2, at the cost of slightly increased computation times.

Amazon Reviews
($p = 17,580$, $n = 2500$)

Toolkit	**Time**	**MSE**$_{\times 10^2}$	$\|\beta\|_0$
glmnet (L1)	7.3	4.82	542
L0Learn (L_0L_2)	3.3	**4.77**	**159**
L0Learn (L_0L_1)	**2.8**	4.79	173
ncvreg (MCP)	10.9	6.71	1484

Amazon Reviews (+Probes)
($p = 174,755$, $n = 2500$)

Toolkit	**Time**	**MSE**$_{\times 10^2}$	$\|\beta\|_0$
glmnet (L1)	49.4	**5.11**	256
L0Learn (L_0L_2)	31.7	5.18	37
L0Learn (L_0L_1)	**29.5**	5.20	**36**
ncvreg (MCP)	67.3	5.33	318

US Census
($p = 55,537$, $n = 5000$)

Toolkit	**Time**	**MSE**	$\|\beta\|_0$
glmnet (L1)	28.7	61.3	222
L0Learn (L_0L_2)	19.6	**60.7**	15
L0Learn (L_0L_1)	**19.5**	60.8	**11**
ncvreg (MCP)	32.7	62.02	16

House Prices
($p = 104,000$, $n = 200$)

Toolkit	**Time**	**MSE**	$\|\beta\|_0$
glmnet (L1)	2.3	100	112
L0Learn (L_0L_2)	**1.8**	**94**	**59**
L0Learn (L_0L_1)	**1.8**	104	74
ncvreg (MCP)	3.9	102	140

Gaussian 1M
($p = 10^6$, $n = 200$)

Toolkit	**Time(s)**	**MSE**	$\|\beta\|_0$
glmnet (L1)	22.5	**4.55**	185
L0Learn (L_0L_2)	**16.5**	4.64	**11**
L0Learn (L_0L_1)	16.7	5.12	15
ncvreg (MCP)	36.5	4.85	147

Table 2.2: Training time (in seconds), out-of-sample MSE, and the corresponding support sizes for a variety of high-dimensional datasets. The training time is for obtaining a regularization path with 100 solutions.

2.6 Conclusion

We proposed new algorithms for Problem (2.1), based on a combination of cyclic coordinate descent and local combinatorial search, and studied their convergence properties. Our algorithms are inspired by a hierarchy of necessary optimality conditions for Problem (2.1), with solutions higher up the hierarchy being of higher quality. In terms of optimization performance, Algorithm 2.1 leads to better solutions and is faster than IHT and random CD. Our local optimization algorithms (Algorithm 2.2) often lead to further improvements over Algorithm 2.1. In many difficult settings, solutions from Algorithm 2.2 match those of global MIO solvers for Problem (2.1), while running much faster.

Our algorithms shed interesting insights onto the statistical properties of high-dimensional regression—in terms of variable selection, estimation error, prediction error vis-a-vis problem parameters ($n, p, \text{SNR}, \beta^\dagger$ and Σ). There is no overall winner among the vanilla versions of Lasso, stepwise, or L_0, across different settings—modifications such as Problem (2.1) or Relaxed Lasso [83] seem necessary. In low signal settings (e.g., low SNR or small n), where recovery (in terms of a small estimation error or full support recovery) seems impossible, one can hope to get a good predictive model that is also sparse. In these regimes, (L_0L_2), Elastic Net, and ridge typically achieve the best predictive performance, with (L_0L_2) selecting much smaller support sizes. We observe that estimators arising from Problem (2.1) typically outperform the state-of-the-art sparse learning algorithms in terms of a combination of metrics (prediction, variable selection and estimation), across a wide range of settings; and promise to be an appealing alternative to the Relaxed Lasso [83]. Our proposed algorithms allow us to uncover regimes (previously unseen due to computational limitations) where there are important differences between L_0-based estimators and existing popular algorithms (based on L_1, stepwise selection, IHT, etc.). We provide an open-source implementation of the algorithms through our toolkit `L0Learn`, which achieves up to a 3x speed-up when compared to competing toolkits.

2.A Appendix: Proofs and Technical Details

Proof of Lemma 2.1

Proof. For any $d \in \mathbb{R}^d$, we will show that $F^{'}(\beta; d)$ is given by:

$$F^{'}(\beta; d) = \begin{cases} \langle \nabla_S f(\beta), d_S \rangle & \text{if } d_{S^c} = 0 \\ \infty & \text{o.w.} \end{cases} \tag{2.27}$$

Let d be an arbitrary vector in $\mathbb{R}^p$. Then,

$$\begin{aligned} F^{'}(\beta; d) &= \liminf_{\alpha \downarrow 0} \left\{ \frac{F(\beta + \alpha d) - F(\beta)}{\alpha} \right\} \\ &= \liminf_{\alpha \downarrow 0} \Big\{ \underbrace{\frac{f(\beta + \alpha d) - f(\beta)}{\alpha}}_{\text{Term I}} + \underbrace{\lambda_0 \sum_{i \in S} \frac{\|\beta_i + \alpha d_i\|_0 - 1}{\alpha}}_{\text{Term II}} + \underbrace{\lambda_0 \sum_{j \notin S} \frac{\|\alpha d_j\|_0}{\alpha}}_{\text{Term III}} \Big\} \end{aligned}$$

First we note that $\lim_{\alpha \downarrow 0} \text{Term II} = 0$ since for any $i \in S$, $\|\beta_i + \alpha d_i\|_0 = 1$ for sufficiently small α. Suppose $d_{S^c} = 0$. Then, the continuity of f implies that $\lim_{\alpha \downarrow 0} \text{Term I} = f^{'}(\beta_S; d_S) = \langle \nabla_S f(\beta), d_S \rangle$, where the second equality follows by observing that $\beta_S \to f(\beta_S)$ is continuously differentiable (in the neighborhood of β_S). Also, Term III $= 0$. Therefore, we have:

$$F^{'}(\beta; d) = \lim_{\alpha \downarrow 0} \text{Term I} + \lim_{\alpha \downarrow 0} \text{Term II} = \langle \nabla_S f(\beta), d_S \rangle.$$

We now consider the case when $d_{S^c} \neq 0$. In this case, $\lim_{\alpha \downarrow 0} \text{Term III} = \infty$; and since the limit of Term I is bounded, we have $F^{'}(\beta; d) = \infty$. Thus, we have shown that (2.27) holds. From (2.27), we have $F^{'}(\beta; d) \geq 0$ for all d iff $\nabla_S f(\beta) = 0$. □

Proof of Lemma 2.2

Proof. Let $g(u)$ denote the objective function minimized in (2.7), i.e.,

$$g(u) := \frac{1 + 2\lambda_2}{2} \Big(u - \frac{\widetilde{\beta}_i^*}{1 + 2\lambda_2} \Big)^2 + \lambda_1 |u| + \lambda_0 \mathbb{1}[u \neq 0].$$

If $|\widetilde{\beta}_i^*| > \lambda_1$, then $\min_{u \neq 0} g(u)$ is attained by $\widehat{u} = \frac{\text{sign}(\widetilde{\beta}_i^*)}{1+2\lambda_2}(|\widetilde{\beta}_i^*| - \lambda_1)$ (this is the well-known soft-thresholding operator). Now, $g(\widehat{u}) < g(0)$ is equivalent to $\frac{|\widetilde{\beta}_i^*|-\lambda_1}{1+2\lambda_2} > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. Hence, $\widehat{u}$ is the minimizer of $g(u)$ when $\frac{|\widetilde{\beta}_i^*|-\lambda_1}{1+2\lambda_2} > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. Both $\widehat{u}$ and 0 are minimizers of $g(u)$ if $\frac{|\widetilde{\beta}_i^*|-\lambda_1}{1+2\lambda_2} = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. Finally, when $\frac{|\widetilde{\beta}_i^*|-\lambda_1}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$, the function $g(u)$ is minimized at $u = 0$. □

Proof of Theorem 2.1

The proof of the theorem is similar to the proofs in [36, 115, 24] (which consider the cardinality constrained version of Problem (2.2)).

Proof of Lemma 2.5

Proof. If β^k is the result of a non-spacer step then $F(\beta^k) \leq F(\beta^{k-1})$ holds by definition. If β^k is obtained after a spacer step, then $f(\beta^k) \leq f(\beta^{k-1})$. Since a spacer step cannot increase the support size of β^{k-1}, this implies that $\|\beta^k\|_0 \leq \|\beta^{k-1}\|_0$, and thus $F(\beta^k) \leq F(\beta^{k-1})$. Since $F(\beta^k)$ is non-increasing and bounded below (by zero), it must converge to some $F^* \geq 0$. □

Proof of Lemma 2.6

Proof. The result holds trivially if $p \leq n$. Suppose $p > n$. In the (L_0) problem, Assumption 2 states that $F(\beta^0) \leq \lambda_0 n$. Since Algorithm 2.1 is a descent method (by Lemma 2.5) we have $F(\beta^k) \leq \lambda_0 n$ for every k, which implies $f(\beta^k) + \lambda_0 \|\beta^k\|_0 \leq \lambda_0 n$ and hence, $\lambda_0 \|\beta^k\|_0 \leq \lambda_0 n$. Therefore, $\|\beta^k\|_0 \leq n$ for all k. Similarly, for the $(L_0 L_1)$ problem, Assumption 2 and the descent property imply $F(\beta^k) \leq f(\beta^{\ell_1}) + \lambda_0 n$ which can be equivalently written as $f(\beta^k) - f(\beta^{\ell_1}) \leq \lambda_0(n - \|\beta^k\|_0)$. But the optimality of the lasso solution implies $f(\beta^k) - f(\beta^{\ell_1}) \geq 0$, which leads to $\|\beta^k\|_0 \leq n$. □

Proof of Theorem 2.2

Before presenting the proof of Theorem 2.2, we present some necessary lemmas. First, we recall how the iterates are indexed by Algorithm 2.1. If β^l is obtained after performing a

spacer step, then β^{l-1} corresponds to a non-spacer step—by this time, a certain support has occurred for Cp times. Suppose, β^k denotes the current value of β in Algorithm 2.1. If the next step is a non-spacer step, then β^{k+1} is obtained from β^k by updating a single coordinate. Otherwise, if the next step is a spacer step, then *all* the coordinates inside the support of β^k will be updated to get β^{k+1}.

The following lemma shows that the sequence generated by Algorithm 2.1 is bounded.

Lemma 2.8. *The sequence $\{\beta^k\}$ is bounded.*

Proof. For the (L_0L_1) and (L_0L_2) problems, for all k, β^k belongs to the level set $G = \{\beta \in \mathbb{R}^p \mid F(\beta) \leq F(\beta^0)\}$ where β^0 is an initial solution. Since in both cases $F(\beta)$ is coercive, G is bounded and therefore, $\{\beta^k\}$ is bounded.

We now study the (L_0) problem. Firstly, if $p \leq n$, then the objective function for the (L_0) problem is coercive (under Assumption 2.1), and the previous argument used for $(L_0L_1)/(L_0L_2)$ applies. Otherwise, suppose that $p > n$. Recall that from Lemma 2.6, we have $\|\beta^k\|_0 \leq n$ for all $k \geq 0$; and from Assumption 2, we have $F(\beta^0) \leq \lambda_0 n$. In addition, by Lemma 2.5, we have $F(\beta^k) \leq \lambda_0 n$ for every k. Therefore, it follows that $\beta^k \in A$ where,

$$A = \bigcup_{S \subseteq [p], |S| \leq n} A_S, \quad \text{and} \quad A_S = \{\beta \in \mathbb{R}^p \mid \frac{1}{2}\|y - X_S\beta_S\|^2 \leq \lambda n, \ \beta_{S^c} = 0\}.$$

Note that in every A_S, the only components of β that might be non-zero are in β_S. By Assumption 2.1, the level set $\{\beta_S | \frac{1}{2}\|y - X_S\beta_S\|^2 \leq \lambda n\} \subseteq \mathbb{R}^{|S|}$ is bounded, which implies that A_S is bounded. Since A is the union of a finite number of bounded sets, it is also bounded. □

The next lemma characterizes the limit points of Algorithm 2.1.

Lemma 2.9. *Let S be a support that is generated infinitely often by the non-spacer steps, and let $\{\beta^l\}_{l \in L}$ be the sequence of spacer steps generated by S. Then, the following hold true:*

1. *There exists an integer N such that for all $l \in L$ and $l \geq N$ we have $Supp(\beta^l) = S$.*

2. *There exists a subsequence of* $\{\beta^l\}_{l\in L}$ *that converges to a stationary solution* β^**, where,* β^*_S *is the unique minimizer of* $\min_{\beta_S} f(\beta_S)$ *and* $\beta^*_{S^c} = 0$.

3. *Every subsequence of* $\{\beta^k\}_{k\geq 0}$ *with support* S *converges to* β^* *(as in Part 2, above).*

4. β^* *satisfies* $|\beta^*_j| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$ *for every* $j \in S$.

Proof. **Part 1.)** Since the spacer steps optimize only over the coordinates in S, no element from outside S can be added to the support by the spacer step. Thus, for every $l \in L$ we have $\text{Supp}(\beta^l) \subseteq S$. We now show that strict containment is not possible using the method of contradiction. To this end, suppose $\text{Supp}(\beta^l) \subsetneq S$ occurs infinitely often; and let us consider some $l \in L$ at which this occurs. By Algorithm 2.1, the previous iterate β^{l-1} has a support S, which implies $\|\beta^{l-1}\|_0 - \|\beta^l\|_0 \geq 1$. Moreover, from the definition of the spacer step we have $f(\beta^l) \leq f(\beta^{l-1})$. Therefore, we get

$$F(\beta^{l-1}) - F(\beta^l) = \underbrace{f(\beta^{l-1}) - f(\beta^l)}_{\geq 0} + \lambda_0(\underbrace{\|\beta^{l-1}\|_0 - \|\beta^l\|_0}_{\geq 1}) \geq \lambda_0.$$

Thus, every time the event $\text{Supp}(\beta^l) \subsetneq S$ occurs, the objective F decreases by at least λ_0. This contradicts the fact that F is lower bounded by 0, which establishes the result.

Part 2.) The proof follows the standard steps for proving the convergence of cyclic CD (e.g., as in [22]) — we provide the proof for completeness. First, we introduce some additional notation for the proof. Fix some $l \in L$. By Algorithm 2.1, β^{l-1} has a support S which we assume (without loss of generality) to be $S = \{1, 2, \ldots, J\}$. We recall that to obtain β^l from β^{l-1}, Algorithm 2.1 performs a spacer step—i.e, it starts from β^{l-1} and updates every coordinate in S via $T(\cdot, 0, \lambda_1, \lambda_2)$. We denote the intermediate iterates generated sequentially by the spacer step as: $\beta^{l,1}, \beta^{l,2}, \ldots, \beta^{l,J}$ where $\beta^{l,J} = \beta^l$.

Since the support S occurs infinitely often, we consider the infinite sequence $\{\beta^{l,1}\}_{l\in L}$ (i.e., the sequence of intermediate spacer steps where coordinate 1 is updated). By Lemma 2.8, $\{\beta^{l,1}\}_{l\in L}$ is bounded and therefore, there exists a further subsequence $\{\beta^{l',1}\}_{l'\in L'}$ that converges to a limit point β^* (say). We assume that for every $l' \in L'$ we have $l' \geq N$, which

implies that $\beta^{l'-1}$, $\beta^{l',1}, \beta^{l',2}, \ldots, \beta^{l',J}$ all have the support S (this follows from Part 1 of this lemma). Next, we will show that $\beta^{l,2}$ also converges to β^*.

Fix some $l' \in L'$. Then, we have:

$$F(\beta^{l',1}) - F(\beta^{l',2}) = f(\beta_S^{l',1}) - f(\beta_S^{l',2}) \geq \frac{1+2\lambda_2}{2}(\beta_2^{l',1} - \beta_2^{l',2})^2 \tag{2.28}$$

where the last inequality follows by replacing $\beta_2^{l',2}$ with the expression given by the thresholding map in (2.12) and simplifying. By Lemma 2.5, $\{F(\beta^k)\}$ converges; so taking the limit as $l' \to \infty$ in (2.28) we get $\beta_2^{l',1} - \beta_2^{l',2} \to 0$ as $l' \to \infty$. Since $\beta^{l',1} \to \beta^*$, we conclude that $\beta_2^{l',2} \to \beta_2^*$. Therefore, $\beta^{l',2} \to \beta^*$. The same argument applies to $\beta^{l',i}$ and $\beta^{l',i+1}$ for every $i \in \{2, 3, \ldots, J-1\}$. Therefore, we conclude that for every $i \in \{1, 2, \ldots, J\}$ we have $\beta^{l',i} \to \beta^*$.

Let $k', l' \in L'$ be such that $k' > l'$, then $f(\beta^{k'}) \leq f(\beta^{l',1}) \leq f(\beta_1, \beta_2^{l'}, \beta_3^{l'}, \ldots)$ for any $\beta_1 \in \mathbb{R}$. As $k', l' \to \infty$, we get $f(\beta^*) \leq f(\beta_1, \beta_2^*, \beta_3^*, \ldots)$ for any β_1. The same result applies to all other coordinates in j, which implies that $0 \in \partial f(\beta_1^*, \beta_2^*, \beta_2^*, \ldots)$ (where, $\partial f(\cdot)$ denotes subgradient) and consequently β^* is a stationary solution for $\min_{\beta_S} f(\beta_S)$. Finally, Lemma 2.6 and Assumption 2.1 for the (L_0) and (L_0L_1) problems imply that $\beta_S \mapsto f(\beta_S)$ is strongly convex and has a unique minimizer — hence β^* is unique.

Part 3.) By Part 2, there exists a subsequence $\{\beta^{l'}\}_{l' \in L'}$ converging to β^*. By Part 1, for every $l' \geq N$ we have $F(\beta^{l'}) = f(\beta^{l'}) + \lambda_0|S|$. Taking $l' \to \infty$ and using the continuity of $f(\beta)$ we get:

$$\lim_{l' \to \infty} F(\beta^{l'}) = f(\beta^*) + \lambda_0|S|.$$

Now, consider any subsequence $\{\beta^{k'}\}_{k' \in K'}$, where $K' \subseteq \{0, 1, 2, \ldots\}$, such that the non-spacer steps in K' have a support S. We will establish convergence of $\{\beta^{k'}\}_{k' \in K'}$ via the method of contradiction. To this end, suppose $\{\beta^{k'}\}_{k' \in K'}$ has a limit point $\widehat{\beta}$ which is not equal to β^*. Then, there exists a subsequence $\{\beta^{k''}\}_{k'' \in K''}$ (with $K'' \subseteq K'$) which converges to $\widehat{\beta}$. Then, for every $k'' \geq N$, we have $F(\beta^{k''}) = f(\beta^{k''}) + \lambda_0|S|$. Taking the limit as $k'' \to \infty$

we get:

$$\lim_{k''\to\infty} F(\beta^{k''}) = f(\widehat{\beta}) + \lambda_0|S|.$$

From Lemma 2.5, we have $\lim_{l'\to\infty} F(\beta^{l'}) = \lim_{k''\to\infty} F(\beta^{k''})$. This implies $f(\beta^*) = f(\widehat{\beta})$ and in particular, $f(\beta^*_S) = f(\widehat{\beta}_S)$ (since the supports of both limits points are a subset of S). However, by Part 2, we know that β^*_S is the unique minimizer of $\min_{\beta_S} f(\beta_S)$—which leads to a contradiction. Hence, we conclude that $\beta^{k'} \to \beta^*$ as $k' \to \infty$.

Part 4.) Let l_1 and l_2 be the indices of any two consecutive spacer steps generated by the support S. Recall from Algorithm 2.1 that $C \geq 1$; and the support S must appear in Cp non-spacer steps between l_1 and l_2. Fix some $i \in S$. We will show that there exists a non-spacer step with index k' such that $l_1 < k' < l_2$, $\text{Supp}(\beta^{k'}) = S$, and $|\beta_i^{k'}| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. We proceed by contradiction. To this end, suppose that such an index does not exist — i.e., every non-spacer step that updates coordinate i thresholds it to 0. Let k_1 denote the iteration index of the first non-spacer step between l_1 and l_2 that updates coordinate i. In the p coordinate updates after l_1, coordinate i must be updated once, which implies $k_1 - l_1 \leq p$. Since at iteration k_1, coordinate i is set to 0, the support S can appear at most $p-1$ times between l_1 and k_1. Moreover, between k_1 and l_2, S appears 0 times — this is because, coordinate i never gets thresholded to a non-zero value by a non-spacer step. Therefore, S appears for at most $p-1$ times between l_1 and l_2 – this contradicts the fact that l_2 is the index after which S appears in Cp non-spacer steps. Therefore, we conclude that there exists an index k' such that $l_1 < k' < l_2$, $\text{Supp}(\beta^{k'}) = S$, and $|\beta_i^{k'}| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$.

Let us now fix some $i \in S$. By considering the infinite sequence of spacer steps generated by S and applying the result we proved above to every two consecutive spacer steps, we can see that there exists an infinite subsequence $\{\beta^{k'}\}$ of non-spacer iterates where, for every k', we have $\text{Supp}(\beta^{k'}) = S$ and $|\beta_i^{k'}| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. By Part 3 of this lemma, $\{\beta^{k'}\}$ converges to the stationary solution β^*. Taking the limit $k \to \infty$ in inequality: $|\beta_i^k| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$, we conclude that $|\beta_i^*| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. □

Lemma 2.10 shows that the support corresponding to any limit point of $\{\beta^k\}$ appears infinitely often.

Lemma 2.10. *Let B be a limit point of $\{\beta^k\}$ with Supp$(B) = S$, then Supp$(\beta^k) = S$ for infinitely many k.*

Proof. We prove this result by using a contradiction argument. To this end, suppose support S occurs only finitely many times. Since there are only finitely many supports, there is a support S' with $S' \neq S$; and a subsequence $\{\beta^{k'}\}$ of $\{\beta^k\}$ which satisfies: Supp$(\beta^{k'}) = S'$ for all k'; and $\beta^{k'} \to B$ as $k' \to \infty$. However, this is not possible by Part 3 of Lemma 2.9. □

Lemma 2.11 is technical and will be needed in the proof of convergence of Algorithm 2.1.

Lemma 2.11. *Let $B^{(1)}$ and $B^{(2)}$ be two limit points of the sequence $\{\beta^k\}$, with supports S_1 and S_2, respectively. Suppose that $S_2 = S_1 \cup \{j\}$ for some $j \notin S_1$. Then, exactly one of the following holds:*

1. *If there exists an $i \in S_1$ such that $\langle X_i, X_j \rangle \neq 0$, then $|B_j^{(2)}| > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$.*
2. *Otherwise, if $\langle X_i, X_j \rangle = 0$ for all $i \in S_1$, then $|B_j^{(2)}| = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. Furthermore, for any $\beta \in \mathbb{R}^p$ with Supp$(\beta) = S_2$, we have $|T(\widetilde{\beta}_j, \lambda_0, \lambda_1, \lambda_2)| = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$.*

Proof. We first derive an useful expression for $B_j^{(2)}$, which will help us establish parts 1 and 2 of this lemma. By Lemma 2.5, $F(\beta^k)$ converges to a finite non-negative limit F^*. Lemmas 2.9 and 2.10 imply that there is a subsequence $\{\beta^{k'}\}_{k' \in K'}$ that converges to $B^{(1)}$ and satisfies Supp$(\beta^{k'}) = S_1$ for every k'. As $k \to \infty$, we get: $F^* = f(B^{(1)}) + |S_1| = F(B^{(1)})$. Similarly, for $B^{(2)}$ we have $F^* = F(B^{(2)})$. Therefore, $F(B^{(1)}) = F(B^{(2)})$, which is equivalent to

$$f(B_{S_1}^{(1)}) + \lambda_0 \|B_{S_1}^{(1)}\|_0 = f(B_{S_2}^{(2)}) + \lambda_0 \|B_{S_2}^{(2)}\|_0.$$

Since $\|B^{(2)}_{S_2}\|_0 = \|B^{(1)}_{S_1}\|_0 + 1$, we can simplify the above to obtain:

$$f(B^{(1)}_{S_1}) - f(B^{(2)}_{S_2}) = \lambda_0. \tag{2.29}$$

The term $f(B^{(2)}_{S_2})$ can be rewritten as follows (using elementary algebraic manipulations)

$$\begin{aligned}
f(B^{(2)}_{S_2}) &= \frac{1}{2}\|y - X_{S_2}B^{(2)}_{S_2}\|^2 + \lambda_1\|B^{(2)}_{S_2}\|_1 + \lambda_2\|B^{(2)}_{S_2}\|_2^2 \\
&= \frac{1}{2}\|y - X_{S_1}B^{(2)}_{S_1} - X_jB^{(2)}_j\|^2 + \lambda_1\|B^{(2)}_{S_1}\|_1 + \lambda_1|B^{(2)}_j| + \lambda_2\|B^{(2)}_{S_1}\|_2^2 + \lambda_2(B^{(2)}_j)^2 \\
&= \Big(\frac{1}{2}\|y - X_{S_1}B^{(2)}_{S_1}\|^2 + \lambda_1\|B^{(2)}_{S_1}\|_1 + \lambda_2\|B^{(2)}_{S_1}\|_2^2\Big) \\
&- \langle y - X_{S_1}B^{(2)}_{S_1}, X_j\rangle B^{(2)}_j + \frac{1}{2}\|X_j\|^2{B^{(2)}_j}^2 + \lambda_1|B^{(2)}_j| + \lambda_2(B^{(2)}_j)^2 \\
&= f(B^{(2)}_{S_1}) - \langle y - X_{S_1}B^{(2)}_{S_1}, X_j\rangle B^{(2)}_j + \frac{1}{2}\|X_j\|^2(B^{(2)}_j)^2 + \lambda_1|B^{(2)}_j| + \lambda_2(B^{(2)}_j)^2. \quad (2.30)
\end{aligned}$$

From Lemma 2.9 we know that $B^{(2)}$ is a stationary solution. Using the characterization of stationary solutions in (2.5) and rearranging the terms, we get:

$$\begin{aligned}
\langle y - X_{S_1}B^{(2)}_{S_1}, X_j\rangle &= (1 + 2\lambda_2)B^{(2)}_j + \lambda_1\text{sign}(\langle y - X_{S_1}B^{(2)}_{S_1}, X_j\rangle) \\
|\langle y - X_{S_1}B^{(2)}_{S_1}, X_j\rangle| &> \lambda_1.
\end{aligned} \tag{2.31}$$

Multiplying the first equation in the above by $B^{(2)}_j$ and using that the fact $\langle y - X_{S_1}B^{(2)}_{S_1}, X_j\rangle$ and $B^{(2)}_j$ have the same sign (which is evident from the system above), we arrive at

$$\langle y - X_{S_1}B^{(2)}_{S_1}, X_j\rangle B^{(2)}_j = (1 + 2\lambda_2)(B^{(2)}_j)^2 + \lambda_1|B^{(2)}_j|. \tag{2.32}$$

Plugging in the above expression in the second term on the r.h.s of (2.30) and using the fact that $\|X_j\|^2 = 1$ we get

$$f(B^{(2)}_{S_2}) = f(B^{(2)}_{S_1}) - \frac{1 + 2\lambda_2}{2}(B^{(2)}_j)^2. \tag{2.33}$$

Substituting (2.33) into equation (2.29) and rearranging terms, we arrive at

$$|B^{(2)}_j| = \sqrt{\frac{2\lambda_0}{1 + 2\lambda_2} + \frac{2}{1 + 2\lambda_2}\Big(f(B^{(2)}_{S_1}) - f(B^{(1)}_{S_1})\Big)}. \tag{2.34}$$

Part 1.) We consider Part 1, where there exists an $i \in S_1$ such that $\langle X_i, X_j \rangle \neq 0$. By Lemma 2.9 we have that $B^{(1)}$ is a stationary solution. Thus, $B^{(1)}_{S_1} \in \arg\min_{\beta_{S_1}} f(\beta_{S_1})$, and the following holds

$$f(B^{(1)}_{S_1}) \leq f(B^{(2)}_{S_1}). \tag{2.35}$$

We will show the inequality above is strict. To this end, suppose that (2.35) holds with equality. Lemma 2.10 implies that S_1 appears in the sequence of iterates. But the function $f(\beta_{S_1})$ is strongly convex (this is trivial for (L_0L_2) and holds due to Assumption 2.1 and Lemma 2.6 for the (L_0) and (L_0L_1) problems). Thus, $B^{(1)}_{S_1}$ is the unique minimizer of $f(\beta_{S_1})$. Therefore, it must be the case that $B^{(1)}_{S_1} = B^{(2)}_{S_1}$, and in particular $B^{(1)}_i = B^{(2)}_i$. By the characterization of stationary solutions in (2.5) we have:

$$\begin{aligned} &\operatorname{sign}(\langle y - X_{S_1\setminus\{i\}} B^{(1)}_{S_1\setminus\{i\}}, X_i\rangle) \overbrace{\frac{|\langle y - X_{S_1\setminus\{i\}} B^{(1)}_{S_1\setminus\{i\}}, X_i\rangle| - \lambda_1}{1+2\lambda_2}}^{\geq 0} \\ = &\operatorname{sign}(\langle y - X_{S_2\setminus\{i\}} B^{(2)}_{S_2\setminus\{i\}}, X_i\rangle) \underbrace{\frac{|\langle y - X_{S_2\setminus\{i\}} B^{(2)}_{S_2\setminus\{i\}}, X_i\rangle| - \lambda_1}{1+2\lambda_2}}_{\geq 0} \end{aligned} \tag{2.36}$$

Observing that the two sign terms in (2.36) are equal, we can simplify the above to:

$$\begin{aligned} \langle y - X_{S_1\setminus\{i\}} B^{(1)}_{S_1\setminus\{i\}}, X_i\rangle &= \langle y - X_{S_2\setminus\{i\}} B^{(2)}_{S_2\setminus\{i\}}, X_i\rangle \\ &= \langle y - X_j B^{(2)}_j - X_{S_1\setminus\{i\}} B^{(2)}_{S_1\setminus\{i\}}, X_i\rangle \end{aligned} \tag{2.37}$$

where, the second line in (2.37) follows by noting $X_{S_2\setminus\{i\}} B^{(2)}_{S_2\setminus\{i\}} = X_j B^{(2)}_j + X_{S_1\setminus\{i\}} B^{(2)}_{S_1\setminus\{i\}}$. Substituting $B^{(2)}_{S_1} = B^{(1)}_{S_1}$ in (2.37) and simplifying, we get $\langle X_j, X_i\rangle = 0$, which contradicts the assumption in Part 1. Thus, we have established that inequality (2.35) is strict. Using this result in (2.34), we conclude that: $|B^{(2)}_j| > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$.

Part 2.) We now consider the case where $\langle X_i, X_j\rangle = 0$ for all $i \in S_1$. In this case, the optimization problem $\min_{\beta_{S_2}} f(\beta_{S_2})$ separates into optimization w.r.t the variables β_{S_1} and β_j. Note that $B^{(2)}_{S_1}$ and $B^{(1)}_{S_1}$ are both minimizers of $\min_{\beta_{S_1}} f(\beta_{S_1})$; and hence $f(B^{(2)}_{S_1}) = f(B^{(1)}_{S_1})$. Thus, from (2.34) we get $|B^{(2)}_j| = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. Finally, we note that for any $\beta \in \mathbb{R}^p$

such that $\text{Supp}(\beta) = S_2$, we have that $T(\widetilde{\beta}_j, \lambda_0, \lambda_1, \lambda_2) = B_j^{(2)}$. This completes the proof. $\square$

The following establishes a lower bound on the decrease in objective value, when a non-zero coordinate is set to zero during Algorithm 2.1.

Lemma 2.12. *Let β^k be an iterate of Algorithm 2.1 with $\beta_j^k \neq 0$ for some $j \in [p]$. Let β^{k+1} correspond to a non-spacer step which updates coordinate j to 0, i.e., $\beta_j^{k+1} = 0$. Then, the following holds:*

$$F(\beta^k) - F(\beta^{k+1}) \geq \frac{1+2\lambda_2}{2}\left(|\beta_j^k| - \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}\right)^2. \tag{2.38}$$

Proof. $F(\beta^k) - F(\beta^{k+1})$ can be simplified by noting that $\beta_i^k = \beta_i^{k+1}$ for all $i \neq j$ and $\beta_j^{k+1} = 0$:

$$\begin{aligned} F(\beta^k) - F(\beta^{k+1}) &= -\widetilde{\beta}_j^k \beta_j^k + \frac{1+2\lambda_2}{2}(\beta_j^k)^2 + \lambda_0 + \lambda_1|\beta_j^k| \\ &\geq -|\widetilde{\beta}_j^k||\beta_j^k| + \frac{1+2\lambda_2}{2}(\beta_j^k)^2 + \lambda_0 + \lambda_1|\beta_j^k| \\ &\geq -|\beta_j^k|(|\widetilde{\beta}_j^k| - \lambda_1) + \frac{1+2\lambda_2}{2}(\beta_j^k)^2 + \lambda_0, \end{aligned} \tag{2.39}$$

where $\widetilde{\beta}_j^k = \langle y - \sum_{i \neq j} X_j \beta_i^k, X_j \rangle$. Since β^{k+1} is a non-spacer step which sets coordinate j to 0, the definition of the thresholding operator (2.12) implies $|\widetilde{\beta}_j^k| - \lambda_1 < \sqrt{2\lambda_0(1+2\lambda_2)}$. Plugging this bound into (2.39) and factorizing, we arrive to the result of the lemma. $\square$

Proof of Theorem 2.2 Finally, we present the proof of Theorem 2.2 below.

Proof. : Let B be a limit point of $\{\beta^k\}$ with the largest support size and denote its support by S. We will show that $\beta^k \to B$ as $k \to \infty$.

Part 1.) By Lemma 2.10, there is a subsequence $\{\beta^r\}_{r \in R}$ of $\{\beta^k\}$ which satisfies: $\text{Supp}(\beta^r) = S$ for all r and $\beta^r \to B$ (as $r \to \infty$). By Lemma 2.9, there exists an integer N such that for every $r \geq N$, if $r+1$ is a spacer step then $\text{Supp}(\beta^{r+1}) = \text{Supp}(\beta^r)$. In what follows, we assume that $r \geq N$. Let j be any element in S. We will show that there exists an integer N_j such that for every $r \geq N_j$, we have $j \in \text{Supp}(\beta^{r+1})$. We show this by contradiction. To this end, let $j \notin \text{Supp}(\beta^{r+1})$ for infinitely many values of r. Hence, there is a further subsequence

$\{\beta^{r'}\}_{r' \in R'}$ of $\{\beta^r\}_{r \in R}$ (with $R' \subseteq R$) such that $\text{Supp}(\beta^{r'+1}) = S \setminus \{j\}$. For every $r' \in R'$, note that $r'+1$ is a non-spacer step (since $r' \geq N$). Therefore, applying Lemma 2.12 with $k = r'$, we get:

$$F(\beta^{r'}) - F(\beta^{r'+1}) \geq \frac{1+2\lambda_2}{2}\left(|\beta_j^{r'}| - \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}\right)^2. \tag{2.40}$$

Taking $r' \to \infty$ in (2.40) and using the convergence of $\{F(\beta^k)\}$ (by Lemma 2.5), we conclude

$$|B_j| = \lim_{r' \to \infty} |\beta_j^{r'}| = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}. \tag{2.41}$$

Since $\text{Supp}(\beta^{r'+1}) = S \setminus \{j\}$ for every r', Lemma 2.9 implies that $\{\beta^{r'+1}\}$ converges to a limit point, which we denote by $\widehat{B}$. If $\langle X_i, X_j \rangle = 0$ for all $i \in S \setminus \{j\}$, then Lemma 2.11 (part 2) implies $j \in \text{Supp}(\beta^{r'+1})$, which contradicts the definition of $\{\beta^{r'}\}_{r' \in R'}$. Thus, it must be the case that there exists an index $i \in S \setminus \{j\}$ such that $\langle X_i, X_j \rangle \neq 0$. Applying Lemma 2.11 (part 1) to B and $\widehat{B}$ we have that $|B_j| > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$ — this contradicts (2.41). Therefore, there exists an integer N_j such that for every $r \geq N_j$, we have $j \in \text{Supp}(\beta^{r+1})$.

The above argument says that no j in the support of B can be dropped infinitely often in the sequence $\{\beta_k\}$. Since S has the largest support size, no coordinate can be added to S infinitely often in the sequence $\{\beta_k\}$. This concludes the proof of Part 1.

Part 2.) Finally, we show that the limit of $\{\beta^k\}$ is a CW minimum. To this end, note that the results of Part 1 (above) and Lemma 2.9 (Parts 3 and 4) imply that β^k converges to the limit B, which satisfies $\text{Supp}(B) = S$, and for every $i \in S$, we have:

$$B_i = \text{sign}(\widetilde{B}_i)\frac{|\widetilde{B}_i| - \lambda_1}{1+2\lambda_2} \quad \text{and} \quad |B_i| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}. \tag{2.42}$$

Fix some $j \notin \text{Supp}(B)$ and let $\{\beta^{k'}\}_{k' \in K'}$ be the sequence of non-spacer iterates at which coordinate j is updated. For every k' after support stabilization, the algorithm maintains:

$$\frac{|\widetilde{\beta}_j^{k'}| - \lambda_1}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$$

where $\widetilde{\beta}_j^{k'} = \langle y - \sum_{i \neq j} X_i \beta_i^{k'}, X_j \rangle$. Taking $k' \to \infty$ in the above, we have:

$$\frac{|\widetilde{B}_j| - \lambda_1}{1 + 2\lambda_2} \leq \sqrt{\frac{2\lambda_0}{1 + 2\lambda_2}}. \tag{2.43}$$

(2.42) and (2.43) together imply that B is a CW minimum (by definition). □

Proof of Theorem 2.3

Proof. By Theorem 2.2, we have $\beta^K \to B$, and there exists an integer M such that for all $K \geq M$, we have $\text{Supp}(\beta^K) = S$ and $\text{Supp}(B) = S$. Therefore, there exists an integer $N \geq M$ such that for $K \geq N$, $\text{sign}(\beta_i^K) = \text{sign}(B_i)$ for every $i \in S$. For $K \geq N$, it can be readily seen that the iterates β^K are the same as those generated by minimizing the following objective

$$g(\beta_S) = \frac{1}{2}\|y - X_S\beta_S\|^2 + \lambda_1 \sum_{i \in S, B_i > 0} \beta_i - \lambda_1 \sum_{i \in S, B_i < 0} \beta_i + \lambda_2 \|\beta_S\|^2, \tag{2.44}$$

using coordinate descent with step size $\frac{1}{1+2\lambda_2}$ and starting from the initial solution β^N. The function $\beta_S \mapsto g(\beta_S)$ is continuously differentiable and its gradient is Lipschitz continuous with parameter $L = M_S + 2\lambda_2$. Moreover, it is strongly convex with strong-convexity parameter $\sigma_S = m_S + 2\lambda_2$. [18] (see Theorem 3.9) has proven a linear rate of convergence for cyclic CD when applied to strongly convex and continuously differentiable functions. Applying [18]'s result in our context leads to the conclusion of the theorem. □

Proof of Theorem 2.4

Proof. Before it terminates, Algorithm 2.2 leads to a sequence $\{\beta^i\}_0^\ell$ such that $F(\beta^\ell) < F(\beta^{\ell-1}) < \cdots < F(\beta^1)$. Since $\beta^\ell, \beta^{\ell-1}, \ldots, \beta^1$ are all outputs of Algorithm 2.1, they are all CW minima (by Theorem 2.2). Any CW minimum on a support S is stationary for the problem: $\min_{\beta_S} f(\beta_S)$. By the convexity of $\beta_S \to f(\beta_S)$, all stationary solutions on support S have the same objective (since they all correspond to the minimum of $\min_{\beta_S} f(\beta_S)$). Thus, we have $\text{Supp}(\beta^i) \neq \text{Supp}(\beta^j)$ for any $1 \leq i, j \leq \ell$ such that $i \neq j$. Therefore, a support

can appear at most once during the course of Algorithm 2.2. Since the number of possible supports is finite, we conclude that Algorithm 2.2 terminates in a finite number of iterations. Finally, we note that Algorithm 2.2 terminates iff there is no feasible solution $\widehat{\beta}$ for (2.14) satisfying $F(\widehat{\beta}) < F(\beta^\ell)$. This implies that β^ℓ is a minimizer of (2.14) and thus a PSI(k) minimum (by Definition 2.3). □

Proof of Lemma 2.7

Proof. Let us consider the case where, $\lambda_0^{i+1} < M^i$. It follows from (2.26) that:

$$\max_{j \in S^c} \frac{|\ |\langle r, X_j\rangle| - \lambda_1|}{1 + 2\lambda_2} > \sqrt{\frac{2\lambda_0^{i+1}}{1 + 2\lambda_2}}, \tag{2.45}$$

which implies that $\beta^{(i)}$ is not a CW minimum for the given λ_0^{i+1} (see (2.8)). By Theorem 2.2, Algorithm 2.1 converges to a CW minimum. Therefore, Algorithm 2.1 initialized with $\beta^{(i)}$ leads to $\beta^{(i+1)} \neq \beta^{(i)}$.

We now consider the case where, $\lambda_0^{i+1} \in (M^i, \lambda_0^i]$. Then (2.26) implies

$$\max_{j \in S^c} \frac{|\ |\langle r, X_j\rangle| - \lambda_1|}{1 + 2\lambda_2} < \sqrt{\frac{2\lambda_0^{i+1}}{1 + 2\lambda_2}} \leq \sqrt{\frac{2\lambda_0^{i}}{1 + 2\lambda_2}}. \tag{2.46}$$

Also, since $\beta^{(i)}$ is a CW minimum for $\lambda = \lambda_0^i$, we have for every $j \in S$

$$|\beta_j^{(i)}| \geq \sqrt{\frac{2\lambda_0^{i}}{1 + 2\lambda_2}} \geq \sqrt{\frac{2\lambda_0^{i+1}}{1 + 2\lambda_2}}, \tag{2.47}$$

where, the second inequality follows from $\lambda_0^{i+1} \leq \lambda_0^i$. The condition $\nabla_S f(\beta_S^{(i)}) = 0$ along with inequalities (2.46) and (2.47) imply that $\beta^{(i)}$ is a CW minimum for Problem (2.2) at $\lambda_0 = \lambda_0^{i+1}$. Therefore, $\beta^{(i)}$ is a fixed point for Algorithm 2.1. □

2.B Appendix: Additional Experimental Results

2.B.1 Oracle Tuning

Statistical Performance for Varying Number of Samples: The results are reported in Figures 2-8 and 2-9.

Statistical Performance for Varying SNR: The results are reported in Figures 2-10 and 2-11.

2.B.2 Random Design Tuning

We use the same definition of Relative Risk (RR) as in [83]: given an estimator $\hat{\beta}$, RR= $\frac{(\hat{\beta}-\beta^\dagger)^T\Sigma(\hat{\beta}-\beta^\dagger)}{(\beta^\dagger)^T\Sigma\beta^\dagger}$.

Statistical Performance for Varying Number of Samples: The results are reported in Figures 2-12 and 2-13.

Statistical Performance for Varying SNR: The results are reported in Figures 2-14 and 2-15.

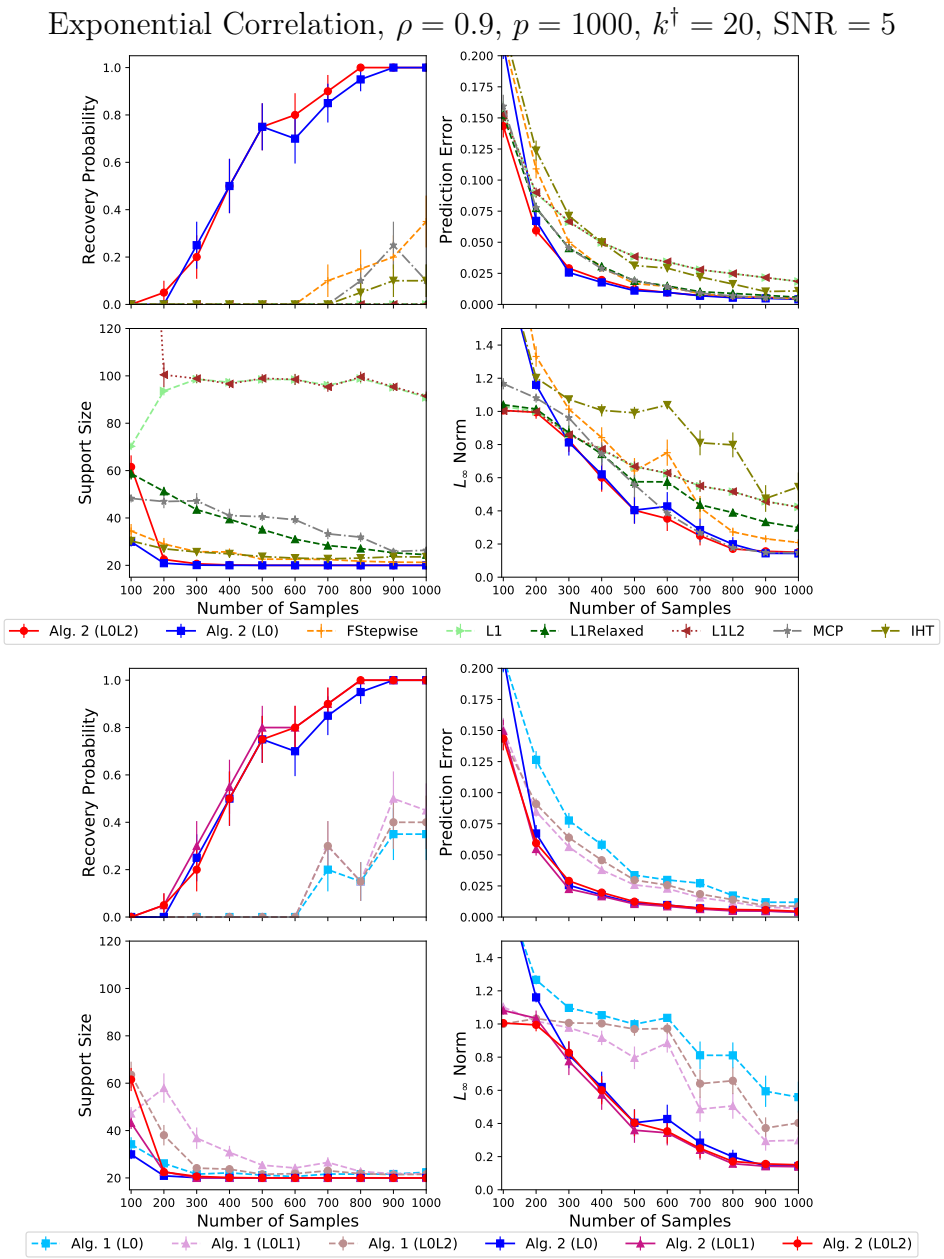


Figure 2-8: Performance measures as the number of samples n varies between 100 and 1000. The top figure compares Algorithm 2.2 (L_0), Algorithm 2.2 (L_0L_2), and other state-of-the-art algorithms. The bottom figure compares all of our proposed algorithms.

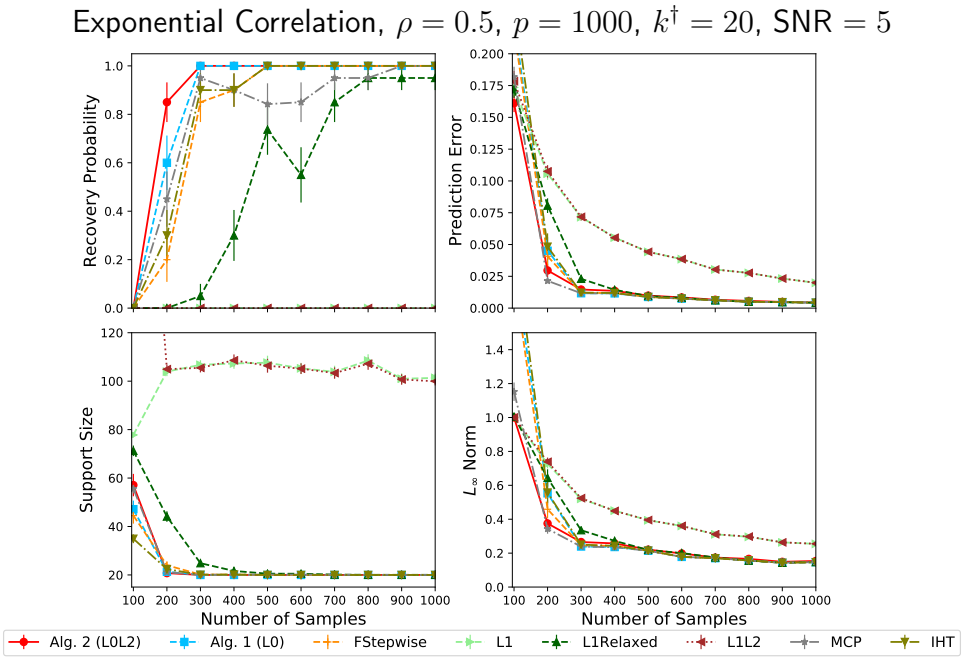


Figure 2-9: Performance measures as the number of samples n varies between 100 and 1000. The figure compares Algorithm 2.1 (L_0), Algorithm 2.2 (L_0L_2), and other state-of-the-art algorithms.

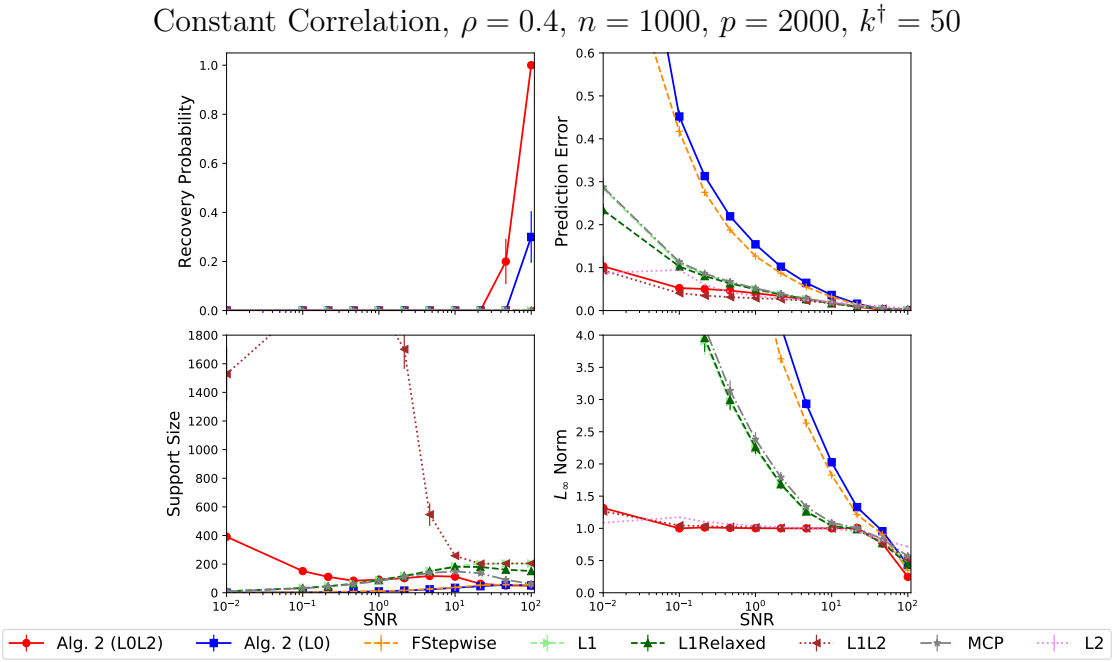


Figure 2-10: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.01 and 100. The figure compares two of our methods and other state-of-the-art algorithms.

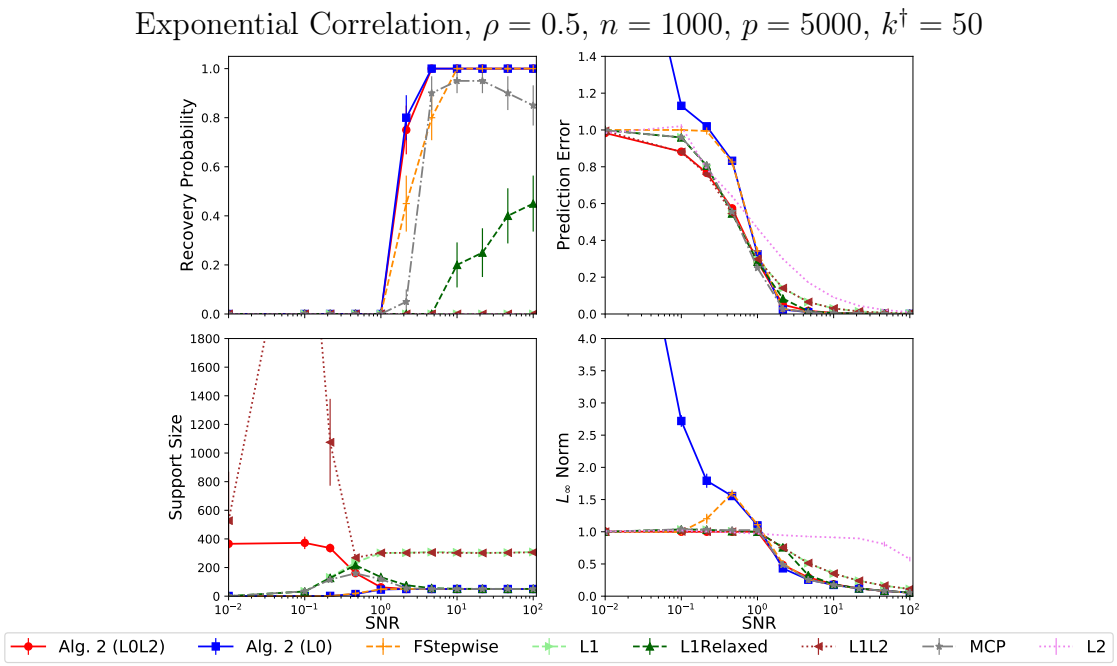


Figure 2-11: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.01 and 100. The figure compares two of our methods and other state-of-the-art algorithms.

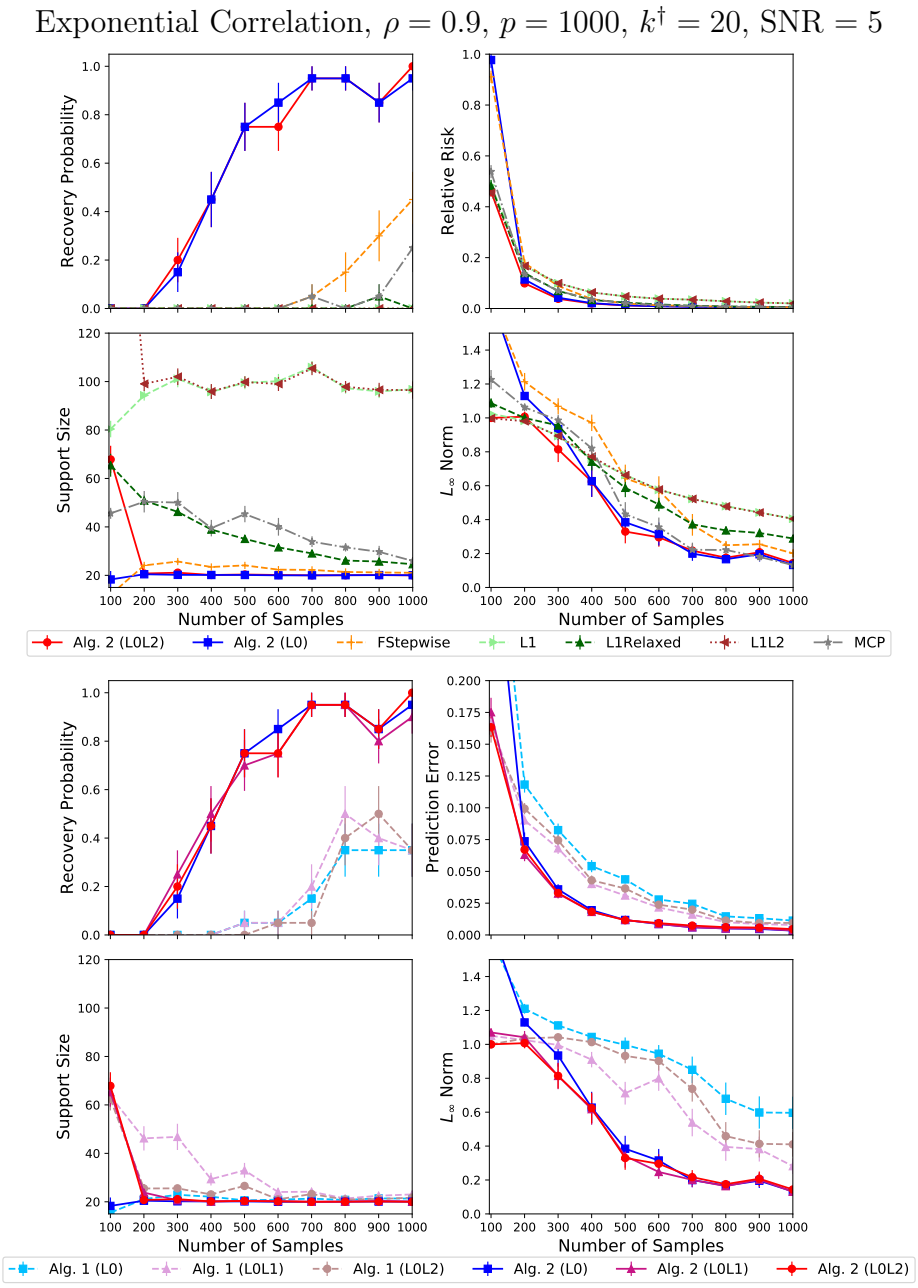


Figure 2-12: Performance measures as the number of samples n varies between 100 and 1000. The top figure compares Algorithm 2.2 (L_0), Algorithm 2.2 (L_0L_2), and other state-of-the-art algorithms. The bottom figure compares all of our proposed algorithms.

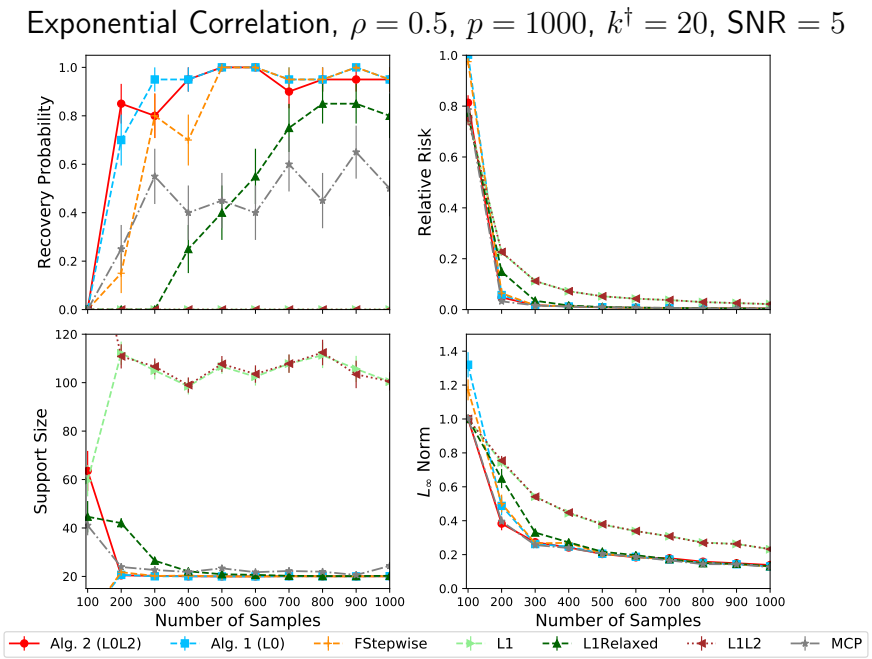


Figure 2-13: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.01 and 100. The figure compares two of our methods and other state-of-the-art algorithms.

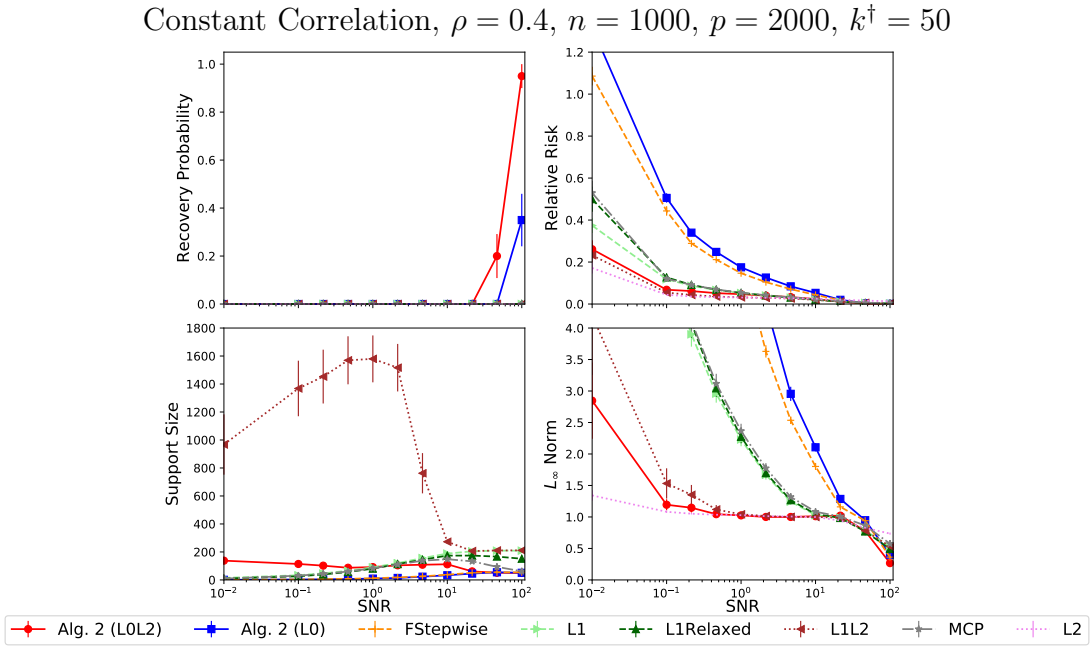


Figure 2-14: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.01 and 100. The figure compares two of our methods and other state-of-the-art algorithms.

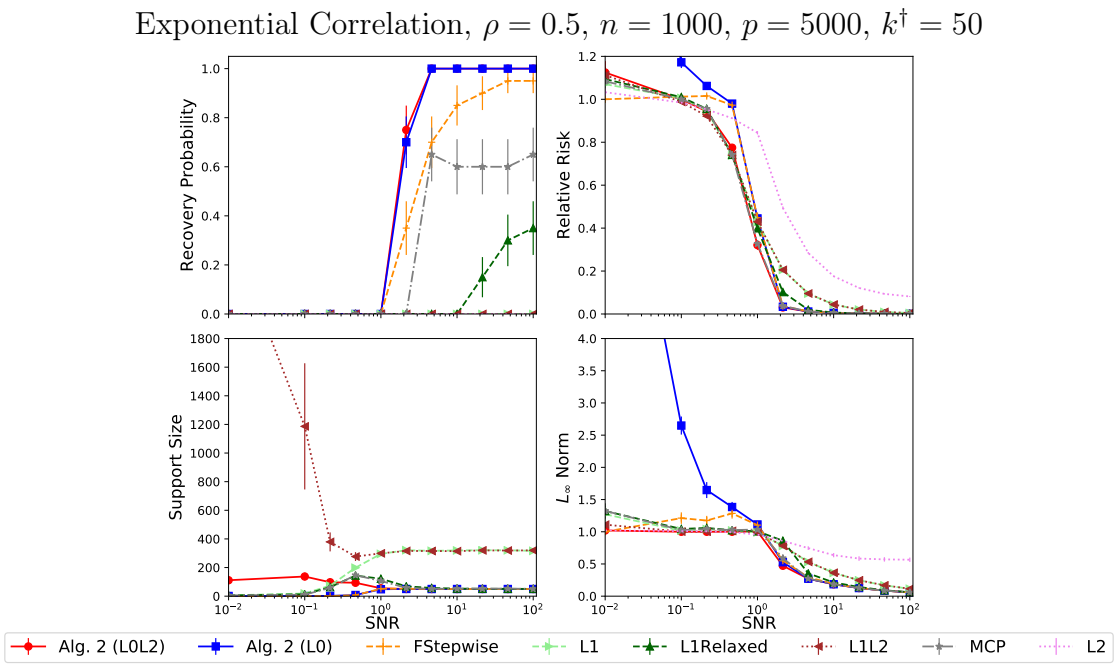


Figure 2-15: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.01 and 100. The figure compares two of our methods and other state-of-the-art algorithms.

Chapter 3

Sparse Regression at Scale: Branch-and-Bound rooted in First-Order Optimization

This chapter is based on [90]. It is a joint work with Rahul Mazumder and Ali Saab.

3.1 Introduction

We consider the sparse linear regression problem with additional ℓ_2 regularization [28, 86]. A natural way to impose sparsity in this context is through controlling the ℓ_0 (pseudo) norm of the estimator, which counts the number of nonzero entries. More concretely, let $X \in \mathbb{R}^{n\times p}$ be the data matrix, with n samples and p features, and $y \in \mathbb{R}^n$ be the response vector. We focus on the least squares problem with a combination of ℓ_0 and ℓ_2 regularization:

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0\|\beta\|_0 + \lambda_2\|\beta\|_2^2, \tag{3.1}$$

where $\|\beta\|_0$ is defined as the number of nonzero entries in the regression coefficients $\beta \in \mathbb{R}^p$, and $\|\beta\|_2^2$ is the squared ℓ_2-norm of β (also referred to as ridge regularization). The

regularization parameters λ_0 and λ_2 are assumed to be specified by the practitioner[1]. We note that the presence of ridge regularization (i.e., $\lambda_2 > 0$) can be important from a statistical viewpoint—see for example [83, 115, 86] for further discussions on this matter. Statistical properties of ℓ_0-based estimators have been extensively studied in the statistics literature [73, 147, 189, 59, 172, 115]. Specifically, under suitable assumptions on the underlying data and appropriate choices of tuning parameters, global solutions of (3.1) have optimal support recovery properties [172, 66]; and optimal prediction error bounds that do not depend upon X [147]. Appealing prediction error bounds available from optimal solutions of (3.1) for the low-signal regime are discussed in [115]. Such strong guarantees are generally not satisfied by heuristic solutions to (3.1)—see [189] for theoretical support and [86] for numerical evidence. From a practical perspective, the ability to certify the optimality of solutions (e.g., via dual bounds) is important and can engender trust in mission critical applications such as healthcare.

Despite its appeal, Problem (3.1) is labeled as NP-Hard [128] and poses computational challenges. Recently, there has been exciting work in developing Mixed Integer Programming (MIP)-based approaches to solve (3.1), e.g., [57, 24, 123, 61, 28, 86, 26, 180, 8]. Specifically, [24] demonstrated that off-the-shelf MIP solvers can handle problem instances for p up to a thousand. Larger instances can be handled when λ_2 is sufficiently large and the feature correlations are low—for example, see the approach of [28]; and the method in [60] for the classification variant of (3.1). The approaches of [28, 60] rely on commercial MIP solvers such as Gurobi. While these state-of-the-art global optimization approaches show very promising results, they are still relatively slow for practical usage [83, 86]. For example, our experiments show that these methods cannot terminate in two hours for typical instances with $p \sim 10^4$. On the other hand, the fast Lasso solvers, e.g., `glmnet` [69], and local optimization methods for (3.1), such as `L0Learn` [86], can handle much larger instances, and they typically terminate in the order of milliseconds to seconds.

Our goal in this chapter is to advance the computational methodology for the global op-

[1]The choice of (λ_0, λ_2) depends upon the particular application and/or dataset. We aim to compute solutions to (3.1) for a family of (λ_0, λ_2)-values.

timization of Problem (3.1). In particular, we aim to (i) reduce the run time for solving problem instances with $p \sim 10^4$ from hours to seconds, and (ii) scale to larger problem instances with $p \sim 10^7$ in reasonable times (order of minutes to hours). To this end, we propose a specialized nonlinear branch-and-bound (BnB) framework that does not rely on commercial MIP solvers. We employ a first-order method, which carefully exploits the problem structure, to solve the node relaxations. This makes our approach quite different from prior work on global optimization for Problem (3.1), which rely on commercial MIP solvers, e.g., Gurobi and CPLEX. These MIP solvers are also based on a BnB framework, but they are equipped with general-purpose relaxation solvers and heuristics that do not take into account the specific structure in Problem (3.1).

Our BnB solves a mixed integer second order cone program (MISOCP) that is based on a perspective reformulation [67, 4, 74] of Problem (3.1). The algorithm exploits the sparsity structure in the problem during different stages: when solving node relaxations, branching, and obtaining upper bounds. The continuous node relaxations that appear in our BnB have not been studied at depth in earlier work. A main contribution of our work is to show that these relaxations, which involve seemingly complicated linear and conic constraints, can be efficiently handled using a primal coordinate descent (CD)-based algorithm. Indeed, this represents a radical change from the primal-dual relaxation solvers commonly used in state-of-the-art MIP solvers [20]. Our choice of CD is motivated by its ability to effectively share information across the BnB nodes (such as warm starts), and more generally by its high scalability in the context of sparse learning, e.g., see [69, 117, 86]. Along with CD, we propose additional strategies, namely, active set updates and gradient screening, which reduce the coordinate update complexity by exploiting the information shared across the BnB tree.

Although our CD-based algorithm for solving BnB node relaxations is highly scalable, it only generates primal solutions. However, dual bounds are required for search space pruning in BnB. Thus, we propose a novel method to efficiently generate dual bounds from the primal solutions. We analyze these dual bounds and prove that their tightness is not affected by the number of features p, but rather by the number of nonzeros in the primal solution. This

result serves as a theoretical justification for why our CD-based algorithm can lead to tight dual bounds in high dimensions.

Contributions and Structure: We summarize our key contributions below.

- We formulate Problem (3.1) as a MISOCP, based on a perspective formulation. We provide a new analysis of the relaxation tightness, which identifies parameter ranges for which the perspective formulation can outperform popular formulations (see Section 3.2).

- To solve the MISOCP, we design a specialized nonlinear BnB, with the following main contributions (see Section 3.3):

  - We show that the node relaxations, which involve linear and conic constraints, can be reformulated as a least squares problem with a non-differentiable but separable penalty. To solve the latter reformulation, we develop a primal CD algorithm, along with active set updates and gradient screening that use information shared across the BnB tree to reduce the coordinate update cost.

  - We develop a new efficient method for obtaining dual bounds from the primal solutions. We analyze these dual bounds and show that their tightness depends on the sparsity level rather than p.

  - We introduce efficient methods that exploit sparsity when selecting branching variables and obtaining incumbents[2].

- We perform a series of experiments on high-dimensional synthetic and real datasets, with p up to 8.3×10^6. We study the effect of the regularization parameters and dataset characteristics on the run time, and perform ablation studies. The results indicate that our approach can be 5000x faster than the state of the art in some settings, and is capable of handling difficult statistical instances which were virtually unsolvable before (See Section 3.4). We open source the implementation through our toolkit L0BnB:

[2]An incumbent, in the context of BnB, refers to the best integral solution found so far.

https://github.com/alisaab/L0BnB

Related Work: As mentioned earlier, an impressive line of recent work considers solving Problem (3.1), or its cardinality-constrained variant, to optimality. [24] used Gurobi on a Big-M formulation, which can handle $n \sim p \sim 10^3$ in the order of minutes to hours. [28] scale the problem even further by applying outer-approximation (using Gurobi) on a boolean reformulation [143] of the problem. Their approach can handle $p \sim 10^5$ in the order of minutes when n and λ_2 are sufficiently large, and the feature correlations are sufficiently small. This outer-approximation approach has also been generalized to sparse classification in [25]. [180] consider solving a perspective formulation [67] of the problem directly using Gurobi and reported timings that compare well with [28]—the largest problem instances they consider have $p \sim 10^3$. [60] show that a variant of Problem (3.1) for classification can be solved through a sequence of MIPs (solved using Gurobi), each having a small number of binary variables, as opposed to p binary variables in the common approaches. Their approach can handle $n = 10^3$ and $p = 50,000$ in minutes if the feature correlations are sufficiently small. Our specialized BnB, on the other hand, can solve all the instances mentioned above with speed-ups that exceed 5000x, and can scale to problems with $p \sim 10^7$. Moreover, our numerical experiments show that, unlike prior work, our BnB can handle difficult problems with relatively small λ_2 and/or high feature correlations.

In addition to the global optimization approaches discussed above, there is an interesting body of work in the broader optimization community on improved relaxations for sparse ridge regression, e.g., [143, 61, 7, 180], and algorithms that locally optimize an ℓ_0-based objective [36, 16, 86].

There is also a rich literature on solving mixed integer nonlinear programs (MINLPs) using BnB, e.g., see [105, 20]. Our approach is based on the nonlinear BnB framework [58], where a nonlinear subproblem is solved at every node of the search tree. Interior point methods are a popular choice for these nonlinear subproblems, especially for MISOCPs [105], e.g., they are used in MOSEK [5] and are also one of the supported options in CPLEX [156]. Generally, interior point based nonlinear solvers are not as effective in exploiting warm starts and

sparsity as linear programming solvers [20], which led to an alternative approach known as outer-approximation (OA) [63]. In OA, a sequence of relaxations, consisting of mixed integer linear programs, are solved until converging to a solution of the MINLP. State-of-the-art solvers such as BARON [160] and Gurobi [76] apply OA on extended formulations—[160] laid the ground work for this approach. There is also a line of work on specialized OA reformulations and algorithms for mixed integer conic programs (which include MISOCPs), e.g., see [169, 113, 170] and the references therein. In this chapter, we pursue a different approach: making use of problem-specific structure, we show that the relaxation of the MISOCP can be effectively handled by our proposed CD-based algorithm.

Notation and Proofs: We denote the set $\{1, 2, \ldots, p\}$ by $[p]$. For any set A, the complement is denoted by A^c. We let $\|\cdot\|_q$ denote the standard ℓ_q norm with $q \in \{0, 1, 2, \infty\}$. For any vector $v \in \mathbb{R}^k$, $\text{sign}(v) \in \mathbb{R}^k$ refers to the vector whose ith component is given by $\text{sign}(v_i) = v_i/|v_i|$ if $v_i \neq 0$ and $\text{sign}(v_i) \in [-1, 1]$ if $v_i = 0$. We denote the support of $\beta \in \mathbb{R}^p$ by $\text{Supp}(\beta) = \{i : \beta_i \neq 0, i \in [p]\}$. For a set $S \subseteq [p]$, use $\beta_S \in \mathbb{R}^{|S|}$ to denote the subvector of β with indices in S. Similarly, X_S refers to the submatrix of X whose columns correspond to S. For a scalar a, we denote $[a]_+ = \max\{a, 0\}$. Given a set of real numbers $\{a_i\}_{i=1}^N$ and a scalar c, we use $\{a_i\}_{i=1}^N \cdot c$ to denote $\{ca_i\}_{i=1}^N$. The proofs of all propositions, lemmas, and theorems are in the appendix of this chapter.

3.2 MIP Formulations and Relaxations

In this section, we present MIP formulations for Problem (3.1) and study their corresponding relaxations.

3.2.1 MIP Formulations

The Big-M Formulation: We assume that there is a finite scalar $M > 0$ (a-priori specified) such that an optimal solution of Problem (3.1), say β^*, satisfies: $\|\beta^*\|_\infty \leq M$. This allows for modeling (3.1) as a mixed integer quadratic program (MIQP) using the following Big-M

formulation:

$$\begin{aligned} \mathrm{B}(M): \quad \min_{\beta,z} \quad & \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0 \sum_{i\in[p]} z_i + \lambda_2 \|\beta\|_2^2 \\ \text{s.t.} \quad & -Mz_i \leq \beta_i \leq Mz_i, \ i \in [p] \\ & z_i \in \{0,1\}, \ i \in [p], \end{aligned} \tag{3.2}$$

where each binary variable z_i controls whether β_i is zero or not via the first constraint in (3.2)—i.e., if $z_i = 0$ then $\beta_i = 0$. Such Big-M formulations are widely used in mixed integer programming and have been recently explored in multiple works on ℓ_0 regularization, e.g., [24, 115, 86, 180]. See [24, 180] for a discussion on how to estimate M in practice.

The Perspective Formulation: In MIP problems where bounded continuous variables are activated by indicator variables, perspective reformulations [67, 4, 74] can lead to stronger MIP relaxations and thus improve the run time of BnB algorithms. Here, we apply a perspective reformulation to the ridge term $\|\beta\|_2^2$ in Problem (3.2). Specifically, we introduce the auxiliary continuous variables $s_i \geq 0, i \in [p]$ and rotated second order cone constraints $\beta_i^2 \leq s_i z_i, \ i \in [p]$. We then replace the term $\|\beta\|_2^2$ with $\sum_{i\in[p]} s_i$. Thus, each s_i takes the place of β_i^2. This leads to the following reformulation of (3.2):

$$\begin{aligned} \mathrm{PR}(M): \quad \min_{\beta,z,s} \quad & \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0 \sum_{i\in[p]} z_i + \lambda_2 \sum_{i\in[p]} s_i \\ \text{s.t.} \quad & \beta_i^2 \leq s_i z_i, \ i \in [p] \\ & -Mz_i \leq \beta_i \leq Mz_i, \ i \in [p] \\ & z_i \in \{0,1\}, s_i \geq 0, \ i \in [p]. \end{aligned} \tag{3.3}$$

Problem (3.3) can be expressed as a MISOCP. Similar to (3.2), formulation (3.3) is equivalent to Problem (3.1) (as long as M is suitably chosen). Algorithms for formulation (3.3) will be the main focus of this chapter.

If we set $M = \infty$ in (3.3), then the constraints $\beta_i \in [-Mz_i, Mz_i], i \in [p]$ can be dropped, which makes PR(∞) independent of a Big-M parameter. If $\lambda_2 > 0$, then PR(∞) is equivalent to (3.1)—this holds since $\beta_i^2 \leq s_i z_i$ enforces $z_i = 0 \implies \beta_i = 0$. We note that [61] have studied Problem (3.3) and focused on the special case of PR(∞). [61] shows that PR(∞) is

equivalent to the pure binary formulations considered in [143, 28]. We also note that [180] have considered a similar perspective formulation for the cardinality constrained variant of Problem (3.1). In Proposition 3.1 below, we present new bounds that quantify the relaxation strengths of PR(M), PR(∞), and B(M). Moreover, we are the first to present a tailored BnB procedure for formulation (3.3).

3.2.2 Relaxation of the Perspective Formulation

In this section, we study the *interval relaxation* of Problem (3.3), which is obtained by relaxing all binary z_i's to the interval $[0, 1]$. Specifically, we present a new compact reformulation of the interval relaxation that leads to useful insights and facilitates our algorithm development. We also discuss how this reformulation compares with the interval relaxations of B(M) and PR(∞).

Theorem 3.1 shows that the interval relaxation of (3.3) can be reformulated purely in the β space. This leads to a regularized least squares criterion, where the regularizer involves the reverse Huber [136] penalty—a hybrid between the ℓ_1 and ℓ_2 (squared) penalties. The reverse Huber penalty $\mathcal{B} : \mathbb{R} \to \mathbb{R}$, is given by:

$$\mathcal{B}(t) = \begin{cases} |t| & |t| \leq 1 \\ (t^2 + 1)/2 & |t| \geq 1. \end{cases} \tag{3.4}$$

Theorem 3.1. *(Reduced Relaxation) Let us define the functions ψ, ψ_1, ψ_2 as*

$$\psi(\beta_i; \lambda_0, \lambda_2, M) = \begin{cases} \psi_1(\beta_i; \lambda_0, \lambda_2) := 2\lambda_0 \mathcal{B}(\beta_i \sqrt{\lambda_2/\lambda_0}) & \text{if } \sqrt{\lambda_0/\lambda_2} \leq M \\ \psi_2(\beta_i; \lambda_0, \lambda_2, M) := (\lambda_0/M + \lambda_2 M)|\beta_i| & \text{if } \sqrt{\lambda_0/\lambda_2} > M. \end{cases}$$

The interval relaxation of (3.3) *is equivalent to:*

$$\min_{\beta \in \mathbb{R}^p} \; F(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i \in [p]} \psi(\beta_i; \lambda_0, \lambda_2, M) \quad s.t. \quad \|\beta\|_\infty \leq M, \tag{3.5}$$

and we let $V_{PR(M)}$ denote the optimal objective value of (3.5).

The reduced formulation (3.5) has an important role in both the subsequent analysis and the algorithmic development in Section 3.3. Theorem 3.1 shows that the conic and Big-M constraints in the relaxation of (3.3) can be completely eliminated, at the expense of introducing (in the objective) the non-differentiable penalty function $\sum_i \psi(\beta_i; \lambda_0, \lambda_2, M)$ which is separable across the p coordinates $\beta_i, i \in [p]$. Depending on the value of $\sqrt{\lambda_0/\lambda_2}$ compared to M, the penalty $\psi(\beta_i; \lambda_0, \lambda_2, M)$ is either the reverse Huber penalty (i.e., ψ_1) or the ℓ_1 penalty (i.e., ψ_2), both of which are sparsity-inducing. In Figure 3-1 (left panel), we plot $\psi_1(\beta; \lambda_0, \lambda_2)$ for $\lambda_2 = 1$ at different values of λ_0. In Figure 1 (right panel), we plot $\psi(\beta; \lambda_0, \lambda_2, M)$ at $\lambda_0 = \lambda_2 = 1$ and for different values of M. The appearance of a pure ℓ_1 penalty in the objective is interesting in this case since the original formulation in (3.3) has a ridge term in the objective. Informally speaking, when $\sqrt{\lambda_0/\lambda_2} > M$, the constraint $|\beta_i| \leq M z_i$ becomes active at any optimal solution, which turns the ridge term into an ℓ_1 penalty—for further discussion on this matter, see the proof of Theorem 3.1.

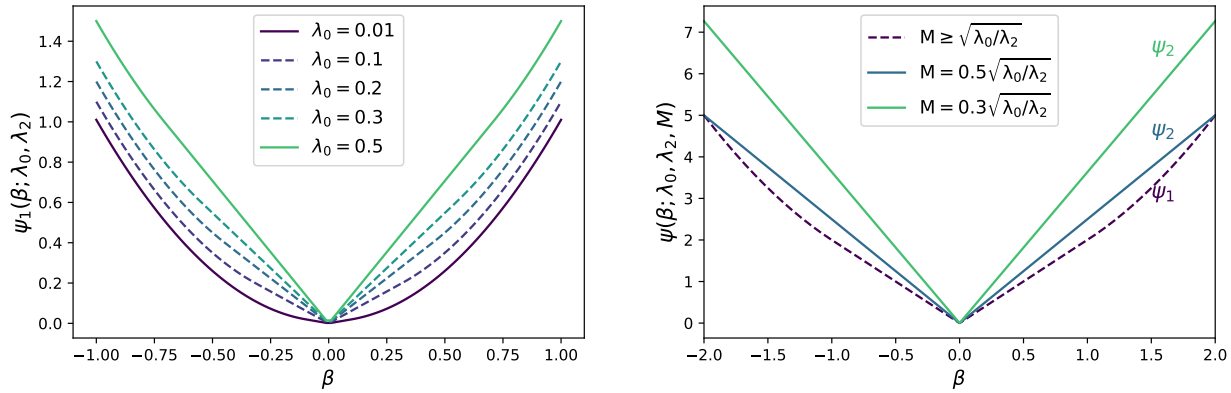


Figure 3-1: [Left]: Plot of $\psi_1(\beta; \lambda_0, 1)$ for different values of λ_0. [Right]: Plot of $\psi(\beta; 1, 1, M)$ for different values of M.

Next, we analyze the tightness of the perspective relaxation in (3.5).

Tightness of the Perspective Relaxation (3.5)**:** [61] has shown that the interval relax-

ation of PR(∞) can be written as:

$$V_{\mathrm{PR}(\infty)} = \min_{\beta \in \mathbb{R}^p} G(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i \in [p]} \psi_1(\beta_i; \lambda_0, \lambda_2), \tag{3.6}$$

where $\psi_1(\beta_i; \lambda_0, \lambda_2)$ is defined in Theorem 3.1. Note that (3.6) can also be obtained from Theorem 3.1 (with $M = \infty$). For $\sqrt{\lambda_0/\lambda_2} \leq M$, the relaxation of PR($M$) in Theorem 3.1 matches that in (3.6), but with the additional box constraint $\|\beta\|_\infty \leq M$. When M is large, this box constraint becomes inactive, making the interval relaxations of PR(M) and PR(∞) equivalent. However, when $\sqrt{\lambda_0/\lambda_2} > M$, the interval relaxation of PR(M) can have a (strictly) larger objective than that of both B(M) and PR(∞), as we show in Proposition 3.1.

Proposition 3.1. *Let $V_{B(M)}$ denote the optimal objective of the interval relaxation of $B(M)$. Let β^* be an optimal solution to* (3.5)*, and define the function $h(\lambda_0, \lambda_2, M) = \lambda_0/M + \lambda_2 M - 2\sqrt{\lambda_0\lambda_2}$. Then, the following holds for $\sqrt{\lambda_0/\lambda_2} > M$:*

$$V_{PR(M)} \geq V_{B(M)} + \lambda_2(M\|\beta^*\|_1 - \|\beta^*\|_2^2) \tag{3.7}$$

$$V_{PR(M)} \geq V_{PR(\infty)} + h(\lambda_0, \lambda_2, M)\|\beta^*\|_1. \tag{3.8}$$

We make a couple of remarks. For bound (3.7), we always have $(M\|\beta^*\|_1 - \|\beta^*\|_2^2) \geq 0$ since $\|\beta^*\|_\infty \leq M$. If there is an $i \in [p]$ with $0 < |\beta_i^*| < M$, then $(M\|\beta^*\|_1 - \|\beta^*\|_2^2) > 0$, and consequently $V_{\mathrm{PR}(M)} > V_{\mathrm{B}(M)}$ (as long as $\lambda_2 > 0$). In bound (3.8), for $\sqrt{\lambda_0/\lambda_2} > M$, $h(\lambda_0, \lambda_2, M)$ is strictly positive and monotonically decreasing in M, which implies that $V_{\mathrm{PR}(M)} > V_{\mathrm{PR}(\infty)}$ (as long as $\beta^* \neq 0$). In Section 3.4.2, our experiments empirically validate Proposition 3.1: using PR(M) with a sufficiently tight (but valid) M can speed up the same BnB solver by more than 90x compared to PR(∞).

Note that Proposition 3.1 does not directly compare $V_{\mathrm{PR}(\infty)}$ with $V_{\mathrm{B}(M)}$. In Proposition 3.2, we establish a new result which compares $V_{\mathrm{PR}(\infty)}$ with $V_{\mathrm{B}(M)}$. Before we present Proposition 3.2, we introduce some notation. Let $\mathcal{S}(\lambda_2)$ be the set of optimal solutions (in β) of the

interval relaxation of PR(∞); and define

$$\mathcal{L}(M) := \{\lambda_2 > 0 \mid \exists \beta \in \mathcal{S}(\lambda_2) \text{ s.t. } \|\beta\|_\infty \leq M\}. \tag{3.9}$$

Proposition 3.2. *Let $\mathcal{L}(M)$ be as defined in* (3.9)*. Then, the following holds:*

$$V_{B(M)} \geq V_{PR(\infty)}, \text{ if } M \leq \tfrac{1}{2}\sqrt{\lambda_0/\lambda_2} \tag{3.10}$$

$$V_{B(M)} \leq V_{PR(\infty)}, \text{ if } M \geq \sqrt{\lambda_0/\lambda_2} \text{ and } \lambda_2 \in \mathcal{L}(M). \tag{3.11}$$

Proposition 3.2 implies that if λ_2 is relatively small (with other parameters remaining fixed), then the relaxation of the Big-M formulation (with objective $V_{\mathrm{B}(M)}$) will have a higher objective than the relaxation of PR(∞). On the other hand, if λ_2 is sufficiently large, then the relaxation of PR(∞) will have a higher objective.

We note that the result of Proposition 3.2 applies for any $M \geq 0$, even if it is mis-specified. However, in the case of mis-specification, $V_{\mathrm{B}(M)}$ (and also $V_{\mathrm{PR}(M)}$) may no longer correspond to a valid relaxation of Problem (3.1). In contrast, $V_{\mathrm{PR}(\infty)}$ is always a valid relaxation of (3.1).

3.3 A Specialized Branch-and-Bound (BnB) Framework

In this section, we develop a specialized nonlinear BnB framework for solving the perspective formulation in (3.3). First, we briefly recall the high-level mechanism behind nonlinear BnB, in the context of our problem.

Nonlinear BnB at a Glance: The algorithm starts by solving the (nonlinear) interval relaxation of (3.3), i.e., the root node. Then, it selects a branching variable, say variable $j \in [p]$, and creates two new nodes (optimization subproblems): one node with $z_j = 0$ and another with $z_j = 1$, where all the other z_i's are relaxed to the interval $[0, 1]$. For every unvisited node, the algorithm proceeds recursively, i.e., by solving an optimization

subproblem at the current node and then branching on a new variable to create two new nodes. This leads to a search tree with nodes corresponding to optimization subproblems and edges representing branching decisions.

To reduce the size of the search tree, BnB prunes a node (i.e., does not branch on it) in either one of the following situations: (i) an optimal solution to the relaxation at the current node has an integral z or (ii) the objective of the current relaxation exceeds the best available upper bound on (3.3). In case (ii), the relaxation need not be solved exactly: lower bounds on the relaxation's objective (a.k.a. dual bounds) can be used for pruning. However, in case (i), if the dual bound does not exceed the best upper bound, the node should be solved to optimality in order to ensure correctness of the pruning decision[3].

As BnB explores more nodes, its (global) lower bound is guaranteed to converge to the optimal objective of (3.3). In practice, we can terminate the algorithm early, if the gap between the best lower and upper bounds is below a pre-specified user-defined threshold.

Overview of our Strategies: The discussion above outlines how nonlinear BnB operates in general. The specific strategies used such as solving the relaxations, passing information across the nodes, and selecting branching variables, can have a key impact on scalability. In the rest of this section, we will give a detailed account of the strategies used in our BnB. We first provide an overview of these strategies:

- **A Primal Relaxation Solver**: Unlike state-of-the-art approaches for nonlinear BnB, which employ primal-dual interior point solvers for node relaxations [20], we rely solely on a primal method which consists of a highly scalable CD-based algorithm. The algorithm solves the node relaxations of (3.3) in the β-space as opposed to the extended (β, s, z) space—these relaxations are variants of the reduced relaxation introduced in (3.5). The algorithm heavily shares and exploits warm starts, active sets, and information on the gradients, across the BnB tree. This will be developed in Section 3.3.1.

[3]In practice, we solve the relaxation problem at the node to optimality by ensuring that the relative difference between the primal and dual bounds is less than a small user-defined numerical tolerance.

- **Dual Bounds**: Dual bounds on the objective of node relaxations are required by BnB for search space pruning, yet our relaxation solver works in the primal space for scalability considerations. We develop a new efficient method for obtaining dual bounds from the primal solutions. We provide an analysis of this method and show that the tightness of the dual bounds depends on the sparsity level and *not* on the number of features p. See Section 3.3.2.

- **Branching and Incumbents**: We present an efficient variant of strong branching, which leverages the solutions and active sets of previous node relaxations to make optimization tractable. Moreover, we employ several efficient heuristics to obtain incumbents. See Section 3.3.3.

For simplicity of exposition, in the remainder of Section 3.3, we assume that the columns of X and y have unit ℓ_2 norm.

3.3.1 Primal Relaxation Solver: Active-set Coordinate Descent

To simplify the presentation, we will focus on solving the root relaxation. To this end, we solve the reduced formulation in the β-space (3.5). We operate on the reduced formulation compared to the interval relaxation of (3.3) in the extended (β, s, z) space due to computational reasons. After solving (3.5), we use the resulting solution, say β^*, to construct a corresponding solution (β^*, s^*, z^*) to the interval relaxation of (3.3)—see the proof of Theorem 3.1 for how to obtain (β^*, s^*, z^*) from β^*. We then use z^* for branching. The rest of the nodes in BnB involve fixing some binary variables in (3.3) to 0 or 1, so their subproblems can be obtained by minor modifications to the root relaxation. For completeness, in Appendix 3.B.2, we discuss how to formulate and solve these node subproblems.

Problem (3.5) is of the composite form [132]: the objective is the sum of a smooth loss function and a non-smooth but separable penalty. In addition, the feasible set, consisting of the constraints $|\beta_i| \leq M$, $i \in [p]$ is separable across the coordinates. This makes Problem (3.5) amenable to cyclic CD [165]. To our knowledge, the use of cyclic CD for Problem (3.5) is novel. We also emphasize that a direct application of cyclic CD to Problem (3.5) will

face scalability issues, and more importantly, it does not readily deliver dual bounds. The additional strategies we develop later in this section are essential for both achieving scalability and obtaining (provably) high quality dual bounds.

Cyclic CD visits the coordinates according to a fixed ordering, updating one coordinate at a time, as detailed in Algorithm 3.1.

Algorithm 3.1: Cyclic CD for Relaxation (3.5)

- **Input:** Initialization $\hat{\beta}$
- **While** not converged:
  - **For** $i \in [p]$:

$$\hat{\beta}_i \leftarrow \underset{\beta_i \in \mathbb{R}}{\arg\min}\, F(\hat{\beta}_1, \ldots, \beta_i, \ldots, \hat{\beta}_p) \quad \text{s.t.} \quad |\beta_i| \leq M. \tag{3.12}$$

Every limit point of Algorithm 3.1 is a global minimizer of (3.5) [165], and the algorithm has a sublinear rate of convergence [95]. We will show that the solution of (3.12) can be computed in closed-form. To this end, since the columns of X have unit ℓ_2-norm, we note that (3.12) is equivalent to:

$$\min_{\beta_i \in \mathbb{R}} \quad \frac{1}{2}(\beta_i - \tilde{\beta}_i)^2 + \psi(\beta_i; \lambda_0, \lambda_2, M) \quad \text{s.t.} \quad |\beta_i| \leq M, \tag{3.13}$$

where $\tilde{\beta}_i := \langle y - \sum_{j \neq i} X_j \hat{\beta}_j, X_i \rangle$. Given non-negative scalar parameters a and m, we define the *boxed soft-thresholding operator* $T : \mathbb{R} \to \mathbb{R}$ as

$$T(t; a, m) := \begin{cases} 0 & \text{if } |t| \leq a \\ (|t| - a)\,\text{sign}(t) & \text{if } a < |t| \leq a + m \\ m\,\text{sign}(t) & \text{otherwise.} \end{cases}$$

For $\sqrt{\lambda_0/\lambda_2} \leq M$, the solution of (3.13) is given by:

$$\hat{\beta}_i = \begin{cases} T(\tilde{\beta}_i; 2\sqrt{\lambda_0\lambda_2}, M) & \text{if } |\tilde{\beta}_i| \leq 2\sqrt{\lambda_0\lambda_2} + \sqrt{\lambda_0/\lambda_2} \\ T(\tilde{\beta}_i(1+2\lambda_2)^{-1}; 0, M) & \text{otherwise} \end{cases} \tag{3.14}$$

and for $\sqrt{\lambda_0/\lambda_2} > M$, the solution of (3.13) is:

$$\hat{\beta}_i = T(\tilde{\beta}_i; \lambda_0/M + \lambda_2 M, M). \tag{3.15}$$

Thus, update (3.12) in Algorithm 3.1 can be computed in closed-form using expressions (3.14) and (3.15). See Appendix 3.B.1 for a derivation of the update rules.

Warm Starts: Interior-point methods used in state-of-the-art nonlinear BnB solvers cannot easily use warm starts. On the other hand, cyclic CD has proven to be very effective in exploiting warm starts, e.g., see [69, 86]. For every node in BnB, we use its parent's primal solution as a warm start. Recall that the parent and children nodes have the same relaxations, except that a single binary variable (used for branching) is fixed to zero or one in the children[4]. Intuitively, this can make the warm start close to the optimal solution. Moreover, the supports of the optimal solutions of the parent and children nodes are typically very close. We exploit this observation by sharing information about the supports across the BnB tree, as we describe next in Section 3.3.1.

Active Sets

Update (3.12) in Algorithm 3.1 requires $\mathcal{O}(n)$ operations, so every full cycle (across all p coordinates) of the algorithm has cost of $\mathcal{O}(np)$. This becomes a major bottleneck for large n or p, since CD can require many cycles before convergence. In practice, the majority of the variables stay zero during the course of Algorithm 3.1 (assuming that the regularization parameters are chosen so that the optimal solution of the relaxation is sparse, and good

[4]Suppose the parent node branches on z_i. After reformulating the node relaxations in the β space (as discussed in Appendix 3.B.2), the relaxation of the child with $z_i = 1$ will be the same as the parent except that the penalty of coordinate i in the parent, $\psi(\beta_i; \lambda_0, \lambda_2, M)$, will be replaced with $\lambda_0 + \lambda_2\beta_i^2$. For the child with $z_i = 0$, the relaxation will be similar to the parent but with β_i fixed to 0.

warm starts are used). Motivated by this observation, we run Algorithm 3.1 restricted to a small subset of the variables $\mathcal{A} \subseteq [p]$, which we refer to as the *active set*, i.e., we solve the following restricted problem:

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, F(\beta) \quad \text{s.t.} \quad \|\beta\|_\infty \leq M, \;\; \beta_{\mathcal{A}^c} = 0. \tag{3.16}$$

After solving (3.16), we augment the active set with any variable $i \in \text{Supp}(\hat{\beta})^c$ that violates the following optimality condition:

$$0 = \underset{\beta_i \in \mathbb{R}}{\arg\min}\, F(\hat{\beta}_1, \ldots, \beta_i, \ldots, \hat{\beta}_p) \quad \text{s.t.} \quad |\beta_i| \leq M.$$

We repeat this procedure of solving the restricted problem (3.16) and then augmenting $\mathcal{A}$ with violating variables, until there are no more violations. The algorithm is summarized below.

Algorithm 3.2: The Active-set Algorithm[a]

- **Input:** Initial solution $\hat{\beta}$ and initial active set $\mathcal{A}$.
- **Repeat**:
  - Step 1: Solve (3.16) using Algorithm 3.1 to get a solution $\hat{\beta}$.
  - Step 2: $\mathcal{V} \leftarrow \{i \in \text{Supp}(\hat{\beta})^c \mid 0 \neq \arg\min_{|\beta_i| \leq M} F(\hat{\beta}_1, \ldots, \beta_i, \ldots, \hat{\beta}_p)\}$.
  - Step 3: If $\mathcal{V}$ is empty **terminate**, otherwise, $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{V}$.

[a] Algorithm 3.2 solves the root relaxation. For other nodes, the objective function $F(\beta)$ will need to be modified to account for the fixed variables (as detailed in Appendix 3.B.2), and all the variables fixed to zero at the current node should be excluded from the set $\mathcal{V}$ in Step 2.

The quality of the initial active set affects the number of iterations in Algorithm 3.2. For example, if $\mathcal{A}$ is a superset of the support of an optimal solution to (3.5), then $\mathcal{V}$ in the first iteration of Algorithm 3.2 will be empty, and the algorithm will terminate in a single iteration. For every node in BnB, we choose the initial active set to be the same as the final active set of the parent node. This works well in practice because parent and children nodes typically have similar supports. For the root relaxation, we obtain the initial active set by choosing a small subset of features which have the highest (absolute) correlation with

y.

In Section 3.3.2, we present a novel method that makes use of the updates in Algorithm 3.2 to obtain provably high-quality dual bounds. We note that our approach goes beyond standard active-set methods [69]. In particular, (i) we use active sets in the context of a BnB tree, percolating information from parents to children (as opposed to warm-start continuation across a grid of regularization parameters); and (ii) we exploit the active sets to deliver dual bounds. In the next remark, we discuss how Step 1 in Algorithm 3.2 can be performed inexactly.

Remark 3.1. *In practice, we solve the restricted optimization problem in Step 1 of Algorithm 3.2 inexactly. Specifically, in Step 1, we terminate Algorithm 3.1 when the relative change in the objective values is below a small numerical tolerance*[5]*. For Step 3, we ensure that $\mathcal{V}$ is (exactly) empty before terminating the algorithm, which guarantees that there are no optimality violations outside the support. Having no optimality violations outside the support will be essential to obtain the tight dual bounds discussed in Section 3.3.2.*

Gradient Screening

Algorithm 3.2 effectively reduces the number of coordinate updates, through restricting optimization to the active set. However, checking the optimality conditions outside the active set, i.e., performing Step 2, requires $\mathcal{O}(np)$ operations. This is a bottleneck when p is large, even if a small number of such checks (or passes) are performed. To mitigate this, we present a *gradient screening* method which reduces the time complexity of these optimality checks. Our method is inspired by the gradient screening technique proposed in [87] for a different problem: learning sparse hierarchical interactions via a convex optimization formulation. In this chapter, the optimality checks in Step 2 of Algorithm 3.2 essentially require computing a gradient of the least squares loss, in order to construct $\mathcal{V}$. Loosely speaking, gradient screening is designed to avoid computing the "non-essential" parts of this gradient by using previously computed quantities in the BnB tree.

[5] If upon termination, an integral z is obtained, we solve the relaxation to optimality, as discussed in the introduction of Section 3.3.

In the following proposition, we give an explicit way to construct the set $\mathcal{V}$ in Step 2 of Algorithm 3.2.

Proposition 3.3. *Let $\hat{\beta}$ and $\mathcal{V}$ be as defined in Algorithm 3.2, and define $\hat{r} = y - X\hat{\beta}$. Then, the set $\mathcal{V}$ can be equivalently written as follows:*

$$\mathcal{V} = \{i \in \mathit{Supp}(\hat{\beta})^c \mid |\langle \hat{r}, X_i \rangle| > c(\lambda_0, \lambda_2, M)\}, \tag{3.17}$$

where X_i denotes the i-th column of X; and $c(\lambda_0, \lambda_2, M) = 2\sqrt{\lambda_0 \lambda_2}$ if $\sqrt{\lambda_0/\lambda_2} \leq M$, and $c(\lambda_0, \lambda_2, M) = (\lambda_0/M + \lambda_2 M)$ if $\sqrt{\lambda_0/\lambda_2} > M$.

By Proposition 3.3, constructing $\mathcal{V}$ directly costs $\mathcal{O}(n(p - \|\hat{\beta}\|_0))$. Next, we will discuss how to compute $\mathcal{V}$ with a lower cost, by making use of previously computed quantities. Suppose we have access to a set $\hat{\mathcal{V}} \subseteq [p]$ such that $\mathcal{V} \subseteq \hat{\mathcal{V}}$. Then, we can construct $\mathcal{V}$ by restricting the checks in (3.17) to $\hat{\mathcal{V}}$ instead of $\mathrm{Supp}(\hat{\beta})^c$, i.e., the following holds:

$$\mathcal{V} = \{i \in \hat{\mathcal{V}} \mid |\langle \hat{r}, X_i \rangle| > c(\lambda_0, \lambda_2, M)\}. \tag{3.18}$$

Assuming $\hat{\mathcal{V}}$ is available, the cost of computing $\mathcal{V}$ in (3.18) is $\mathcal{O}(n|\hat{\mathcal{V}}|)$. This cost can be significantly smaller than that of (3.17), if $|\hat{\mathcal{V}}|$ is sufficiently small. Next, we present a method that can obtain a relatively small $\hat{\mathcal{V}}$ in practice, thereby speeding up the computation of $\mathcal{V}$.

Computation of $\hat{\mathcal{V}}$: Proposition 3.4 presents a method to construct a set $\hat{\mathcal{V}}$ which satisfies $\mathcal{V} \subseteq \hat{\mathcal{V}}$ (as discussed above), using information from a warm start β^0 (e.g., the solution of the relaxation from the parent node in BnB).

Proposition 3.4. *Let $\hat{\beta}$ and $\mathcal{V}$ be as defined in Algorithm 3.2. Let β^0 be an arbitrary vector in $\mathbb{R}^p$. Define $\hat{r} = y - X\hat{\beta}$, $r^0 = y - X\beta^0$, and $\epsilon = \|X\beta^0 - X\hat{\beta}\|_2$. Then, the following holds:*

$$\mathcal{V} \subseteq \hat{\mathcal{V}} := \Big\{i \in \mathit{Supp}(\hat{\beta})^c \mid |\langle r^0, X_i \rangle| > c(\lambda_0, \lambda_2, M) - \epsilon\Big\}. \tag{3.19}$$

The set $\hat{\mathcal{V}}$ defined in (3.19) depends solely on ϵ and the quantities $|\langle r^0, X_i\rangle|$, $i \in \text{Supp}(\hat{\beta})^c$, which we will assume to be sorted and available along with the warm start β^0. This is in contrast to a direct evaluation of $\mathcal{V}$ in (3.17), which requires the costly computation of the terms $|\langle \hat{r}, X_i\rangle|$, $i \in \text{Supp}(\hat{\beta})^c$. Given the sorted values $|\langle r^0, X_i\rangle|$, $i \in \text{Supp}(\hat{\beta})^c$, we can identify $\hat{\mathcal{V}}$ in (3.19) with $\mathcal{O}(\log p)$ operations, using a variant of binary search. Thus, the overall cost of computing $\mathcal{V}$ using (3.18) is $\mathcal{O}(n|\hat{\mathcal{V}}| + \log p)$. We also note that in practice the inclusion $\mathcal{V} \subseteq \hat{\mathcal{V}}$ in (3.19) is strict.

Proposition 3.4 tells us that the closer $X\beta^0$ is to $X\hat{\beta}$, the smaller $\hat{\mathcal{V}}$ is, i.e., the lower is the cost of computing $\mathcal{V}$ in (3.18). Thus, for the method to be successful, we need a good warm start β^0. In practice, we obtain β^0 for the current node in BnB from its parent, and we update β^0 as necessary during the course of Algorithm 3.2 (as detailed below). Let $\epsilon_{\text{gs}} \in (0, 1)$ be a pre-specified parameter for the gradient screening procedure. The procedure, which replaces Step 2 of Algorithm 3.2, is defined as follows:

Gradient Screening

1. If this is the root node and the first iteration of Algorithm 3.2, then: update $\beta^0 \leftarrow \hat{\beta}$, $r^0 = y - X\beta^0$; compute and sort the terms $|\langle r^0, X_i\rangle|$, $i \in [p]$; compute $\mathcal{V}$ using (3.17); and skip all the steps below.

2. If this is the first iteration of Algorithm 3.2, get β^0 from the parent node in the BnB tree.

3. Compute $\hat{\mathcal{V}}$ using (3.19) and then $\mathcal{V}$ using (3.18)[6].

4. If $|\hat{\mathcal{V}}| > \epsilon_{\text{gs}} p$, update $\beta^0 \leftarrow \hat{\beta}$, $r^0 = y - X\beta^0$; re-compute and sort the terms $|\langle r^0, X_i\rangle|$, $i \in [p]$.

If $X\beta^0$ is not a good estimate of $X\hat{\beta}$, then $\hat{\mathcal{V}}$ might become large. To avoid this issue, we update β^0 in Step 4 above, every time the set $\hat{\mathcal{V}}$ becomes relatively large ($|\hat{\mathcal{V}}| > \epsilon_{\text{gs}} p$). The parameter ϵ_{gs} controls how often β^0 is updated. In our implementation, we use $\epsilon_{\text{gs}} = 0.05$

[6]Each variable i fixed to zero by BnB at the current node should be excluded when computing $\mathcal{V}$ and $\hat{\mathcal{V}}$.

by default, but we note that the parameter can be generally tuned in order to improve the running time. The updated β^0 will be passed to the children in the BnB tree. While Step 4 above can be costly, it is not performed often in practice as the solutions of the relaxations in the parent and children nodes are typically close.

Based on our current implementation and experiments, we see notable benefits from gradient screening when $p \geq 10^4$ (e.g., more than 2x speedup for $p = 10^6$). For smaller values of p, we typically observe a small additional overhead from gradient screening. Moreover, we note that gradient screening will increase memory consumption, because each of the open nodes in the tree will need to store a p-dimensional vector (consisting of the quantities $|\langle r^0, X_i \rangle|$, $i \in [p]$, which are maintained by gradient screening).

3.3.2 Dual Bounds

In practice, we use Algorithm 3.2 to obtain inexact primal solutions for relaxation (3.5), as discussed in Remark 3.1. However, dual bounds are needed to perform search space pruning in BnB. Here, we present a new efficient method to obtain dual bounds from the primal solutions. Moreover, we prove that our method can obtain dual bounds whose tightness depends on the sparsity level of the relaxation rather than p. We start by introducing a Lagrangian dual of relaxation (3.5) in Theorem 3.2.

Theorem 3.2. *For $\sqrt{\lambda_0/\lambda_2} \leq M$, a dual of Problem (3.5) is given by:*

$$\max_{\alpha \in \mathbb{R}^n, \gamma \in \mathbb{R}^p} \; h_1(\alpha, \gamma) := -\frac{1}{2}\|\alpha\|_2^2 - \alpha^T y - \sum_{i \in [p]} v(\alpha, \gamma_i), \tag{3.20}$$

where $v : \mathbb{R}^{n+1} \to \mathbb{R}$ is defined as follows:

$$v(\alpha, \gamma_i) := \Big[\frac{(\alpha^T X_i - \gamma_i)^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i|. \tag{3.21}$$

Otherwise, if $\sqrt{\lambda_0/\lambda_2} > M$*, a dual of Problem* (3.5) *is given by*

$$\begin{aligned} \max_{\rho\in\mathbb{R}^n,\mu\in\mathbb{R}^p} \; & h_2(\rho,\mu) := -\frac{1}{2}\|\rho\|_2^2 - \rho^T y - M\|\mu\|_1 \\ s.t. \quad & |\rho^T X_i| - \mu_i \le \lambda_0/M + \lambda_2 M, \; i \in [p]. \end{aligned} \tag{3.22}$$

Let β^* *be an optimal solution to Problem* (3.5) *and define* $r^* = y - X\beta^*$*. The optimal dual variables for* (3.20) *are given by:*

$$\alpha^* = -r^* \;\; and \;\; \gamma_i^* = \mathbb{1}_{[|\beta_i^*|=M]}(\alpha^{*T} X_i - 2M\lambda_2 \operatorname{sign}(\alpha^{*T} X_i)), i \in [p]. \tag{3.23}$$

Moreover, the optimal dual variables for (3.22) *are:*

$$\rho^* = -r^* \quad and \quad \mu_i^* = \mathbb{1}_{[|\beta_i^*|=M]}(|\rho^{*T} X_i| - \lambda_0/M - \lambda_2 M), i \in [p]. \tag{3.24}$$

Note that strong duality holds for relaxation (3.5) since it satisfies Slater's condition [22], so the optimal objective of the dual in Theorem 3.2 matches that of (3.5).

Dual Feasible Solutions: Let $\hat{\beta}$ be an inexact primal solution obtained using Algorithm 3.2 and define $\hat{r} = y - X\hat{\beta}$. We discuss next how to construct a dual feasible solution using $\hat{\beta}$, i.e., without solving the dual in Theorem 3.2.

Case of $\sqrt{\lambda_0/\lambda_2} \le M$**:** Here, we construct a dual solution $(\hat{\alpha}, \hat{\gamma})$ for Problem (3.20) as follows:

$$\hat{\alpha} = -\hat{r} \;\; \text{and} \;\; \hat{\gamma} \in \arg\max_{\gamma\in\mathbb{R}^p} h_1(\hat{\alpha}, \gamma). \tag{3.25}$$

The choice $\hat{\alpha} = -\hat{r}$ is motivated by the optimality conditions in Theorem 3.2, while $\hat{\gamma}$ maximizes the dual objective (with α fixed to $\hat{\alpha}$). Note that the constructed solution is (trivially) feasible since the dual in (3.20) is unconstrained. Since (3.20) is separable across the γ_i's, $\hat{\gamma}_i$ is equivalently the solution of $\min_{\gamma_i\in\mathbb{R}} v(\hat{\alpha}, \gamma_i)$, whose corresponding solution is

given by:

$$\hat{\gamma}_i = T(\hat{\alpha}^T X_i; 2M\lambda_2, \infty). \tag{3.26}$$

Thus, $\hat{\gamma}$ can be obtained in closed-form using (3.26). Since $\hat{\beta}$ is the output of Algorithm 3.2, we know that the corresponding set $\mathcal{V}$ in Algorithm 3.2 is empty. Thus, by Proposition 3.3, we have $|\hat{r}^T X_i| \leq 2\sqrt{\lambda_0\lambda_2}$ for any $i \in \text{Supp}(\hat{\beta})^c$. Using $\hat{r} = -\hat{\alpha}$ and $\sqrt{\lambda_0/\lambda_2} \leq M$ in the previous in inequality, we get $|\hat{\alpha}^T X_i| \leq 2M\lambda_2$. Using the latter bound in (3.26), we get that $\hat{\gamma}_i = 0$ for any $i \in \text{Supp}(\hat{\beta})^c$. However, for $i \in \text{Supp}(\hat{\beta})$, $\hat{\gamma}_i$ can be potentially nonzero.

Case of $\sqrt{\lambda_0/\lambda_2} > M$: We construct a dual feasible solution $(\hat{\rho}, \hat{\mu})$ for (3.22) as follows:

$$\hat{\rho} = -\hat{r} \;\text{ and }\; \hat{\mu} \in \underset{\mu \in \mathbb{R}^p}{\arg\max}\; h_2(\hat{\rho}, \mu) \quad \text{s.t.} \quad (\hat{\rho}, \mu) \text{ is feasible for (3.22).} \tag{3.27}$$

Similar to (3.27), the choice $\hat{\rho} = -\hat{r}$ is motivated by the optimality conditions in Theorem 3.2, whereas the choice $\hat{\mu}$ maximizes the dual objective under the condition that $\rho = -\hat{r}$ (while ensuring feasibility). It can be readily seen that $\hat{\mu}_i$ is given in closed form by:

$$\hat{\mu}_i = \Big[|\hat{\rho}^T X_i| - \lambda_0/M - \lambda_2 M\Big]_+. \tag{3.28}$$

Note that for any $i \in \text{Supp}(\hat{\beta})^c$, we have $|\hat{\rho}^T X_i| = |\hat{r}^T X_i| \leq \lambda_0/M + \lambda_2 M$ (by Proposition 3.3), which implies that $\hat{\mu}_i = 0$. However, for $i \in \text{Supp}(\hat{\beta})$, $\hat{\mu}_i$ can be potentially nonzero.

Quality of the Dual Bounds: In Theorem 3.3, we quantify the tightness of the dual bounds obtained from the dual feasible solutions (3.25) and (3.27).

Theorem 3.3. *Let α^*, γ^*, ρ^*, and μ^* be the optimal dual variables defined in Theorem 3.2. Let β^* be an optimal solution to* (3.5)*, and $\hat{\beta}$ be an inexact solution obtained using Algorithm 3.2. Define the primal gap $\epsilon = \|X(\beta^* - \hat{\beta})\|_2$. Let $k = \|\hat{\beta}\|_0$ denote the number of nonzeros in the inexact solution to* (3.5)*. For a fixed (M, λ_2), the following holds for the dual solution*

$(\hat{\alpha}, \hat{\gamma})$ *defined in* (3.25)*:*

$$h_1(\hat{\alpha}, \hat{\gamma}) \geq h_1(\alpha^*, \gamma^*) - k\mathcal{O}(\epsilon) - k\mathcal{O}(\epsilon^2), \tag{3.29}$$

and for the dual solution $(\hat{\rho}, \hat{\mu})$ *defined in* (3.27)*, we have:*

$$h_2(\hat{\rho}, \hat{\mu}) \geq h_2(\rho^*, \mu^*) - k\mathcal{O}(\epsilon) - \mathcal{O}(\epsilon^2). \tag{3.30}$$

Interestingly, the bounds established in Theorem 3.3 do not depend on p, but rather on the support size k. Specifically, the constants in $\mathcal{O}(\epsilon)$ and $\mathcal{O}(\epsilon^2)$ only involve M and λ_2 (these constants are made explicit in the proof). In practice, we seek highly sparse solutions, i.e., $k \ll p$—suggesting that the quality of the dual bounds deteriorates with k and not p. The main driver behind these tight bounds is Algorithm 3.2, which performs optimality checks on the coordinates outside the support. If vanilla CD was used instead of Algorithm 3.2, then the term k appearing in the bounds (3.29) and (3.30) will be replaced by p, making the bounds loose[7].

Efficient Computation of the Dual Bounds: A direct computation of the dual bound $h_1(\hat{\alpha}, \hat{\gamma})$ or $h_2(\hat{\rho}, \hat{\mu})$ costs $\mathcal{O}(np)$ operations. Interestingly, we show that this cost can be reduced to $\mathcal{O}(n + n\|\hat{\beta}\|_0)$ (where we recall that $\hat{\beta}$ is a solution from Algorithm 3.2). First, we consider the case of $\sqrt{\lambda_0/\lambda_2} \leq M$, where the goal is to compute $h_1(\hat{\alpha}, \hat{\gamma})$. By Lemma 3.2 (in the appendix), we have $v(\hat{\alpha}, \hat{\gamma}_i) = 0$ for every $i \in \text{Supp}(\hat{\beta})^c$. Thus, $h_1(\hat{\alpha}, \hat{\gamma})$ can be simplified to:

$$h_1(\hat{\alpha}, \hat{\gamma}) = -\frac{1}{2}\|\hat{\alpha}\|_2^2 - \hat{\alpha}^T y - \sum_{i \in \text{Supp}(\hat{\beta})} v(\hat{\alpha}, \hat{\gamma}_i). \tag{3.31}$$

Now, we consider the case of $\sqrt{\lambda_0/\lambda_2} > M$, where the goal is to evaluate $h_2(\hat{\rho}, \hat{\mu})$. By construction, the solution (3.27) is dual feasible, which means that the constraints in (3.22) need not be checked when computing the bound. Moreover, $\hat{\mu}_i = 0$ for every $i \in \text{Supp}(\hat{\beta})^c$

[7]This holds by using the argument in the proof of Theorem 3.3 for Algorithm 3.1 instead of Algorithm 3.2.

(see the discussion after (3.28)). Thus, the dual bound can be expressed as follows:

$$h_2(\hat{\rho}, \hat{\mu}) = -\frac{1}{2}\|\hat{\rho}\|_2^2 - \hat{\rho}^T y - M \sum_{i \in \text{Supp}(\hat{\beta})} |\hat{\mu}_i|. \tag{3.32}$$

The expressions in (3.31) and (3.32) can be computed in $\mathcal{O}(n + n\|\hat{\beta}\|_0)$ operations.

3.3.3 Branching and Incumbents

Many branching strategies for BnB have been explored in the literature, e.g., random branching, strong branching, and pseudo-cost branching [20, 37, 6, 2]. Among these strategies, strong branching has proven to be very effective in minimizing the size of the search tree [6, 37, 20]. Strong branching selects the variable which leads to the maximum increase in the lower bounds of the children nodes. To select such a variable, two temporary node subproblems should be solved for every non-integral variable in the current relaxation. This can become a computational bottleneck, as each temporary subproblem involves solving a nonlinear optimization problem similar to (3.5). To address this challenge, we use a fast (approximate) version of strong branching, in which we restrict the optimization in these temporary subproblems to the active set of the current node (instead of optimizing over all p variables). In practice, this often leads to very similar search trees compared to exact strong branching, since the active set of the parent is typically close to the support of the children.

We obtain the initial upper bound using `L0Learn` [86], which uses CD and efficient local search algorithms to obtain good quality feasible solutions for Problem (3.3). Moreover, at every node in the BnB tree, we attempt to improve the incumbent by making use of the support of a solution to the node relaxation. Specifically, we solve the following ℓ_2 regularized least squares problem restricted to the relaxation's support S: $\min_{\beta_S \in \mathbb{R}^{|S|}} \frac{1}{2}\|y - X_S\beta_S\|_2^2 + \lambda_2\|\beta_S\|_2^2$. Since S is typically small, this problem can be solved efficiently by inverting a small $|S| \times |S|$ matrix. If S is similar to the support of the parent node, then a solution for the current node can be computed via a low-rank update [78].

3.4 Experiments

We perform a series of high-dimensional experiments to study the run time of our BnB, understand its sensitivity to parameter and algorithm choices, and compare to state-of-the-art approaches. While our dataset and parameter choices are motivated by statistical considerations, our focus here is not to study the statistical properties of ℓ_0 estimators. We refer the reader to [86] for empirical studies on statistical properties.

3.4.1 Experimental Setup

Synthetic Data Generation: We generate a multivariate Gaussian data matrix with samples drawn from $\text{MVN}(0, \Sigma_{p\times p})$, a sparse coefficient vector $\beta^\dagger \in \mathbb{R}^p$ with $k^\dagger$ equi-spaced nonzero entries all set to 1, and a noise vector $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. We denote the support of the true regression coefficients $\beta^\dagger$ by $S^\dagger$. The response is then obtained from the linear model $y = X\beta^\dagger + \epsilon$. We define the *signal-to-noise ratio (SNR)* as follows: $\text{SNR} = \text{Var}(X\beta^\dagger)/\sigma^2$. Unless otherwise specified, we set σ^2 to achieve $\text{SNR} = 5$—this is a relatively difficult setting which still allows for full support recovery, under suitable choices of n, p, and Σ (see [86, 116] for a discussion on appropriate levels of SNR). We perform mean-centering followed by normalization (to have unit ℓ_2 norm) on y and each column of X. To help in exposition, we denote the resulting processed dataset by $(\tilde{y}, \tilde{X})$ and the (scaled) regression coefficients by $\tilde{\beta}^\dagger$.

Warm Starts, λ_0, λ_2, and M: The parameters λ_2 and M can affect the run time significantly, so we study the sensitivity to these choices in our experiments. We consider choices that are relevant from a statistical perspective, as we discuss next. For a fixed λ_2, let $\beta(\lambda_2)$ be the solution of ridge regression restricted to the support of the true solution $\tilde{\beta}^\dagger$:

$$\beta(\lambda_2) \in \underset{\beta\in\mathbb{R}^p}{\arg\min} \frac{1}{2}\|\tilde{y} - \tilde{X}\beta\|_2^2 + \lambda_2\|\beta\|_2^2 \quad \text{s.t.} \quad \beta_{(S^\dagger)^c} = 0, \tag{3.33}$$

We define λ_2^* as the λ_2 which minimizes the ℓ_2 estimation error of $\beta(\lambda_2)$, i.e.,

$$\lambda_2^* \in \arg\min_{\lambda_2 \geq 0} \|\tilde{\beta}^\dagger - \beta(\lambda_2)\|_2. \tag{3.34}$$

We estimate λ_2^* using grid search, with 50 points equi-spaced on a logarithmic scale in the range $[10^{-4}, 10^4]$. In the experiments, we report our λ_2 choices as a fraction or multiple of λ_2^* (e.g., $\lambda_2 = 0.1\lambda_2^*$). Moreover, for each λ_2, we define $M^*(\lambda_2) = \|\beta(\lambda_2)\|_\infty$ and report our choices of the Big-M in terms of $M^*(\lambda_2)$—we also use the notation M^* to refer to $M^*(\lambda_2)$ when λ_2 is clear from context. Note that if Problem (3.1) has a unique solution, and this solution has support $S^\dagger$, then $M^*(\lambda_2)$ is the smallest valid choice of M in formulation (3.3). Unless otherwise specified, for each λ_2 considered, we fix λ_0 to a value λ_0^* (which is a function of λ_2) that leads to the true support size $k^\dagger$. We obtain λ_0^* using `L0Learn`[8] [86]. Note that λ_0^* might not exist in general, but in all the experiments we considered, `L0Learn` was able to such a value. Moreover, Section 3.4.2, presents an experiment where λ_0 is varied over a range that is independent of `L0Learn`.
We obtain warm starts from `L0Learn`[9] and use them for all the MIP solvers considered.

Solvers and Settings: Our solver, L0BnB[10], is written in Python with critical code sections optimized using Numba [104]. We compare L0BnB with Gurobi (GRB), MOSEK (MSK), and BARON (B) on formulation (3.3). We also compare with [28] who solve the cardinality-constrained variant of (3.1) with the number of nonzeros set to $k^\dagger$. In all solvers (including L0BnB), we use the following settings:

- Relative Optimality Gap: Given an upper bound UB and a lower bound LB, the relative optimality gap is defined as $(\text{UB} - \text{LB})/\text{UB}$. We set this to 1%.

- Integer Feasibility Tolerance (ϵ_{if}): This tolerance is used to determine whether a variable obtained from a node relaxation will be declared as integral (for the purpose of

[8] `L0Learn` computes a data-dependent grid of λ_0 values. When CD is used to optimize (3.1), each λ_0 in the grid leads to a different support size. See Section 4.1 of [86] for more details.

[9] We use the default CD-based algorithm in `L0Learn`, which does not depend on any Big-M parameter. We remind the reader that `L0Learn` is a local optimization method that does not provide certificates of global optimality.

[10] `https://github.com/alisaab/l0bnb`

branching). Specifically, in our context, if a variable z_i is within ϵ_{if} from 0 or 1, then it is declared as integral. We set $\epsilon_{\text{if}} = 10^{-4}$.

- Primal-Dual Optimality Gap (ϵ_{pd}): This is the relative gap between the primal and dual bounds at a given node, which is used as termination criterion for the subproblem solver. We set $\epsilon_{\text{pd}} = 10^{-5}$. In L0BnB, this gap is satisfied when an integral solution to a node's relaxation is encountered.

Gradient Screening in L0BnB: As discussed in Section 3.3.1, our gradient screening implementation leads to noticeable speed-ups for large p ($\geq 10^4$ in our experience). For smaller values of p, we do not observe run time improvements. In the experiment of Section 3.4.2 (Table 3.6), where we vary p, we report the running time with and without gradient screening. In the other experiments, we do not use gradient screening.

Reproducibility and Additional Details: In the spirit of reproducibility, the function used to generate and process the synthetic datasets is publicly available on L0BnB's Github page. In all experiments, we report the values of M, λ_0, and λ_2 (either in the main text or in Appendix 3.C). Moreover, for all approaches, we report the number of BnB nodes explored in Appendix 3.C.

3.4.2 Comparison with State-of-the-art Solvers

Varying Number of Features

In this section, we study the scalability of the different solvers in terms of the number of features p. We generate synthetic datasets[11] with $n = 10^3$, $p \in \{10^3, 10^4, 10^5, 10^6\}$, and $k^\dagger = 10$. We consider a constant correlation setting, where $\Sigma_{ij} = 0.1\ \forall i \neq j$ and 1 otherwise. The parameters λ_2 and M can have a significant effect on the run time. Thus, we report the timings for different choices of these parameters. In particular, in Table 3.1 (top panel), we report the timings for $\lambda_2 \in \{0.1\lambda_2^*, \lambda_2^*, 10\lambda_2^*\}$, where for each λ_2, we fix $M = 1.5M^*(\lambda_2)$

[11]To ensure that the same estimates of λ_2^* and M^* are used for the different choices of p, we generate a single data matrix with 10^6 features. For each p, we take a submatrix that consists of all the rows and a subset of p columns (which includes the true support).

(where λ_2^* and M^* are defined in Section 3.4.1). In Table 3.1 (bottom panel), we fix $\lambda_2 = \lambda_2^*$ and report the timings for $M \in \{M^*(\lambda_2), 2M^*(\lambda_2), 4M^*(\lambda_2), \infty\}$. Note that the results for L0BnB in Table 3.1 are without gradient screening; in Table 3.6 in Appendix 3.C, we report the timings with gradient screening enabled. Moreover, in Appendix 3.C, we report the values of λ_0^* and λ_2^*, along with the number of BnB nodes explored.

Table 3.1: (Sensitivity to λ_2 and M) Running time in seconds for solving (3.1) (via formulation (3.3)) by our proposal (L0BnB), Gurobi (GRB), MOSEK (MSK), and BARON (B). The method [28] solves the cardinality-constrained variant of (3.1). For methods that do not terminate in 4 hours: the optimality gap is shown in parenthesis, and a dash (-) is used in the special case of a 100% gap. [Top panel] For each λ_2, we use $M = 1.5M^*(\lambda_2)$. [Bottom panel] We use $\lambda_2 = \lambda_2^*$, and four different values of M. The Big-M values are based on: $M^*(\lambda_2^*) = 0.232$, $M^*(0.1\lambda_2^*) = 0.164$, and $M^*(10\lambda_2^*) = 0.243$. The true solution satisfies: $\|\tilde{\beta}^\dagger\|_\infty = 0.208$. In the bottom panel, we found that PR(M^*) and PR(∞) have the same optimal solution.

	$\lambda_2 = \lambda_2^*$					$\lambda_2 = 0.1\lambda_2^*$					$\lambda_2 = 10\lambda_2^*$				
p	L0BnB	GRB	MSK	B	[28]	L0BnB	GRB	MSK	B	[28]	L0BnB	GRB	MSK	B	[28]
10^3	**0.7**	70	92	(4%)	(34%)	**2**	57	154	-	-	**0.01**	148	28	(5%)	0.08
10^4	**3**	(15%)	1697	-	(78%)	**12**	(12%)	3872	-	-	**0.06**	(11%)	314	-	5
10^5	**34**	-	-	-	(86%)	**545**	-	-	-	-	**0.5**	-	-	-	8
10^6	**1112**	-	-	-	-	**(23%)**	-	-	-	-	**8**	-	-	-	46

	$M = M^*$				$M = 2M^*$				$M = 4M^*$				$M = \infty$			
p	L0BnB	GRB	MSK	B	L0BnB	GRB	MSK	B	L0BnB	GRB	MSK	B	L0BnB	GRB	MSK	B
10^3	**0.2**	27	57	(2%)	**0.8**	112	128	-	**0.8**	1219	137	-	**0.8**	3974	91	-
10^4	**0.9**	10571	665	-	**5**	(24%)	3260	-	**5**	(50%)	3289	-	**6**	-	9025	-
10^5	**8**	-	-	-	**70**	-	-	-	**81**	-	-	-	**81**	-	-	-
10^6	**121**	-	-	-	**9265**	-	-	-	**10986**	-	-	-	**11010**	-	-	-

The results in the top panel of Table 3.1 indicate significant speed-ups, reaching over $200,000$x compared to Gurobi, $28,000$x compared to MOSEK, and $20,000$x compared to [28]. At $\lambda_2 = \lambda_2^*$, L0BnB is the only solver that can handle $p \geq 10^5$, and can, in fact, handle $p = 10^6$ in less than 20 minutes. Recall that λ_2^* minimizes the ℓ_2 estimation error and leads to an estimator that is the closest to the ground truth. When the ridge parameter is weak i.e., for $\lambda_2 = 0.1\lambda_2^*$, L0BnB is again the only solver that can handle $p \geq 10^5$. When the ridge parameter is large, i.e., for $\lambda_2 = 10\lambda_2^*$, the optimization problem seems to become easier: L0BnB can be more than 100x faster compared to λ_2^*, and [28] can handle up to $p \approx 10^6$. The speed-ups for $\lambda_2 = 10\lambda_2^*$ can be attributed to the fact that a larger λ_2 adds a large amount of

regularization to the objective (via the perspective term)—improving the performance of the relaxation solvers[12]. It is also worth emphasizing that L0BnB is prototyped in Python, as opposed to the highly efficient BnB routines available in commercial solvers such as Gurobi and MOSEK.

Ideally, we desire a solver that can solve Problem (3.1) over a range of λ_2 values, which includes values in the neighborhood of λ_2^*. However, the results in Table 3.1 suggest that the state-of-the-art methods (except L0BnB) seem to only work for quite large values of λ_2 (which, in this case, do not correspond to solutions that are interesting from a statistical viewpoint). On the other hand, L0BnB seems to be the only method that can scale to $p \sim 10^6$ while being relatively robust to the choice of λ_2.

In the bottom panel of Table 3.1, the results also indicate that L0BnB significantly outperforms Gurobi, MOSEK, and BARON for different choices of M. For all the solvers, the run time increases with M, and the longest run times are for $M = \infty$. This empirically validates our result in Proposition 3.1, where we show that for a sufficiently small (but valid) value of M, PR(M) can be better than PR(∞), in terms of the relaxation quality. However, even with $M = \infty$, L0BnB can solve $p = 10^6$ in around 3 hours, whereas all other solvers have a 100% gap after 4 hours. We also note that Table 3.1 considers both the two settings in Theorem 3.1: $\sqrt{\lambda_0/\lambda_2} > M$ and $\sqrt{\lambda_0/\lambda_2} \leq M$. Specifically, we have $\sqrt{\lambda_0^*/\lambda_2^*} > M$ for $M \in \{M^*(\lambda_2^*), 1.5M^*(\lambda_2^*), 2M^*(\lambda_2^*)\}$, and $\sqrt{\lambda_0^*/\lambda_2^*} \leq M$ for $M \in \{4M^*(\lambda_2^*), \infty\}$. L0BnB achieves notable improvements over other solvers for both of these settings.

Varying Signal-to-Noise Ratio (SNR)

Here we investigate the effect of SNR on the running time of the different solvers. To this end, we generate synthetic datasets with $n = 10^3$, $p = 10^3$, $k^\dagger = 10$, under a constant correlation setting, where $\Sigma_{ij} = 0.1\ \forall i \neq j$ and 1 otherwise. We vary SNR in $\{0.5, 1, 2, 3, 4, 5\}$. We set $\lambda_0 = \lambda_0^*$, $\lambda_2 = \lambda_2^*$ (where λ_0^* and λ_2^* depend on SNR), and $M = 1.5M^*(\lambda_2)$. We report the running time versus SNR in Table 3.2. The corresponding values of λ_0^* and λ_2^*, and

[12]Gurobi is the only exception to this observation. We investigated this: Gurobi generates additional cuts only for the case of $10\lambda_2^*$, which seem to slow down the relaxation solver.

the number of BnB nodes are reported in Appendix 3.C. The results indicate that L0BnB is much faster, and less sensitive to changes in SNR, compared to the other solvers. For example, L0BnB can handle SNR=0.5 in less than 4 minutes, whereas the fastest competing solver (MOSEK) takes around 46 minutes.

Table 3.2: Running time (seconds) for solving (3.3) on a synthetic dataset with $n = p = 10^3$, at different SNR levels. The parameter λ_2 is set to the optimal choice λ_2^*, which depends on the current SNR level. For methods that do not terminate in 2 hours: the optimality gap is shown in parenthesis, and a dash (-) is used in the special case of a 100% gap.

SNR	M	L0BnB	GRB	MSK	B
0.5	0.269	**231**	(2%)	2757	-
1	0.307	**1.7**	1213	1046	-
2	0.333	**1.4**	119	288	-
3	0.346	**1.3**	102	173	-
4	0.347	**1.0**	77	113	-
5	0.348	**0.8**	77	92	-

Varying λ_0 and λ_2

In the experiment of Section 3.4.2, we studied the running time for $\lambda_0 = \lambda_0^*$ so that the model recovers the true support size. In this experiment, we study the running time over a grid of λ_0 and λ_2 values, which includes various support sizes. We consider synthetic instances with $p \in \{10^3, 10^4\}$, $n = 10^3$, $k^\dagger = 10$, under a constant correlation setting, where $\Sigma_{ij} = 0.1\ \forall i \neq j$ and 1 otherwise. We vary $\lambda_2 \in \{0.1, 0.5, 1, 2, 10\} \cdot \lambda_2^*$ and $\lambda_0 \in \{0.5, 0.1, 0.01\} \cdot \lambda_0^{\mathrm{m}}$, where λ_0^{m} is a value of λ_0 which sets all coefficients to zero. Following [86][13], we use $\lambda_0^{\mathrm{m}} = (2 + 4\lambda_2)^{-1}\|X^T y\|_\infty^2$. Next, we describe, how given a pair (λ_0, λ_2) in the grid, we estimate the corresponding Big-M value $M(\lambda_0, \lambda_2)$. Let $\beta(\lambda_0, \lambda_2)$ be the solution of Problem (3.1) restricted to the true support:

$$\beta(\lambda_0, \lambda_2) \in \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0\|\beta\|_0 + \lambda_2\|\beta\|_2^2 \ \text{ s.t. } \ \beta_{(S^\dagger)^c} = 0.$$

We compute $\beta(\lambda_0, \lambda_2)$ exactly using formulation (3.3) with $M = \infty$. We then compute an

[13] [86] shows that λ_0^{m} is the smallest choice of λ_0 for which $\beta = 0$ is a coordinate-wise minimizer for Problem (3.1). In this experiment, we verified numerically that $\beta = 0$ is a global minimizer at λ_0^{m}.

estimate of the Big-M value: $\tilde{M}(\lambda_0, \lambda_2) := \|\beta(\lambda_0, \lambda_2)\|_\infty$. For every (λ_0, λ_2) in the grid, we solve Problem (3.3) for $M(\lambda_0, \lambda_2) = 1.5\tilde{M}(\lambda_0, \lambda_2)$. In Table 3.3, we report the running time for $p = 10^3$ (top panel) and $p = 10^4$ (bottom panel). The values of λ_2^* and λ_0, along with the number of BnB nodes explored, are reported in Appendix 3.C.

In Table 3.3, for all values of λ_2 considered and $\lambda_0 \in \{0.5, 0.1\} \cdot \lambda_0^{\text{m}}$, L0BnB is faster than the competing methods, with speed-ups exceeding 10,000x compared to both Gurobi and MOSEK. However, for $\lambda_0 = 0.01\lambda_0^{\text{m}}$, none of the solvers are able to solve the problem in 1 hour, and the gaps seem to be comparable. We also note that for $p = 10^4$, L0BnB is able to solve (to optimality) 8 out of the 15 instances in the 1 hour limit. In contrast, Gurobi could not solve any of the instances and MOSEK could only solve 5.

3.4.3 Sensitivity to Data Parameters

We study how L0BnB's running time is affected by the following data specific parameters: number of samples (n), feature correlations (Σ), and number of nonzero coefficients ($k^\dagger$). In the experiments below, we fix $\lambda_2 = \lambda_2^*$ and $M = 1.5M^*(\lambda_2^*)$. For all the experiments in this section, the values of λ_0, λ_2, and M used are reported in Appendix 3.C.

Number of Samples: We fix $k^\dagger = 10$, $p = 10^4$, and consider a constant correlation setting $\Sigma_{ij} = 0.1$ for $i \neq j$ and 1 otherwise[14]. The timings for $n \in \{10^2, 10^3, 10^4, 10^5\}$ are in Table 3.4. The results indicate that the problem can be solved in reasonable times (order of seconds to minutes) for $n \geq 10^3$. The problems can be solved the fastest when n is close to p, and the extreme cases $n = 10^2$ and $n = 10^5$ are the slowest. We contend that for large n, the CD updates (which cost $\mathcal{O}(n)$ each) become a bottleneck. For $n = 10^2$, the CD updates are cheaper, however, the size of the search tree is significantly larger—suggesting that a small value of n can lead to loose relaxations and require more branching. Note also that the underlying statistical problem is the most difficult for $n = 10^2$ compared to other larger values of n.

Feature Correlations: We generate synthetic datasets with $p \in \{10^4, 10^5\}$, $n = 10^3$,

[14]We choose $p = 10^4$ to ensure that the matrix fits into memory when n is large.

Table 3.3: Running time for solving (3.3) over a grid of λ_0 and λ_2 values. "NNZ" is the number of nonzeros in the solution to (3.3) (or the best incumbent if all solvers cannot terminate in 1 hour). For methods that do not terminate in 1 hour: the optimality gap is shown in parenthesis, and a dash (-) is used in the special case of a 100% gap. The true solution satisfies: $\|\tilde{\beta}^\dagger\|_\infty = 0.208$.

$p = 10^3$							
λ_2	λ_0	NNZ	M	L0BnB	GRB	MSK	B
$10\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	5	0.293	**1**	(3%)	158	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.245	**0.01**	(5%)	5	-
	$0.01\lambda_0^{\mathrm{m}}$	36	0.245	**(12%)**	(17%)	(16%)	-
$2\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	4	0.491	**3**	651	2737	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.332	**0.1**	68	106	-
	$0.01\lambda_0^{\mathrm{m}}$	15	0.332	**(3%)**	(5%)	(4%)	-
λ_2^*	$0.5\lambda_0^{\mathrm{m}}$	3	0.524	**20**	2648	1278	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.348	**0.3**	109	65	-
	$0.01\lambda_0^{\mathrm{m}}$	10	0.348	(10%)	(11%)	**(6%)**	-
$0.5\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	3	0.542	**59**	965	(6%)	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.356	**1**	63	172	-
	$0.01\lambda_0^{\mathrm{m}}$	10	0.356	(19%)	(16%)	**(12%)**	-
$0.1\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	3	0.557	**128**	697	(8%)	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.364	**1**	58	303	-
	$0.01\lambda_0^{\mathrm{m}}$	10	0.364	(47%)	**(32%)**	(55%)	-

$p = 10^4$							
λ_2	λ_0	NNZ	M	L0BnB	GRB	MSK	B
$10\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	5	0.293	**2**	-	1617	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.245	**0.1**	-	62	-
	$0.01\lambda_0^{\mathrm{m}}$	45	0.245	**(3%)**	-	(7%)	-
$2\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	8	0.491	**263**	-	(13%)	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.332	**0.7**	-	2459	-
	$0.01\lambda_0^{\mathrm{m}}$	17	0.332	**(15%)**	-	**(15%)**	-
λ_2^*	$0.5\lambda_0^{\mathrm{m}}$	3	0.524	**1920**	-	(18%)	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.348	**2**	-	3338	-
	$0.01\lambda_0^{\mathrm{m}}$	14	0.348	(29%)	-	**(24%)**	-
$0.5\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	3	0.542	**(3%)**	(29%)	(26%)	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.356	**5**	-	3483	-
	$0.01\lambda_0^{\mathrm{m}}$	14	0.356	(42%)	-	**(36%)**	-
$0.1\lambda_2^*$	$0.5\lambda_0^{\mathrm{m}}$	3	0.557	**(8%)**	-	(19%)	-
	$0.1\lambda_0^{\mathrm{m}}$	10	0.364	**24**	-	(46%)	-
	$0.01\lambda_0^{\mathrm{m}}$	13	0.364	**(73%)**	-	(77%)	-

$k^\dagger = 10$, and $\Sigma = \text{Diag}(\Sigma^{(1)}, \Sigma^{(2)}, \ldots, \Sigma^{(k^\dagger)})$ is block-diagonal. Such block structures are commonly used in the sparse learning literature [191]. Each $\Sigma^{(l)}$ is a correlation matrix with the same dimension. Given a correlation parameter $\rho \in (0, 1)$, for each $l \in [k^\dagger]$, we assume that $\Sigma^{(l)}_{ij} = \rho^{|i-j|}$ for any $i, j \in [p/k^\dagger]$, i.e., the correlation is exponentially decaying in each block. We report the timings for different values of the parameter ρ in Table 3.4. The results indicate that higher correlations lead to an increase in the run time, but even the high correlation instances can be solved in reasonable time. For example, when $p = 10^4$, $\rho = 0.9$ can be solved in less than 10 minutes. When $p = 10^5$, $\rho = 0.8$ can be solved in around 13 minutes, and $\rho = 0.9$ can be solved to a 13% gap in 2 hours.

Sparsity of the Regression Coefficients: We consider datasets with $n = 10^3$, $p = 10^4$, and the same correlation setting as the experiment for the number of samples. The results in Table 3.4 show that problems with 15 nonzeros can be handled in 166 seconds, and for 20 and 25 nonzeros, decent gaps ($\leq 10\%$) can be obtained in 2 hours. We also note that for larger λ_2 or tighter M choices, larger values of $k^\dagger$ can be handled.

Table 3.4: Run time in seconds (denoted by t) for L0BnB. If L0BnB does not solve the problem in a 2 hour time limit, the optimality gap is shown in parenthesis.

Varying n

$\boldsymbol{n}$	10^2	10^3	10^4	10^5
$\mathbf{t}$	(42%)	11	30	1701

Varying correlation coefficient ρ

$\boldsymbol{\rho}$	0.1	0.3	0.5	0.7	0.8	0.9
$\mathbf{t}(p = 10^4)$	4	4	5	16	55	530
$\mathbf{t}(p = 10^5)$	35	39	60	231	808	(13%)

Varying $k^\dagger$

$\boldsymbol{k}^\dagger$	10	15	20	25
$\mathbf{t}$	4	166	(10%)	(6%)

3.4.4 Real Data and Ablation Studies

High-dimensional Real Data: Here we investigate the run time of L0BnB on the Riboflavin dataset [46]—a genetics dataset used for predicting Vitamin B2 production levels. The original dataset has $p = 4088$ and $n = 71$. We augment the dataset with pairwise feature interactions to get $p = 8,362,004$. We mean center and normalize the response and the columns of the data matrix. We then run 5-fold cross-validation in `L0Learn` (with default parameters) to find the optimal regularization parameters $\hat{\lambda}_0$ and $\hat{\lambda}_2$. We set M

to be 10 times the ℓ_∞ norm of the solution obtained from cross-validation. In L0BnB, we solve the problem for $\lambda_2 \in \{0.1\hat{\lambda}_2, \hat{\lambda}_2\}$ and vary λ_0 to generate solutions of different support sizes. The run time, for each λ_2, as a function of the support size is reported in Figure 3-2. Interestingly, at $\hat{\lambda}_2$, all the support sizes obtained (up to 15 nonzeros) can be handled in less than a minute. Moreover, the increase in time is relatively slow as the number of nonzeros increases. When λ_2 becomes smaller (i.e., $\lambda_2 = 0.1\hat{\lambda}_2$), the run times increase (though they are reasonable) as the problem becomes more difficult. When an optimal solution has six nonzeros, L0BnB for $\lambda_2 = 0.1\hat{\lambda}_2$ is approximately 20x slower than $\lambda_2 = \hat{\lambda}_2$.

Ablation Study: We perform an ablation study to measure the effect of key choices in our BnB on the run time. Particularly, we consider the following changes: (i) replacing our relaxation solver with MOSEK, (ii) turning off warm starts in our solver, and (iii) turning off active sets in our solver. We measure the run time before and after these changes on synthetic data with $n = 10^3$, $k^\dagger = 10$, $\Sigma_{ij} = 0.1$ for all $i \neq j$ and 1 otherwise. We use the parameters $\lambda_0 = \lambda_0^*$, $\lambda_2 = \lambda_2^*$, and $M = 1.5M^*(\lambda_2^*)$ (with the same values as those used in the experiment of Section 3.4.2). The run times for different choices of p are reported in Table 3.5. The results show that replacing our relaxation solver with MOSEK will slow down the BnB by more than 1200x at $p = 10^4$. The results for MOSEK are likely to be even slower for $p = 10^6$ as it still had a 100% gap after 2 hours. We note that MOSEK employs a state-of-the-art conic optimization solver (based on an interior point method). The significant speed-ups here can be attributed to our CD-based algorithm, which is designed to effectively exploit the sparsity structure in the problem. The results also indicate that warm starts and active sets are important for run time, e.g., removing warm starts and active sets at $p = 10^5$ can slow down the algorithm by more than 16x and 67x, respectively.

3.5 Conclusion

We considered the exact computation of estimators from the $\ell_0\ell_2$-regularized least squares problem. While current approaches for this problem rely on commercial MIP-solvers, we propose a highly specialized nonlinear BnB framework for solving the problem. A key workhorse

Table 3.5: Time (seconds) after the following changes to our relaxation solver: (i) replacing it with MOSEK, (ii) removing warm starts, and (iii) removing active sets. Gap is shown in parenthesis if the method does not terminate in 2 hours.

p	10^4	10^5	10^6
L0BnB	3	34	1112
(i)	3802	(13%)	(100%)
(ii)	25	560	(21%)
(iii)	66	2291	(23%)

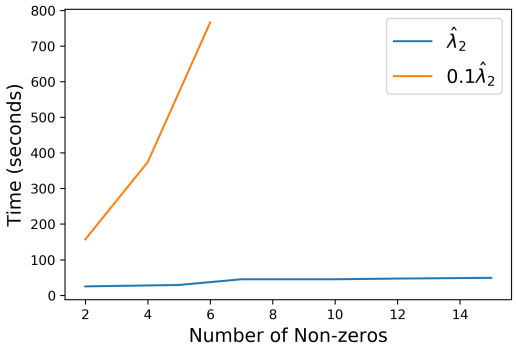


Figure 3-2: L0BnB run time on a real genomics dataset (Riboflavin) with $n = 71$ and $p \approx 8.3 \times 10^6$.

in our approach is a fast coordinate descent procedure to solve the node relaxations along with active set updates and gradient screening, which exploit information across the search tree for computational efficiency. Moreover, we proposed a new method for obtaining dual bounds from our primal coordinate descent solutions and showed that the quality of these bounds depend on the sparsity level, rather than the number of features p. Our experiments on both real and synthetic data indicate that our method exhibits over 5000x speedups compared to the fastest solvers, handling high-dimensional instances with $p = 8.3 \times 10^6$ in the order of seconds to few minutes. Our method appears to be more robust to the choices of the regularization parameters and can handle difficult statistical problems (e.g., relatively high correlations or small number of samples).

Our work demonstrates for the first time that carefully designed first-order methods can be highly effective within a BnB framework; and can perhaps, be applied to more general mixed integer programs involving sparsity. There are multiple directions for future work. [8] recently showed that safe screening rules, which eliminate variables from the optimization problem, can be a very effective preprocessing step for $\ell_0\ell_2$ regularized regression. One promising direction is to extend such rules to dynamically eliminate variables during the course of BnB. Another important direction is to develop specialized methods that can dynamically infer and tighten the Big-M values used in our formulation.

3.A Appendix: Proofs of Technical Results

Proof of Theorem 3.1

The interval relaxation of (3.3) can be expressed as:

$$\min_{\beta} \quad \Big\{\frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i\in[p]} \min_{s_i,z_i}(\lambda_0 z_i + \lambda_2 s_i)\Big\} \tag{3.35a}$$

$$\beta_i^2 \le s_i z_i,\ i \in [p] \tag{3.35b}$$

$$-Mz_i \le \beta_i \le Mz_i,\ i \in [p] \tag{3.35c}$$

$$z_i \in [0,1], s_i \ge 0,\ i \in [p] \tag{3.35d}$$

Let $\omega(\beta_i; \lambda_0, \lambda_2, M) := \min_{s_i,z_i}(\lambda_0 z_i + \lambda_2 s_i)$ s.t. (3.35b), (3.35c), (3.35d). Next we obtain an expression for $\omega(\beta_i; \lambda_0, \lambda_2, M)$.

Let (β_i, z_i, s_i) be a feasible solution for (3.35). Note that $\hat{z}_i := \max\{\frac{\beta_i^2}{s_i}, \frac{|\beta_i|}{M}\}$ is the smallest possible value of z_i, which satisfies constraints (3.35b) and (3.35c) (for the case of $\beta_i = s_i = 0$, we define $\beta_i^2/s_i = 0$). Thus, the objective value corresponding to $(\beta_i, \hat{z}_i, s_i)$ is less than or equal to that of a feasible solution (β_i, z_i, s_i). This implies that we can replace constraints (3.35b) and (3.35c) with the constraint $z_i = \max\{\frac{\beta_i^2}{s_i}, \frac{|\beta_i|}{M}\}$ without changing the optimal objective of the problem. This replacement leads to:

$$\omega(\beta_i; \lambda_0, \lambda_2, M) = \min_{s_i,z_i}\ (\lambda_0 z_i + \lambda_2 s_i) \quad \text{s.t.} \quad z_i = \max\Big\{\frac{\beta_i^2}{s_i}, \frac{|\beta_i|}{M}\Big\},\ z_i \in [0,1],\ s_i \ge 0.$$

In the above, we can eliminate the variable z_i, leading to:

$$\omega(\beta_i; \lambda_0, \lambda_2, M) = \min_{s_i} \max\Big\{\underbrace{\lambda_0\frac{\beta_i^2}{s_i} + \lambda_2 s_i}_{\text{Case I}}, \underbrace{\lambda_0\frac{|\beta_i|}{M} + \lambda_2 s_i}_{\text{Case II}}\Big\} \ \text{ s.t. } \ s_i \ge \beta_i^2,\ |\beta_i| \le M. \tag{3.36}$$

Suppose that Case I attains the maximum in (3.36). This holds if $s_i \le |\beta_i|M$. The function $\lambda_0\frac{\beta_i^2}{s_i} + \lambda_2 s_i$ is convex in s_i, and the optimality conditions of this function imply that the optimal solution is $s_i^* = |\beta_i|\sqrt{\lambda_0/\lambda_2}$ if $|\beta_i| \le \sqrt{\lambda_0/\lambda_2} \le M$ and $s_i^* = \beta_i^2$ if $\sqrt{\lambda_0/\lambda_2} \le$

$|\beta_i| \leq M$. Plugging s_i^* into (3.36) leads to $\omega(\beta_i; \lambda_0, \lambda_2, M) = 2\lambda_0 \mathcal{B}(\beta_i \sqrt{\lambda_2/\lambda_0})$, assuming $\sqrt{\lambda_0/\lambda_2} \leq M$.

Now suppose Case II attains the maximum in (3.36). This holds if $s_i \geq |\beta_i| M$. The function $\lambda_0 \frac{|\beta_i|}{M} + \lambda_2 s_i$ is monotonically increasing in s_i, where s_i is lower bounded by $|\beta_i| M$ and β_i^2. But we always have $|\beta_i| M \geq \beta_i^2$ (since $|\beta_i| \leq M$), which implies that the optimal solution is $s_i^* = |\beta_i| M$. Substituting s_i^* into (3.36) we get $\omega(\beta_i; \lambda_0, \lambda_2, M) = (\lambda_0/M + \lambda_2 M)|\beta_i|$, and this holds as long as $\sqrt{\lambda_0/\lambda_2} > M$.

Proof of Proposition 3.1

First, we show (3.7). Note that $V_{\mathrm{B}(M)}$ can be simplified to:

$$V_{\mathrm{B}(M)} = \min_{\|\beta\|_\infty \leq M} H(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i \in [p]} \Big(\frac{\lambda_0}{M} |\beta_i| + \lambda_2 \beta_i^2 \Big). \tag{3.37}$$

Recalling Theorem 3.1 and the definition of $F(\beta)$ in (3.5), note that for $\sqrt{\lambda_0/\lambda_2} > M$:

$$\begin{aligned} V_{\mathrm{PR}(M)} - V_{\mathrm{B}(M)} \geq F(\beta^*) - H(\beta^*) &= \sum\nolimits_{i \in [p]} \Big\{ \psi_2(\beta_i^*; \lambda_0, \lambda_2, M) - \frac{\lambda_0}{M} |\beta_i^*| - \lambda_2 (\beta_i^*)^2 \Big\} \\ &= \lambda_2 \sum\nolimits_{i \in [p]} (M|\beta_i^*| - (\beta_i^*)^2) \end{aligned}$$

where the inequality holds since β^*, an optimal solution to (3.5), is feasible for (3.37). This establishes (3.7).

We now show (3.8). Since $|\beta_i^*| \leq M$ and $\sqrt{\lambda_0/\lambda_2} > M$, we must have $\psi_1(\beta_i^*; \lambda_0, \lambda_2) = 2|\beta_i^*| \sqrt{\lambda_0 \lambda_2}$ (this corresponds to the first case in (3.4)). Also recall that $V_{\mathrm{PR}(\infty)} = \min_\beta G(\beta)$, where $G(\beta)$ is defined in (3.6) (this follows from applying Theorem 3.1 with $M = \infty$). The following then holds:

$$\begin{aligned} V_{\mathrm{PR}(M)} - V_{\mathrm{PR}(\infty)} \geq F(\beta^*) - G(\beta^*) &= \sum\nolimits_{i \in [p]} \Big\{ \psi_2(\beta_i^*; \lambda_0, \lambda_2, M) - \psi_1(\beta_i^*; \lambda_0, \lambda_2) \Big\} \\ &= h(\lambda_0, \lambda_2, M) \|\beta^*\|_1, \end{aligned} \tag{3.38}$$

where the inequality holds since β^*, an optimal solution to (3.5), is feasible for (3.6).

Proof of Proposition 3.2

First, we recall that $V_{\mathrm{PR}(\infty)} = \min_\beta G(\beta)$, where $G(\beta)$ is defined in (3.6), and that $V_{\mathrm{B}(M)}$ is defined in (3.37). Define a function $t : \mathbb{R} \to \mathbb{R}$ as follows

$$t(\beta_i) := 2\lambda_0 \mathcal{B}(\beta_i \sqrt{\lambda_2/\lambda_0}) - \frac{\lambda_0}{M}|\beta_i| - \lambda_2 \beta_i^2.$$

Next, we prove (3.10).

Proof of (3.10)**:** Suppose that $M \leq \frac{1}{2}\sqrt{\lambda_0/\lambda_2}$ and $|\beta_i| \leq M$. Using the fact that $|\beta_i| \leq \sqrt{\lambda_0/\lambda_2}$ and the definition of $\mathcal{B}$, we have $\mathcal{B}(\beta_i\sqrt{\lambda_2/\lambda_0}) = \sqrt{\lambda_2/\lambda_0}|\beta_i|$. This leads to:

$$t(\beta_i) = \Big(2\sqrt{\lambda_0\lambda_2} - \tfrac{\lambda_0}{M}\Big)|\beta_i| - \lambda_2\beta_i^2 \leq 0, \tag{3.39}$$

where the inequality follows since $2\sqrt{\lambda_0\lambda_2} - \lambda_0/M \leq 0$ for $M \leq \frac{1}{2}\sqrt{\lambda_0/\lambda_2}$. Now, let $\beta^\dagger$ be an optimal solution to (3.37). Then, the following holds:

$$V_{\mathrm{B}(M)} - V_{\mathrm{PR}(\infty)} \geq H(\beta^\dagger) - G(\beta^\dagger) = -\sum\nolimits_{i\in[p]} t(\beta_i^\dagger) \geq 0, \tag{3.40}$$

where the first inequality follows from $V_{\mathrm{B}(M)} = H(\beta^\dagger)$ and $V_{\mathrm{PR}(\infty)} \leq G(\beta^\dagger)$. The second inequality in (3.40) follows from (3.39). This establishes (3.10).

To show (3.11), we will need the following lemma.

Lemma 3.1. *Let $M \geq \sqrt{\lambda_0/\lambda_2}$, then for any $\beta_i \in [-M, M]$, we have $t(\beta_i) \geq 0$.*

Proof of Lemma 3.1. Suppose that $M \geq \sqrt{\lambda_0/\lambda_2}$ and $|\beta_i| \leq M$. There are two cases to consider here: Case (i): $|\beta_i| \leq \sqrt{\lambda_0/\lambda_2}$ and Case (ii): $|\beta_i| \geq \sqrt{\lambda_0/\lambda_2}$.
For Case (i), from the definition of $\mathcal{B}$, we have $2\lambda_0\mathcal{B}(\beta_i\sqrt{\lambda_2/\lambda_0}) = 2\sqrt{\lambda_0\lambda_2}|\beta_i|$. Therefore,

$$t(\beta_i) = \Big(2\sqrt{\lambda_0\lambda_2} - \frac{\lambda_0}{M}\Big)|\beta_i| - \lambda_2\beta_i^2.$$

In the above, it is easy to check that for $M \geq \sqrt{\lambda_0/\lambda_2}$ and $|\beta_i| \leq \sqrt{\lambda_0/\lambda_2}$, we have $t(\beta_i) \geq 0$.

Now for Case (ii), we have $2\lambda_0\mathcal{B}(\beta_i\sqrt{\lambda_2/\lambda_0}) = \lambda_0 + \lambda_2\beta_i^2$—this leads to $t(\beta_i) = \lambda_0\Big(1 - \frac{|\beta_i|}{M}\Big)$, which is non-negative since we assume that $|\beta_i| \leq M$. This establishes Lemma 3.1. □

Proof of (3.11)**:** Define $v^*(M) = \min_{\|\beta\|_\infty \leq M} G(\beta)$ (recall, $G(\beta)$ is defined in (3.6)) and let β^* be an optimal solution i.e., $v^*(M) = G(\beta^*)$. Suppose that $M \geq \sqrt{\lambda_0/\lambda_2}$. Then, the following holds:

$$v^*(M) - V_{\mathrm{B}(M)} \geq G(\beta^*) - H(\beta^*) = \sum_{i\in[p]} t(\beta_i^*) \geq 0 \tag{3.41}$$

where the first inequality holds since β^* is a feasible solution to (3.37) so $V_{\mathrm{B}(M)} \leq H(\beta^*)$, and the last inequality is due to Lemma 3.1.

Now suppose that $\lambda_2 \in \mathcal{L}(M)$ (defined in (3.9)) and let $\hat{\beta} \in \mathcal{S}(\lambda_2)$ (i.e., $\hat{\beta}$ is optimal for (3.6)) be such that it satisfies $\|\hat{\beta}\|_\infty \leq M$. Since $V_{\mathrm{PR}(\infty)} \leq v^*(M)$ and $\hat{\beta}$ is feasible for the problem corresponding to $v^*(M)$, then $v^*(M) = G(\hat{\beta})$. But by (3.41) we have $v^*(M) \geq V_{\mathrm{B}(M)}$, which combined with $v^*(M) = G(\hat{\beta})$, leads to $G(\hat{\beta}) \geq V_{\mathrm{B}(M)}$. This establishes (3.11) (since by definition $G(\hat{\beta}) = V_{\mathrm{PR}(\infty)}$); and completes the proof of Proposition 3.2.

Proof of Proposition 3.3

Fix some $i \in \mathrm{Supp}(\hat{\beta})^c$. Recall that the one-dimensional problem: $\min_{|\beta_i|\leq M} F(\hat{\beta}_1, \ldots, \beta_i, \ldots, \hat{\beta}_p)$ is equivalent to Problem (3.13), where $\tilde{\beta}_i = \langle y - \sum_{j\neq i} X_j\hat{\beta}_j, X_i\rangle$. Since $\hat{\beta}_i = 0$, we have $\tilde{\beta}_i = \langle \hat{r}, X_i\rangle$ where, $\hat{r} = y - X\hat{\beta}$. For $\sqrt{\lambda_0/\lambda_2} \leq M$, (3.14) implies that the solution of (3.13) is nonzero iff $|\tilde{\beta}_i| > 2\sqrt{\lambda_0\lambda_2}$. Similarly, for $\sqrt{\lambda_0/\lambda_2} > M$, by (3.15), the solution of (3.13) is nonzero iff $|\tilde{\beta}_i| > \lambda_0/M + \lambda_2 M$. Using these observations in the definition of $\mathcal{V}$ in Algorithm 3.2, leads to the result of the proposition.

Proof of Proposition 3.4

Fix some $i \in \mathcal{V}$. Note that

$$|\langle \hat{r}, X_i\rangle - \langle r^0, X_i\rangle| = |\langle X\beta^0 - X\hat{\beta}, X_i\rangle| \leq \|X\beta^0 - X\hat{\beta}\|_2\|X_i\|_2 \leq \epsilon.$$

Using the triangle inequality and the bound above, we get:

$$|\langle \hat{r}, X_i \rangle| \leq |\langle r^0, X_i \rangle| + |\langle \hat{r}, X_i \rangle - \langle r^0, X_i \rangle| \leq |\langle r^0, X_i \rangle| + \epsilon.$$

Therefore, if $i \in \mathcal{V}$, i.e., $|\langle \hat{r}, X_i \rangle| > c(\lambda_0, \lambda_2, M)$, then $|\langle r^0, X_i \rangle| + \epsilon > c(\lambda_0, \lambda_2, M)$, implying that $i \in \hat{\mathcal{V}}$, as defined in (3.19).

Proof of Theorem 3.2

Problem (3.5) can be written equivalently as:

$$\min_{\beta \in \mathbb{R}^p, r \in \mathbb{R}^n} \quad \frac{1}{2}\|r\|_2^2 + \sum_{i \in [p]} \psi(\beta_i; \lambda_0, \lambda_2, M) \;\; \text{s.t.} \;\; r = y - X\beta, \;\; |\beta_i| \leq M, \; \forall i \in [p]. \tag{3.42}$$

Case of $\sqrt{\lambda_0/\lambda_2} \leq M$: We first consider the case when $\sqrt{\lambda_0/\lambda_2} \leq M$. We dualize all the constraints in (3.42), leading to the following Lagrangian dual:

$$\max_{\alpha \in \mathbb{R}^n, \eta \in \mathbb{R}^p_{\geq 0}} \quad \min_{\beta \in \mathbb{R}^p, r \in \mathbb{R}^n} \quad L(\beta, r, \alpha, \eta) \tag{3.43}$$

where $L(\beta, r, \alpha, \eta)$ is the Lagrangian function defined as follows:

$$L(\beta, r, \alpha, \eta) := \frac{1}{2}\|r\|_2^2 + \sum_{i \in [p]} \psi_1(\beta_i; \lambda_0, \lambda_2) + \alpha^T(r - y + X\beta) + \sum_{i \in [p]} \eta_i(|\beta_i| - M).$$

Next, we discuss how to solve the inner minimization problem in (3.43). Let us write:

$$\min_{\beta, r} \quad L(\beta, r, \alpha, \eta) = \min_r \left\{ \frac{1}{2}\|r\|_2^2 + \alpha^T r \right\} + \sum_{i \in [p]} \min_{\beta_i} \Big\{ D_i(\beta_i) \Big\} - \alpha^T y - \sum_{i \in [p]} M\eta_i \tag{3.44}$$

where $D_i(\beta_i) := \psi_1(\beta_i; \lambda_0, \lambda_2) + \alpha^T X_i \beta_i + \eta_i |\beta_i|$. Note that the minimizer wrt r is given by $\tilde{r}^* = -\alpha$. In what follows, we consider some $i \in [p]$ and derive an optimal solution $\tilde{\beta}_i^*$ of $\min_{\beta_i} D_i(\beta_i)$. There are two cases to consider here.

Case 1: $|\beta_i| \leq \sqrt{\lambda_0/\lambda_2}$. Here, $\psi_1(\beta_i; \lambda_0, \lambda_2) = 2\sqrt{\lambda_0\lambda_2}|\beta_i|$. Note that

$$\tilde{\beta}_i^* = \arg\min_{\beta_i} \; D_i(\beta_i) = \begin{cases} 0 & \text{if} \quad 2\sqrt{\lambda_0\lambda_2} + \eta_i - |\alpha^T X_i| \geq 0 \\ -\sqrt{\lambda_0/\lambda_2}\,\text{sign}(\alpha^T X_i) & \text{otherwise} \end{cases} \tag{3.45}$$

and as we will see, the second case above is a special case of Case 2 below.

Case 2: $|\beta_i| \geq \sqrt{\lambda_0/\lambda_2}$. In this case, $\psi_1(\beta_i; \lambda_0, \lambda_2) = \lambda_2\beta_i^2 + \lambda_0$. Note that

$$\tilde{\beta}_i^* = \arg\min_{\beta_i} D_i(\beta_i) = \arg\min_{\beta_i} \lambda_2\beta_i^2 + \alpha^T X_i\beta_i + \eta_i|\beta_i| = T(-\alpha^T X_i; \eta_i, \infty)/(2\lambda_2) \tag{3.46}$$

where, T above is the soft-thresholding operator [69]. But $\tilde{\beta}_i^*$ in (3.46) should satisfy $|\tilde{\beta}_i^*| \geq \sqrt{\lambda_0/\lambda_2}$ in this case. Using the definition of $T(.)$, the latter condition can be written as: $(|\alpha^T X_i| - \eta_i)/(2\lambda_2) \geq \sqrt{\lambda_0/\lambda_2}$, which simplifies to $|\alpha^T X_i| - \eta_i \geq 2\sqrt{\lambda_0\lambda_2}$. In this case, $D_i(\tilde{\beta}_i^*)$ can be written as:

$$D_i(\tilde{\beta}_i^*) = -\frac{1}{4\lambda_2}\Big(|\alpha^T X_i| - \eta_i\Big)^2 + \lambda_0 \quad \text{if} \quad |\alpha^T X_i| - \eta_i \geq 2\sqrt{\lambda_0\lambda_2}. \tag{3.47}$$

Combining (3.45) and (3.47), we have: $D_i(\tilde{\beta}_i^*) = -\Big[\frac{(|\alpha^T X_i| - \eta_i)^2}{4\lambda_2} - \lambda_0\Big]_+$. Plugging the above expression of $D_i(\tilde{\beta}_i^*)$ along with $\tilde{r}^* = -\alpha$ in (3.43), we get the following dual problem:

$$\max_{\alpha \in \mathbb{R}^n, \eta \in \mathbb{R}^p_{\geq 0}} \; -\frac{1}{2}\|\alpha\|_2^2 - \alpha^T y - \sum_{i=1}^p \Big[\frac{(|\alpha^T X_i| - \eta_i)^2}{4\lambda_2} - \lambda_0\Big]_+ - \sum_{i=1}^p M\eta_i. \tag{3.48}$$

Note we can drop the non-negativity constraint $\eta_i \geq 0$ above. Using a new variable γ in place of η (note that $\eta_i = |\gamma_i|$ for all i), Problem (3.48) can be reformulated as:

$$\max_{\alpha \in \mathbb{R}^n, \gamma \in \mathbb{R}^p} \; -\frac{1}{2}\|\alpha\|_2^2 - \alpha^T y - \sum_{i=1}^p \Big[\frac{(\alpha^T X_i - \gamma_i)^2}{4\lambda_2} - \lambda_0\Big]_+ - \sum_{i=1}^p M|\gamma_i| \tag{3.49}$$

which is the formulation in (3.20).

We now show (3.23). Let (α^*, η^*) be an optimal solution of (3.48). First, it is easy to see

$r^* = \arg\min_r L(\beta^*, r, \alpha^*, \eta^*) = -\alpha^*$. For the second part of (3.23), note by complementary slackness: if $|\beta_i^*| < M$ then $\eta_i^* = 0$ and consequently $\gamma_i^* = 0$. Next, we will derive γ_i^* for the case of $|\beta_i^*| = M > 0$. We first consider the case of $\beta_i^* = M$. Using (3.46) with the identifications: $\tilde{\beta}_i^* = M$, $\alpha = \alpha^*$, and $\eta = \eta^*$, we get $M = (-\alpha^{*T} X_i - \eta_i^* \text{sign}(-\alpha^{*T} X_i))/(2\lambda_2)$, where $\text{sign}(\alpha^{*T} X_i) = -1$. Using this along with the fact that $\gamma_i^* = \eta_i^* \text{sign}(\alpha^{*T} X_i)$, we obtain $\gamma_i^* = \alpha^{*T} X_i + 2\lambda_2 M = \alpha^{*T} X_i - 2\lambda_2 M \text{sign}(\alpha^{*T} X_i)$. Using this identity along with a similar argument for $\beta_i = -M$, we arrive at the expression in (3.23).

Case of $\sqrt{\lambda_0/\lambda_2} > M$: The proof for this case follows along the lines similar to what was shown above. We omit the proof.

The following lemma is useful for proving Theorem 3.3.

Lemma 3.2. *Suppose $\sqrt{\lambda_0/\lambda_2} \leq M$. Let $\hat{\beta}$ be a solution from Algorithm 3.2 and $(\hat{\alpha}, \hat{\gamma})$ be the corresponding dual solution defined in* (3.25)*. Let β^* and r^* be as defined in Theorem 3.2, and define the primal gap $\epsilon = \|X(\beta^* - \hat{\beta})\|_2$. Then, for every $i \in \text{Supp}(\hat{\beta})^c$, we have $v(\hat{\alpha}, \hat{\gamma}_i) = 0$ (see* (3.21) *for definition of v); and for every $i \in \text{Supp}(\hat{\beta})$, we have*

$$v(\hat{\alpha}, \hat{\gamma}_i) \leq c_i \epsilon + (4\lambda_2)^{-1} \epsilon^2 + v(\alpha^*, \gamma_i^*), \tag{3.50}$$

where $c_i = (2\lambda_2)^{-1}$ if $|\beta_i^*| < M$, and $c_i = M$ if $|\beta_i^*| = M$.

Proof of Lemma 3.2. Fix some $i \in \text{Supp}(\hat{\beta})^c$. Since $\hat{\beta}$ is the output of Algorithm 3.2, the set $\mathcal{V}$ in Algorithm 3.2 must be empty. Thus, using Proposition 3.3, we have: $|\hat{r}^T X_i| \leq 2\sqrt{\lambda_0 \lambda_2}$. As $\hat{r} = -\hat{\alpha}$, and consequently using the definition of $\hat{\gamma}$ in (3.26), we have $\hat{\gamma}_i = 0$. Thus,

$$(\hat{\alpha}^T X_i - \hat{\gamma}_i)^2/(4\lambda_2) = (\hat{r}^T X_i)^2/(4\lambda_2) \leq \lambda_0,$$

which implies that $v(\hat{\alpha}, \hat{\gamma}_i) = 0$.

Now fix some $i \in \text{Supp}(\hat{\beta})$, and let $a := (\hat{\alpha} - \alpha^*)^T X_i$ and $b := \alpha^{*T} X_i - \gamma_i^*$. By (3.25) and the definition of h_1 in (3.20), we have $\hat{\gamma}_i = \arg\min_{\gamma_i} v(\hat{\alpha}, \gamma_i)$, which leads to $v(\hat{\alpha}, \hat{\gamma}_i) \leq v(\hat{\alpha}, \gamma_i^*)$.

An upper bound on $v(\hat{\alpha}, \hat{\gamma}_i)$ can be then obtained as follows:

$$\begin{aligned} v(\hat{\alpha}, \hat{\gamma}_i) \leq v(\hat{\alpha}, \gamma_i^*) &= \Big[\frac{(\hat{\alpha}^T X_i - \gamma_i^*)^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\ &= \Big[\frac{((\hat{\alpha} - \alpha^*)^T X_i + \alpha^{*T} X_i - \gamma_i^*)^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\ &= \Big[\frac{a^2 + 2ab + b^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\ &\leq \frac{a^2 + 2|a||b|}{4\lambda_2} + \Big[\frac{b^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\ &\leq \frac{a^2 + 2|a||b|}{4\lambda_2} + v(\alpha^*, \gamma_i^*). \end{aligned} \tag{3.51}$$

Next, we obtain upper bounds on $|a|$ and $|b|$. By the Cauchy-Schwarz inequality, we have $|a| \leq \|X(\beta^* - \hat{\beta})\|_2 \|X_i\|_2 = \epsilon \|X_i\|_2$. Since the columns of X are normalized, the bound simplifies to $|a| \leq \epsilon$. Since, $\|y\|_2 = 1$ and $\alpha^* = -r^*$ (see Theorem 3.2), we have:

$$\frac{1}{2}\|\alpha^*\|_2^2 = \frac{1}{2}\|r^*\|_2^2 \leq F(\beta^*) \leq F(0) = \frac{1}{2}\|y\|_2^2 = \frac{1}{2},$$

implying that $\|\alpha^*\|_2 \leq 1$.

We now present a bound on $|b| = |\alpha^{*T} X_i - \gamma_i^*|$ by considering two cases:

Case 1: ($|\beta_i^*| < M$): From the definition of γ_i^* in (3.23), we have $\gamma_i^* = 0$, which leads to $|b| = |\alpha^{*T} X_i| \leq \|\alpha^*\|_2 \|X_i\|_2 \leq 1$.

Case 2: ($|\beta_i^*| = M$): By (3.23), $\gamma_i^* = \alpha^{*T} X_i - 2M\lambda_2 \operatorname{sign}(\alpha^{*T} X_i)$, which leads to $|b| \leq 2M\lambda_2$.

Plugging $|a| \leq \epsilon$ and the bounds on $|b|$ (above) into (3.51), we arrive to (3.50). □

Proof of Theorem 3.3

We consider two cases based on $\sqrt{\lambda_0/\lambda_2} \leq M$ or $\sqrt{\lambda_0/\lambda_2} > M$.

Showing Bound (3.29)**:** First, we consider the case of $\sqrt{\lambda_0/\lambda_2} \leq M$, i.e., we will establish

the bound in (3.29). Note that the following holds:

$$\begin{aligned}\frac{1}{2}\|\hat{\alpha}\|_2^2 + \hat{\alpha}^T y &= \frac{1}{2}\|\hat{\alpha} - \alpha^* + \alpha^*\|_2^2 + (\hat{\alpha} - \alpha^* + \alpha^*)^T y \\ &= \frac{1}{2}\|\hat{\alpha} - \alpha^*\|_2^2 + (\hat{\alpha} - \alpha^*)^T \alpha^* + \frac{1}{2}\|\alpha^*\|_2^2 + (\hat{\alpha} - \alpha^*)^T y + \alpha^{*T} y \\ &\leq \frac{1}{2}\epsilon^2 + \epsilon\|\alpha^*\|_2 + \frac{1}{2}\|\alpha^*\|_2^2 + \epsilon\|y\|_2 + \alpha^{*T} y, \end{aligned} \tag{3.52}$$

where the inequality above follows by applying Cauchy-Schwarz and noting that $\|\hat{\alpha} - \alpha^*\|_2 = \|X(\beta^* - \hat{\beta})\|_2 = \epsilon$. Using $\|y\|_2 = 1$ and $\|\alpha^*\|_2 \leq 1$ (this was established in the proof of Lemma 3.2) in (3.52), we get:

$$\frac{1}{2}\|\hat{\alpha}\|_2^2 + \hat{\alpha}^T y \leq \frac{1}{2}\epsilon^2 + 2\epsilon + \frac{1}{2}\|\alpha^*\|_2^2 + \alpha^{*T} y. \tag{3.53}$$

Plugging (3.53) into the objective function in (3.20) (with $\alpha = \hat{\alpha}$ and $\gamma = \hat{\gamma}$):

$$h_1(\hat{\alpha}, \hat{\gamma}) \geq -\frac{1}{2}\|\alpha^*\|_2^2 - \alpha^{*T} y - 2\epsilon - \frac{1}{2}\epsilon^2 - \sum_{i \in [p]} v(\hat{\alpha}, \hat{\gamma}_i). \tag{3.54}$$

By Lemma 3.2, for every $i \in \text{Supp}(\hat{\beta})^c$, we have $v(\hat{\alpha}, \hat{\gamma}_i) = 0$, which implies $v(\hat{\alpha}, \hat{\gamma}_i) \leq v(\alpha^*, \gamma_i^*)$ (since v is a non-negative function).

Using the inequality $v(\hat{\alpha}, \hat{\gamma}_i) \leq v(\alpha^*, \gamma_i^*)$ for every $i \in \text{Supp}(\hat{\beta})^c$; and inequality (3.50) for every $i \in \text{Supp}(\hat{\beta})$, in (3.54), we get:

$$\begin{aligned} h_1(\hat{\alpha}, \hat{\gamma}) &\geq -\frac{1}{2}\|\alpha^*\|_2^2 - \alpha^{*T} y - \sum_{i \in [p]} v(\alpha^*, \gamma_i^*) - 2\epsilon - \frac{1}{2}\epsilon^2 - \sum_{i \in \text{Supp}(\hat{\beta})} \Big(c_i \epsilon + (4\lambda_2)^{-1} \epsilon^2 \Big) \\ &= h_1(\alpha^*, \gamma^*) - 2\epsilon - \frac{1}{2}\epsilon^2 - \sum_{i \in \text{Supp}(\hat{\beta})} \Big(c_i \epsilon + (4\lambda_2)^{-1} \epsilon^2 \Big). \end{aligned} \tag{3.55}$$

Let

$$k_1 = |\{i \in \text{Supp}(\hat{\beta}) \mid |\hat{\beta}_i| < M\}| \quad \text{and} \quad k_2 = |\{i \in \text{Supp}(\hat{\beta}) \mid |\hat{\beta}_i| = M\}|.$$

Using the expressions for c_is (from Lemma 3.2) in (3.55), we get:

$$h_1(\hat{\alpha},\hat{\gamma}) \geq h_1(\alpha^*,\gamma^*) - 2\epsilon - \frac{1}{2}\epsilon^2 - k_1\Big((2\lambda_2)^{-1}\epsilon + (4\lambda_2)^{-1}\epsilon^2\Big) - k_2\Big(M\epsilon + (4\lambda_2)^{-1}\epsilon^2\Big)$$

Rearranging the terms in the above and using the fact that $k = k_1 + k_2$, we get:

$$h_1(\hat{\alpha},\hat{\gamma}) \geq h_1(\alpha^*,\gamma^*) - \epsilon(2 + k_1(2\lambda_2)^{-1} + k_2 M) - \epsilon^2\Big(\frac{1}{2} + \frac{k}{4\lambda_2}\Big),$$

which leads to (3.29).

Showing Bound (3.30)**:** Now, we consider the case of $\sqrt{\lambda_0/\lambda_2} > M$, where we will establish the bound in (3.30). By the same argument used in deriving (3.54), we have:

$$h_2(\hat{\rho},\hat{\mu}) \geq -\frac{1}{2}\|\rho^*\|_2^2 - \rho^{*T}y - 2\epsilon - \frac{1}{2}\epsilon^2 - M\|\hat{\mu}\|_1. \tag{3.56}$$

Next, we fix some $i \in \text{Supp}(\hat{\beta})$ and we upper bound $\hat{\mu}_i$ as follows:

$$\begin{aligned}
|\hat{\mu}_i| = \Big[|\hat{\rho}^T X_i| - \lambda_0/M - \lambda_2 M\Big]_+ \leq& \Big[|(\hat{\rho}-\rho^*)^T X_i| + |\rho^{*T} X_i| - \lambda_0/M - \lambda_2 M\Big]_+ \\
\leq& |(\hat{\rho}-\rho^*)^T X_i| + \Big[|\rho^{*T} X_i| - \lambda_0/M - \lambda_2 M\Big]_+ \\
\leq& \epsilon + |\mu_i^*|,
\end{aligned} \tag{3.57}$$

where in the last step above we use Cauchy-Schwarz for the first term (noting that $\rho^* = -r^*$); and for the second term, we use Theorem 3.2 along with the following:

$$|\rho^{*T} X_i| \leq \lambda_0/M + \lambda_2 M \;\text{ if }\; |\beta_i^*| < M \;\text{ and }\; |\rho^{*T} X_i| \geq \lambda_0/M + \lambda_2 M \;\text{ if }\; |\beta_i^*| = M. \tag{3.58}$$

Conditions in (3.58) follow from the optimality of (ρ^*, μ^*) for (3.22). (For $|\beta_i^*| < M$, (3.24) implies that $\mu_i^* = 0$, which leads to $|\rho^{*T} X_i| \leq \lambda_0/M + \lambda_2 M$ from the feasibility condition in (3.22). When $|\beta_i^*| = M$, we will establish (3.58) via contradiction: If $|\rho^{*T} X_i| < \lambda_0/M + \lambda_2 M$ holds true, then (3.24) implies $\mu_i^* < 0$, which would violate optimality for (3.22).)

For $i \in \text{Supp}(\hat{\beta})^c$, we have $\hat{\mu}_i = 0$ (see the discussion after (3.28)), which implies $|\hat{\mu}_i| \leq |\mu_i^*|$.

Using the inequalities $|\hat{\mu}_i| \leq |\mu_i^*|$, $i \in \text{Supp}(\hat{\beta})^c$ and (3.57) (for all $i \in \text{Supp}(\hat{\beta})$) in (3.56), and simplifying, we get:

$$h_2(\hat{\rho}, \hat{\mu}) \geq h_2(\rho^*, \mu^*) - \epsilon(2 + Mk) - \epsilon^2/2, \tag{3.59}$$

which leads to (3.30).

3.B Appendix: Additional Technical Details

3.B.1 Derivation of Solutions to Problem (3.13)

First, we consider the setting of $\sqrt{\lambda_0/\lambda_2} \leq M$. From Theorem 3.1, the penalty $\psi(\beta_i; \lambda_0, \lambda_2, M) = \psi_1(\beta_i; \lambda_0, \lambda_2) = 2\lambda_0 \mathcal{B}(\beta_i \sqrt{\lambda_2/\lambda_0})$. Using the definition of $\mathcal{B}$ in (3.4), we have:

$$\psi_1(\beta_i; \lambda_0, \lambda_2) = \begin{cases} 2\sqrt{\lambda_0\lambda_2}|\beta_i| & \text{if } |\beta_i| \leq \sqrt{\lambda_0/\lambda_2} \\ \lambda_2\beta_i^2 + \lambda_0 & \text{if } |\beta_i| \geq \sqrt{\lambda_0/\lambda_2}. \end{cases}$$

Case of $|\beta_i| \leq \sqrt{\lambda_0/\lambda_2}$: The penalty in this case is $2\sqrt{\lambda_0\lambda_2}|\beta_i|$, and the first-order optimality conditions imply that the solution of Problem (3.13) is given by a capped version (due to the box constraint on β_i) of the soft thresholding operator [69]: $\beta_i^* = T(\tilde{\beta}_i; 2\sqrt{\lambda_0\lambda_2}, M)$; and this is optimal as long as: $|\beta_i^*| \leq \sqrt{\lambda_0/\lambda_2}$, i.e., $|T(\tilde{\beta}_i; 2\sqrt{\lambda_0\lambda_2}, M)| \leq \sqrt{\lambda_0/\lambda_2}$. Therefore, if $|\tilde{\beta}_i| \leq \sqrt{\lambda_0/\lambda_2} + 2\sqrt{\lambda_0\lambda_2}$, the solution is given by $T(\tilde{\beta}_i; 2\sqrt{\lambda_0\lambda_2}, M)$.

Case of $|\beta_i| \geq \sqrt{\lambda_0/\lambda_2}$: Here the penalty is $\lambda_2\beta_i^2 + \lambda_0$, and the solution is given by $T(\tilde{\beta}_i(1 + 2\lambda_2)^{-1}; 0, M)$. The latter solution is optimal as long as $|T(\tilde{\beta}_i(1 + 2\lambda_2)^{-1}; 0, M)| \geq \sqrt{\lambda_0/\lambda_2}$, which can be simplified to $|\tilde{\beta}_i| \geq \sqrt{\lambda_0/\lambda_2} + 2\sqrt{\lambda_0\lambda_2}$.
This completes the derivation of (3.14).

Finally, we consider the setting of $\sqrt{\lambda_0/\lambda_2} > M$. By Theorem 3.1, the penalty is $\psi(\beta_i; \lambda_0, \lambda_2, M) = (\lambda_0/M + \lambda_2 M)|\beta_i|$. Thus, using arguments similar to the above, the solution is $T(\tilde{\beta}_i; \lambda_0/M + \lambda_2 M, M)$. This completes the derivation of (3.15).

3.B.2 Node Subproblems

In Theorem 3.1, we presented a reformulation of the root relaxation in the β space. Here we discuss how to similarly reformulate and solve the subproblem at an arbitrary node in the search tree. Recall that at any node, some z_is can be fixed to 0 or 1 (this depends on the branching decisions made until reaching the node). At a given node, let $\mathcal{Z}$ and $\mathcal{N}$ be the sets of indices of the z_is that are fixed to 0 and 1, respectively. Then, the following convex subproblem needs to be solved at the node:

$$\begin{aligned} \min_{\beta,z,s} \quad & \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0 \sum_{i\in[p]} z_i + \lambda_2 \sum_{i\in[p]} s_i \\ \text{s.t.} \quad & \beta_i^2 \le s_i z_i, \ i \in [p] \\ & -Mz_i \le \beta_i \le Mz_i, \ i \in [p] \\ & s_i \ge 0, \ i \in [p] \\ & z_i = 0, i \in \mathcal{Z}, \quad z_i = 1, i \in \mathcal{N}, \quad z_i \in [0,1], i \in [p] \setminus (\mathcal{Z} \cup \mathcal{N}). \end{aligned} \tag{3.60}$$

Define the penalty $\tilde{\psi}(\beta_i; \lambda_0, \lambda_2) := \lambda_0 + \lambda_2 \beta_i^2$. Then, using an argument similar to that in the proof of Theorem 3.1, Problem (3.60) can be equivalently expressed in the β space as:

$$\begin{aligned} \min_{\beta\in\mathbb{R}^p} \ \tilde{F}(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i\in\mathcal{N}^c} \psi(\beta_i; \lambda_0, \lambda_2, M) + \sum_{i\in\mathcal{N}} \tilde{\psi}(\beta_i; \lambda_0, \lambda_2) \\ \text{s.t.} \ \ \|\beta\|_\infty \le M, \quad \beta_i = 0, i \in \mathcal{Z}. \end{aligned} \tag{3.61}$$

Note that Problem (3.61) is similar to Problem (3.5), except that (i) the variables corresponding to $\mathcal{N}$ use the penalty $\tilde{\psi}(\beta_i; \lambda_0, \lambda_2)$ instead of $\psi(\beta_i; \lambda_0, \lambda_2, M)$, and (ii) the variables corresponding to $\mathcal{Z}$ are fixed to zero. To optimize Problem (3.61) using cyclic CD, the coordinates in $(\mathcal{N} \cup \mathcal{Z})^c$ are updated exactly as described in Section 3.3.1. Next, we discuss how cyclic CD updates the coordinates in $\mathcal{N}$. For any $i \in \mathcal{N}$, the algorithm updates coordinate i by solving the problem:

$$\min_{\beta_i\in\mathbb{R}} \tilde{F}(\hat{\beta}_1, \ldots, \beta_i, \ldots, \hat{\beta}_p) \qquad \text{s.t.} \qquad |\beta_i| \le M.$$

Assuming the columns of X have unit ℓ_2 norm, the problem above is equivalent to (3.13) but with $\psi(\beta_i; \lambda_0, \lambda_2, M)$ replaced with $\tilde{\psi}(\beta_i; \lambda_0, \lambda_2)$; and the corresponding solution is given by: $T(\tilde{\beta}_i(1+2\lambda_2)^{-1}; 0, M)$.

3.C Appendix: Additional Experimental Results and Details

Experiment of Section 3.4.2

In Table 3.6, we report the running time of L0BnB, with and without gradient screening. The number of nodes explored by the different solvers is reported in Table 3.7. In this experiment: $\lambda_2^* = 0.0409$; and λ_0^* is equal to 0.012, 0.0115, and 0.0132, for λ_2 set to λ_2^*, $0.1\lambda_2^*$, and $10\lambda_2^*$, respectively.

Table 3.6: Running time in seconds for L0BnB, with and without gradient screening (GS). This experiment is based on the same data and parameters as Table 3.1. Both approaches explore exactly the same BnB tree.

	$\lambda_2 = \lambda_2^*$		$\lambda_2 = 0.1\lambda_2^*$		$\lambda_2 = 10\lambda_2^*$	
p	No GS	With GS	No GS	With GS	No GS	With GS
10^3	**0.7**	0.9	**1.6**	2.1	0.011	0.011
10^4	3.3	**2.9**	11.5	**11.3**	0.06	**0.05**
10^5	34	**19**	545	**459**	0.5	0.5
10^6	1112	**414**	(23%)	**(8%)**	8.1	**7.3**

	$M = M^*$		$M = 2M^*$		$M = 4M^*$		$M = \infty$	
p	No GS	With GS	No GS	With GS	No GS	With GS	No GS	With GS
10^3	**0.16**	0.25	**0.8**	0.98	**0.8**	1.1	**0.8**	1.1
10^4	0.9	**0.6**	4.9	**4.3**	5.4	**4.8**	5.5	**4.8**
10^5	7.5	**3.9**	70	**43**	81	**50**	81	**50**
10^6	121	**52**	9265	**4640**	10986	**5639**	11010	**5612**

Experiment of Section 3.4.2

The parameters λ_0 and λ_2, and the number of BnB nodes explored are reported in Table 3.8.

Table 3.7: Number of BnB nodes explored in the experiment of Section 3.4.2. For [28], we report the number of cuts used by the cutting-plane algorithm. A dash indicates that the method could not solve more than 1 node in 4 hours.

	$\lambda_2 = \lambda_2^*$					$\lambda_2 = 0.1\lambda_2^*$					$\lambda_2 = 10\lambda_2^*$				
p	L0BnB	GRB	MSK	B	[28]	L0BnB	GRB	MSK	B	[28]	L0BnB	GRB	MSK	B	[28]
10^3	159	532	161	2	9895	329	31	13	-	35563	3	374	347	-	9
10^4	225	506	382	-	4769	531	3	13	-	5212	3	357	1143	-	9
10^5	385	-	-	-	-	4337	-	-	-	-	3	-	-	-	10
10^6	1087	-	-	-	-	10104	-	-	-	-	3	-	-	-	11

	$M = M^*$				$M = 2M^*$				$M = 4M^*$				$M = \infty$			
p	L0BnB	GRB	MSK	B	L0BnB	GRB	MSK	B	L0BnB	GRB	MSK	B	L0BnB	GRB	MSK	B
10^3	49	65	48	3	185	714	258	-	193	2120	261	-	193	658	273	-
10^4	61	128	67	-	287	2	867	-	305	1	852	-	305	-	1403	-
10^5	75	-	-	-	693	-	-	-	761	-	-	-	761	-	-	-
10^6	119	-	-	-	8919	-	-	-	10059	-	-	-	9805	-	-	-

Table 3.8: The parameters λ_0 and λ_2, and the number of nodes explored by the different solvers in the experiment of Section 3.4.2. The parameter M is reported in the main text.

SNR	λ_0^*	λ_2^*	L0BnB	GRB	MSK	B
0.5	0.00402	0.126	34155	85402	7784	1
1	0.0057	0.0869	223	4020	2840	1
2	0.0097	0.0596	217	868	736	1
3	0.0113	0.0409	221	461	400	1
4	0.0121	0.0409	189	428	227	1
5	0.0120	0.0409	159	532	161	1

Experiment of Section 3.4.2

The parameter $\lambda_2^* = 0.0409$. The parameter λ_0 and the number of BnB nodes explored are reported in Table 3.9.

Experiment of Section 3.4.3

The number of nodes for all experiments of Section 3.4.3, and the parameters for varying n and k are presented in Table 3.10. For varying correlation coefficient: for $p = 10^4$, we have $\lambda_0^* = 0.0326$, $\lambda_2^* = 0.0091$, and $M = 0.465$ (for all values of ρ). For $p = 10^5$, we have $\lambda_0^* = 0.0298$, $\lambda_2^* = 0.00202$, and $M = 0.449$ for all ρ, except for $\rho = 0.9$ which has $\lambda_0^* = 0.0304$.

Computing Setup and Software

We used four machines in our experiments, each using an Intel(R) Xeon(R) Platinum 8252C CPU and 48GB of RAM. We ran Gurobi and MOSEK using their Python APIs, BARON using the Pyomo Python API [81], and [28]'s OA approach using the SubsetSelectionCIO toolkit [25] in Julia. Relevant software versions: Ubuntu 18.04.3, Python 3.7.10, R 3.6.3, Julia 0.6.4, Gurobi 9.0.1, MOSEK 9.2.3, BARON 2021.1.13, Pyomo 5.7.1, and L0Learn 2.0. For Julia, exceptionally, we used Gurobi 8.0.1 due to compatibility issues with Gurobi 9.

Table 3.9: The parameter λ_0 and the number of BnB nodes explored in the experiment of Section 3.4.2.

$p = 10^3$					
λ_2	λ_0	L0BnB	GRB	MSK	B
$10\lambda_2^*$	0.02692	343	60303	376	1
	0.00538	1	27103	1	1
	0.00054	270993	2056	10271	1
$2\lambda_2^*$	0.04207	1051	55882	1764	1
	0.00841	27	354	89	1
	0.00084	187743	2045	9977	1
λ_2^*	0.04526	2825	26721	7564	1
	0.00905	71	343	237	1
	0.00091	175876	27259	9819	1
$0.5\lambda_2^*$	0.04704	6417	18821	10959	1
	0.00941	113	242	394	1
	0.00094	66769	18817	9884	1
$0.1\lambda_2^*$	0.04856	11811	12518	13643	1
	0.00971	175	218	802	1
	0.00097	22096	19983	10228	1

$p = 10^4$					
λ_2	λ_0	L0BnB	GRB	MSK	B
$10\lambda_2^*$	0.02692	343	1	378	1
	0.00538	1	1	1	1
	0.00054	56504	1	826	1
$2\lambda_2^*$	0.04207	10097	1	901	1
	0.00841	37	1	470	1
	0.00084	49358	1	820	1
λ_2^*	0.04526	43745	1	886	1
	0.00905	111	1	801	1
	0.00091	32142	1	824	1
$0.5\lambda_2^*$	0.04704	75893	353	915	1
	0.00941	191	1	906	1
	0.00094	18253	1	822	1
$0.1\lambda_2^*$	0.04856	96091	1	857	1
	0.00971	507	1	1056	1
	0.00097	10433	1	828	1

Table 3.10: Problem parameters and number of nodes for the experiment of Section 3.4.3.

Varying n

$\boldsymbol{n}$	10^2	10^3	10^4	10^5
$\boldsymbol{\lambda}_0$	0.0104	0.0121	0.0152	0.0181
$\boldsymbol{\lambda}_2$	0.0409	0.00139	0.00295	0.0001
$\boldsymbol{M}$	0.410	0.337	0.329	0.316
Nodes	1386125	383	259	421

Varying $k^\dagger$

$\boldsymbol{k}^\dagger$	10	15	20	25
$\boldsymbol{\lambda}_0$	0.0119	0.00615	0.00415	0.00275
$\boldsymbol{\lambda}_2$	0.0409	0.0409	0.0596	0.126
$\boldsymbol{M}$	0.348	0.275	0.224	0.186
Nodes	225	5261	218667	139353

Varying correlation coefficient ρ

$\boldsymbol{\rho}$	0.1	0.3	0.5	0.7	0.8	0.9
Nodes ($p = 10^4$)	597	613	845	2295	7913	86083
Nodes ($p = 10^5$)	589	633	943	3145	11873	127267

Chapter 4

Learning Sparse Classifiers: Continuous and Mixed Integer Optimization Perspectives

This chapter is based on [60] (to appear in the Journal of Machine Learning Research). It is a joint work with Antoine Dedieu and Rahul Mazumder.

4.1 Introduction

We consider the problem of sparse linear classification, where the output depends on a linear combination of a small subset of features. This is a core problem in high-dimensional statistics [85] where the number of features p is comparable to or exceeds the number of samples n. In such settings, sparsity can be useful from a statistical viewpoint and can lead to more interpretable models. We consider the typical binary classification problem with samples $(x_i, y_i), i = 1, \ldots, n$, features $x_i \in \mathbb{R}^p$, and outcome $y_i \in \{-1, +1\}$. In the spirit of best-subset selection in linear regression [122], we consider minimizing the empirical loss (i.e., a surrogate for the misclassification error) while penalizing the number of nonzero

coefficients:

$$\min_{\beta \in \mathbb{R}^p} \quad \frac{1}{n} \sum_{i=1}^{n} f\left(\langle x_i, \beta \rangle, y_i\right) + \lambda_0 \|\beta\|_0, \tag{4.1}$$

where $f : \mathbb{R} \times \{-1, +1\} \to \mathbb{R}$ is the loss function (for example, hinge or logistic loss). The term $\|\beta\|_0$ is the ℓ_0 (pseudo)-norm of β which is equal to the number of nonzeros in β, and $\lambda_0 > 0$ is a regularization parameter which controls the number of nonzeros in β. We ignore the intercept term in the above and throughout this chapter to simplify the presentation. Problem (4.1) is known to be NP-Hard and poses computational challenges [128]. In this chapter, we introduce scalable algorithms for this optimization problem using techniques based on both continuous and discrete optimization (specifically, mixed integer programming [177]).

There is an impressive body of work on obtaining approximate solutions to Problem (4.1): popular candidates include greedy (a.k.a. stepwise) procedures [13], proximal gradient methods [36], among others. The ℓ_1-norm [162] is often used as a convex surrogate to the ℓ_0-norm, leading to a convex optimization problem. Nonconvex continuous penalties (such as MCP and SCAD) [185] provide better approximations of the ℓ_0-penalty but lead to nonconvex problems, for which gradient-based methods [72, 138, 107] and coordinate descent [41, 117] are often used. These algorithms may not deliver optimal solutions for the associated nonconvex problem. Fairly recently, there has been considerable interest in exploring Mixed Integer Programming (MIP)-based methods [177, 24, 166, 153] to solve Problem (4.1) to optimality. MIP-based methods create a branch-and-bound tree that simultaneously leads to feasible solutions and corresponding lower-bounds (aka dual bounds). Therefore, these methods deliver optimality certificates for the nonconvex optimization problem. Despite their appeal in delivering nearly optimal solutions to Problem (4.1), MIP-based algorithms are usually computationally expensive compared to convex relaxations or greedy (heuristic) algorithms [86, 84]—possibly limiting their use in time-sensitive applications that arise in practice.

The vanilla version of best-subset selection is often perceived as a gold-standard for high-dimensional sparse linear regression, when the signal-to-noise ratio (SNR) is high. However,

it suffers from overfitting when the SNR becomes moderately low [68, 115, 84]. A possible way to mitigate this shortcoming is by imposing additional continuous regularization—see for example, [115, 86] for studies in the (linear) regression setting. Thus, we consider an extended family of estimators which combines ℓ_0 and ℓ_q (for $q \in \{1, 2\}$) regularization:

$$\min_{\beta \in \mathbb{R}^p} \quad \frac{1}{n} \sum_{i=1}^{n} f\left(\langle x_i, \beta \rangle, y_i\right) + \lambda_0 \|\beta\|_0 + \lambda_q \|\beta\|_q^q, \tag{4.2}$$

where the regularization parameter $\lambda_0 \geq 0$ explicitly controls the sparsity in β, and $\lambda_q \geq 0$ controls the amount of continuous shrinkage on the nonzero coefficients of β (for example, the margin in linear SVM). In other words, $\lambda_0 \|\beta\|_0$ is used to directly control the number of nonzeros in the model coefficients, while $\lambda_q \|\beta\|_q^q$ controls the ℓ_q-norm of β i.e., provides *shrinkage*. Together, the ℓ_0-ℓ_q penalty controls model complexity versus data-fidelity. For flexibility, our framework allows for both choices of $q \in \{1, 2\}$, and the value of q needs to be specified a-priori by the practitioner. When $q = 2$, Problem (4.2) seeks to deliver a solution β with few nonzeros (controlled by the ℓ_0-penalty) and a small ℓ_2-norm (controlled by the ridge penalty). Similarly, when $q = 1$, we seek a model β that has a small ℓ_1-norm and a small ℓ_0-norm. If λ_1 is large, the ℓ_1-penalty may also encourage zeros in the coefficients. Note that the primary role of the ℓ_0-penalty is to control the number of nonzeros in β; and that of the ℓ_1-penalty is to shrink the model coefficients. In our numerical experiments we observe both choices of $q \in \{1, 2\}$ to work quite well, with no penalty uniformly dominating the other. We refer the reader to [115] for complementary discussions in the regression setting.

A primary focus of our work is to propose new scalable algorithms for solving Problem (4.2), with certificates of optimality (suitably defined). Problem (4.2) can be expressed using MIP formulations. However, these formulations lead to computational challenges for off-the-shelf commercial MIP solvers (such as Gurobi and CPLEX). To this end, we propose a new MIP-based algorithm that we call "integrality generation", which allows for solving instances of Problem (4.2) with $p \approx 50,000$ (where n is small) to optimality within a few minutes.[1] This appears to be well beyond the capabilities of state-of-the-art MIP solvers,

[1]The runtime also depends upon the number of nonzeros in the solution. In our experience, the runtime can increase if the number of nonzeros in an optimal solution becomes large.

including recent MIP-based approaches, as outlined below. To obtain high-quality solutions for larger problem instances, in times comparable to the fast ℓ_1-based solvers [69], we propose approximate algorithms based on coordinate descent (CD) [178] and local combinatorial optimization,[2] where the latter leads to higher quality solutions compared to CD. Our CD and local combinatorial optimization algorithms are publicly available through our fast C++/R toolkit `L0Learn`: on CRAN at `https://cran.r-project.org/package=L0Learn` and also at `https://github.com/hazimehh/L0Learn`.

From a statistical viewpoint, we establish new upper bounds on the estimation error for solutions obtained by globally minimizing ℓ_0-based estimators (4.2). These error bounds (rates) appear to be better than current known bounds for ℓ_1-regularization; and have rates similar to the optimal minimax rates for sparse least squares regression [147], achieved by ℓ_0-based regression procedures.

Related Work and Contributions: There is a vast body of work on developing optimization algorithms and understanding the statistical properties of various sparse estimators [85, 44]. We present a brief overview of work that relates to this chapter.

Computation: An impressive body of work has developed fast algorithms for minimizing the empirical risk regularized with convex or nonconvex proxies to the ℓ_0-norm, e.g., [69, 41, 117, 132, 154]. Below, we discuss related work that directly optimize objective functions involving an ℓ_0 norm (in the objective or as a constraint).

Until recently, global optimization with ℓ_0-penalization was rarely used beyond $p > 30$ as popular software packages for best-subset selection (for example, `leaps` and `bestglm`) are unable to handle larger instances. [24] demonstrated that ℓ_0-regularized regression problems could be solved to near-optimality for $p \approx 10^3$ by leveraging advances in first order methods and the capabilities of modern MIP solvers such as Gurobi. [23, 153] extend the work of [24] to solve ℓ_0-regularized logistic regression by using an outer-approximation approach that can address problems with p in the order of a few hundreds. [28] propose a cutting plane

[2]The local combinatorial optimization algorithms are based on solving MIP problems over restricted search-spaces; and are usually much faster to solve compared to the full problem (4.2).

algorithm for the ℓ_0-constrained least squares problem with additional ridge regularization—they can handle problems with $n \approx p$, when the feature correlations are low and/or the amount of ridge regularization is taken to be sufficiently large. [25] adapt [28]'s work to solve classification problems (e.g., with logistic or hinge loss). The approach of [25] appears to require a fairly high amount of ridge regularization for the cutting plane algorithm to work well.

A separate line of research investigates algorithms to obtain feasible solutions for ℓ_0-regularized problems. These algorithms do not provide dual bounds like MIP-based algorithms, but can be computationally much faster. These include: (i) first-order optimization algorithms based on hard thresholding, such as Iterative Hard Thresholding (IHT) [36] and GraSP [13], (ii) second-order optimization algorithms inspired by the Newton method such as NTGP [184], NHTP [192], and NSLR [175], and (iii) coordinate descent methods based on greedy and random coordinate selection rules [16, 140].

[86] present algorithms that offer a *bridge* between MIP-based global optimization and good feasible solutions for ℓ_0-regularized problems, by using a combination of CD and local combinatorial optimization. This chapter is similar in spirit, but makes new contributions. We extend the work of [86] (which is tailored to the least squares loss function) to address the more general class of problems in (4.2). Our algorithms can deliver solutions with better statistical performance (for example, in terms of variable selection and prediction error) compared to the popular fast algorithms for sparse learning (e.g., based on ℓ_1 and MCP regularizers). Unlike heuristics that simply deliver an upper bound, MIP-based approaches attempt to solve (4.2) to optimality. They can (i) certify (via dual bounds) the quality of solutions obtained by our CD and local search algorithms; and (ii) improve the solution if it is not optimal. However, as off-the-shelf MIP-solvers do not scale well, we present a new method: the *Integrality Generation Algorithm* (IGA) (see Section 4.3) that allows us to *solve* (to optimality) the MIP problems for instances that are larger than current methods [24, 23, 25, 28]. The key idea behind our proposed IGA is to solve a sequence of relaxations of (4.2) by allowing only a subset of variables to be binary. On the contrary, a direct MIP formulation for (4.2) requires p many binary variables; and can be prohibitively

expensive for moderate values of p.

Statistical Properties: Statistical properties of high-dimensional linear regression have been widely studied [47, 147, 45, 48, 29]. One important statistical performance measure is the ℓ_2-estimation error defined as $\|\beta^* - \hat{\beta}\|_2^2$, where β^* is the k-sparse vector used in generating the true model and $\hat{\beta}$ is an estimator. For regression problems, [47, 147] established a $(k/n)\log(p/k)$ lower bound on the ℓ_2-estimation error. This optimal minimax rate is known to be achieved by a global minimizer of an ℓ_0-regularized estimator [45]. It is well known that the Dantzig Selector and Lasso estimators achieve a $(k/n)\log(p)$ error rate [48, 29] under suitable assumptions for the high-dimensional regression setting. Compared to regression, there has been limited work in deriving estimation error bounds for classification tasks. [159] study margin adaptation for ℓ_1-norm SVM. A sizable amount of work focuses on the analysis of generalization error and risk bounds [73, 168]. [188] study variable selection consistency of a nonconvex penalized SVM estimator, using a local linear approximation method with a suitable initialization. Recently, [142] proved a $(k/n)\log(p)$ upper-bound for the ℓ_2-estimation error of ℓ_1-regularized support vector machines (SVM), where k is the number of nonzeros in the estimator that minimizes the population risk. [150] show consistent neighborhood selection for high-dimensional Ising model using an ℓ_1-regularized logistic regression estimator. [144] show that one can obtain an error rate of $k/n\log(p/k)$ for 1-bit compressed sensing problems. In this chapter, we present (to our knowledge) new ℓ_2-estimation error bounds for a (global) minimizer of Problem (4.1)—our framework applies to a family of loss functions including the hinge and logistic loss functions.

Our Contributions: We summarize our contributions below:

- We develop fast first-order algorithms based on cyclic CD and local combinatorial search to (approximately) solve Problem (4.2) (see Section 4.2). We prove a new result which establishes the convergence of cyclic CD under an asymptotic linear rate. We show that combinatorial search leads to solutions of higher quality than IHT and CD-based methods. We discuss how solutions from the ℓ_0-penalized formulation, i.e., Problem (4.2), can be used to obtain solutions to the cardinality constrained variant

of (4.2). We open source these algorithms through our sparse learning toolkit `L0Learn`.

- We propose a new algorithm: IGA, for solving Problem (4.2) to optimality. Our algorithm reduces the time for solving a MIP formulation of Problem (4.2) from the order of hours to seconds, and it can solve high-dimensional instances with $p \approx 50,000$ and small n. The algorithm is presented in Section 4.3.

- We establish upper bounds on the squared ℓ_2-estimation error for a cardinality constrained variant of Problem (4.2). Our $(k/n)\log(p/k)$ upper bound matches the optimal minimax rate known for regression.

- On a series of high-dimensional synthetic and real data sets (with $p \approx 10^5$), we show that our proposed algorithms can achieve significantly better statistical performance in terms of prediction (AUC), variable selection accuracy, and support sizes, compared to state-of-the-art algorithms (based on ℓ_1 and local solutions to ℓ_0 and MCP regularizers). Our proposed CD algorithm compares favorably in terms of runtime compared to current popular toolkits [69, 41, 13, 192] for sparse classification.

4.1.1 Preliminaries and Notation

For convenience, we introduce the following notation:

$$g(\beta) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f\left(\langle x_i, \beta\rangle, y_i\right) \quad \text{and} \quad G(\beta) \stackrel{\text{def}}{=} g(\beta) + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2.$$

Problem (4.2) is an instance of the following (more general) problem:

$$\min_{\beta\in\mathbb{R}^p} P(\beta) \stackrel{\text{def}}{=} G(\beta) + \lambda_0\|\beta\|_0. \tag{4.3}$$

In particular, Problem (4.2) with $q = 1$ is equivalent to Problem (4.3) with $\lambda_2 = 0$. Similarly, Problem (4.2) with $q = 2$ is equivalent to Problem (4.3) with $\lambda_1 = 0$. Problem (3) will be the focus in our algorithmic development.

We denote the set $\{1, 2, \ldots, p\}$ by $[p]$ and the canonical basis for $\mathbb{R}^p$ by $e_1, \ldots, e_p$. For $\beta \in \mathbb{R}^p$,

Loss	$f(\hat{v}, v)$	FO & Local Search	MIP	Error Bounds
Logistic	$\log(1 + e^{-\hat{v}v})$	✓	✓	✓
Squared Hinge	$\max(0, 1 - \hat{v}v)^2$	✓	✓	✗
Hinge	$\max(0, 1 - \hat{v}v)$	✓*	✓	✓

Table 4.1: Examples of loss functions we consider. "*" denotes that our proposed first-order and local search methods apply upon using [133]'s smoothing on the non-smooth loss function.

we use $\text{Supp}(\beta)$ to denote the support of β, i.e., the indices of its nonzero entries. For $S \subseteq [p]$, $\beta_S \in \mathbb{R}^{|S|}$ denotes the subvector of β with indices in S. Moreover, for a differentiable function $g(\beta)$, we use the notation $\nabla_S g(\beta)$ to refer to the subvector of $\nabla g(\beta)$ restricted to coordinates in S. We let $\mathbb{Z}$ and $\mathbb{Z}_+$ denote the set of integers and non-negative integers, respectively. A convex function $g(\beta)$ is said to be μ-strongly convex if $\beta \mapsto g(\beta) - \mu\|\beta\|_2^2/2$ is convex. A function $h(\beta)$ is said to be Lipschitz with parameter L if $\|h(\beta) - h(\alpha)\|_2 \leq L\|\beta - \alpha\|_2$ for all β, α in the domain of the function.

4.1.2 Examples of Loss Functions Considered

In Table 4.1, we give examples of popular classification loss functions that fall within the premise of our algorithmic framework and statistical theory. The column "FO & Local Search" indicates whether these loss functions are amenable to our first-order and local search algorithms (discussed in Section 4.2).[3] The column "MIP" indicates whether the loss function leads to an optimization problem that can be solved (to optimality) via the MIP methods discussed in Section 4.3. Finally, the column "Error Bounds" indicates if the statistical error bounds (estimation error) discussed in Section 4.4 apply to the loss function.

[3]That is, these algorithms are guaranteed to converge to a stationary point (or a local optimum) for the corresponding optimization problems.

4.2 First-Order and Local Combinatorial Search Algorithms

Here we present fast cyclic CD and local combinatorial search algorithms for obtaining high-quality local minima (we make this notion precise later) for Problem (4.3). Our framework assumes that $g(\beta)$ is differentiable and has a Lipschitz continuous gradient. We first present a brief overview of the key ideas presented in this section, before diving into the technical details.

Due to the nonconvexity of (4.3), the quality of the solution obtained depends on the algorithm—with local search and MIP-based algorithms leading to solutions of higher quality. The fixed points of the algorithms considered satisfy certain necessary optimality conditions for (4.3), leading to different classes of local minima. In terms of solution quality, there is a hierarchy among these classes. We show that for Problem (4.3), the minima corresponding to the different algorithms satisfy the following hierarchy:

$$\text{MIP Minima} \subseteq \text{Local Search Minima} \subseteq \text{CD Minima} \subseteq \text{IHT Minima}. \tag{4.4}$$

The fixed points of the IHT algorithm contain the fixed points of the CD algorithm. As we move to the left in the hierarchy, the fixed points of the algorithms satisfy stricter necessary optimality conditions. At the top of the hierarchy, we have the global minimizers, which can be obtained by solving a MIP formulation of (4.3).

Our CD and local search algorithms can run in times comparable to the fast ℓ_1-regularized approaches [69]. These algorithms can lead to high-quality solutions that can be used as warm starts for MIP-based algorithms. The MIP framework of Section 4.3 can be used to certify the quality of these solutions (via dual bounds) and to improve over them (if they are sub-optimal).

In Section 4.2.1, we introduce cyclic CD for Problem (4.3) and study its convergence properties. Section 4.2.2 discusses how the solutions of cyclic CD can be improved by local search

and presents a fast heuristic for performing local search in high dimensions. In Section 4.2.3, we discuss how our algorithms can be used to obtain high-quality (feasible) solutions to the *cardinality constrained* counterpart of (4.3), in which the complexity measure $\|\beta\|_0$ appears as a constraint and not a penalty as in (4.3). Finally, in Section 4.2.4, we briefly present implementation aspects of our toolkit `L0Learn`.

4.2.1 Cyclic Coordinate Descent: Algorithm and Computational Guarantees

We describe a cyclic CD algorithm for Problem (4.3) and establish its convergence to stationary points of (4.3).

Why cyclic CD? We briefly discuss our rationale for choosing cyclic CD. Cyclic CD has been shown to be among the fastest algorithms for fitting generalized linear models with convex and nonconvex regularization (e.g., ℓ_1, MCP, and SCAD) [69, 117, 41]. Indeed, it can effectively exploit sparsity and active-set updates, making it suitable for solving high-dimensional problems (e.g., with $p \sim 10^6$ and small n). Algorithms that require evaluation of the full gradient at every iteration (such as proximal gradient, stepwise, IHT or greedy CD algorithms) have difficulties in scaling with p [133]. In an earlier work, [140] proposed random CD for problems similar to (4.3) (without an ℓ_1-regularization term in the objective). However, recent studies have shown that cyclic CD can be faster than random CD [18, 75, 86]. Furthermore, for ℓ_0-regularized regression problems, cyclic CD is empirically seen to obtain solutions of higher quality (e.g., in terms of optimization and statistical performance) compared to random CD [86].

Setup. Our cyclic CD algorithm for (4.3) applies to problems where $g(\beta)$ is convex, continuously differentiable, and non-negative. Moreover, we will assume that the gradient of $\beta \mapsto g(\beta)$ is coordinate-wise Lipschitz continuous, i.e., for every $i \in [p]$, $\beta \in \mathbb{R}^p$ and $s \in \mathbb{R}$, we have:

$$|\nabla_i g(\beta + e_i s) - \nabla_i g(\beta)| \leq L_i |s|, \tag{4.5}$$

where $L_i > 0$ is the Lipschitz constant for coordinate i. This assumption leads to the block

Descent Lemma [22], which states that

$$g(\beta + e_i s) \leq g(\beta) + s\nabla_i g(\beta) + \frac{1}{2} L_i s^2. \tag{4.6}$$

Several popular loss functions for classification fall under the above setup. For example, logistic loss and squared hinge loss satisfy (4.5) with $L_i = \|X_i\|_2^2/4n$ and $L_i = 2\|X_i\|_2^2/n$, respectively (here, X_i denotes the i-th column of the data matrix X).

CD Algorithm. Cyclic CD [22] updates one coordinate at a time (with others held fixed) in a cyclical fashion. Given a solution $\beta \in \mathbb{R}^p$, we attempt to find a new solution by changing the i-th coordinate of β—i.e., we find α such that $\alpha_j = \beta_j$ for all $j \neq i$ and α_i minimizes the one-dimensional function: $\beta_i \mapsto P(\beta)$. However, for the examples we consider (e.g., logistic and squared hinge losses), there is no closed-form expression for this minimization problem. This makes the algorithm computationally inefficient compared to g being the squared error loss [86]. Using (4.6), we consider a quadratic upper bound $\tilde{g}(\alpha;\beta)$ for $g(\alpha)$ as follows:

$$g(\alpha) \leq \tilde{g}(\alpha;\beta) \stackrel{\text{def}}{=} g(\beta) + (\alpha_i - \beta_i)\nabla_i g(\beta) + \frac{\hat{L}_i}{2}(\alpha_i - \beta_i)^2, \tag{4.7}$$

where $\hat{L}_i$ is a constant which satisfies $\hat{L}_i > L_i$. (For notational convenience, we hide the dependence of $\tilde{g}$ on i.) Let us define the function

$$\psi(\alpha) = \sum_i \psi_i(\alpha_i) \quad \text{where} \quad \psi_i(\alpha_i) = \lambda_0 \mathbb{1}(\alpha_i \neq 0) + \lambda_1|\alpha_i| + \lambda_2\alpha_i^2.$$

Adding $\psi(\alpha)$ to both sides of equation (4.7), we get:

$$P(\alpha) \leq \widetilde{P}_{\hat{L}_i}(\alpha;\beta) \stackrel{\text{def}}{=} \tilde{g}(\alpha;\beta) + \psi(\alpha). \tag{4.8}$$

Following [133], we can approximately minimize $P(\alpha)$ w.r.t. α_i (with other coordinates held fixed) by minimizing its upper bound $\widetilde{P}_{\hat{L}_i}(\alpha;\beta)$ w.r.t. α_i. A solution $\hat{\alpha}_i$ for this one-

dimensional optimization problem is given by

$$\hat{\alpha}_i \in \arg\min_{\alpha_i} \ \widetilde{P}_{\hat{L}_i}(\alpha;\beta) \quad = \quad \arg\min_{\alpha_i} \ \frac{\hat{L}_i}{2}\left(\alpha_i - \left(\beta_i - \frac{1}{\hat{L}_i}\nabla_i g(\beta)\right)\right)^2 + \psi_i(\alpha_i). \tag{4.9}$$

Let $\hat{\alpha}$ be a vector whose i-th component is $\hat{\alpha}_i$, and $\hat{\alpha}_j = \beta_j$ for all $j \neq i$. Note that $P(\hat{\alpha}) \leq P(\beta)$—i.e., updating the i-th coordinate via (4.9) with all other coefficients held fixed, leads to a decrease in the objective value $P(\beta)$. A solution of (4.9) can be computed in closed-form; and is given by the thresholding operator $T : \mathbb{R} \to \mathbb{R}$ defined as follows:

$$T(c;\lambda,\hat{L}_i) = \begin{cases} \frac{\hat{L}_i}{\hat{L}_i+2\lambda_2}\left(|c| - \frac{\lambda_1}{\hat{L}_i}\right)\text{sign}(c) & \text{if } \frac{\hat{L}_i}{\hat{L}_i+2\lambda_2}\left(|c| - \frac{\lambda_1}{\hat{L}_i}\right) \geq \sqrt{\frac{2\lambda_0}{\hat{L}_i+2\lambda_2}} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

where $c = \beta_i - \nabla_i g(\beta)/\hat{L}_i$ and $\lambda = (\lambda_0, \lambda_1, \lambda_2)$.

Algorithm 4.1 below, summarizes our cyclic CD algorithm.

Algorithm 4.1: Cyclic Coordinate Descent (CD)

- **Input:** Initialization β^0 and constant $\hat{L}_i > L_i$ for every $i \in [p]$
- **Repeat for** $l = 0, 1, 2, \ldots$ **until convergence:**
  1. $i \leftarrow 1 + (l \mod p)$ and $\beta_j^{l+1} \leftarrow \beta_j^l$ for all $j \neq i$
  2. Update $\beta_i^{l+1} \leftarrow \arg\min_{\beta_i} \widetilde{P}_{\hat{L}_i}(\beta_1^l, \ldots, \beta_i, \ldots, \beta_p^l; \beta^l)$ using (4.10) with $c = \beta_i^l - \nabla_i g(\beta^l)/\hat{L}_i$

Computational Guarantees. The convergence of cyclic CD has been extensively studied for certain classes of continuous objective functions, e.g., see [165, 22, 18] and the references therein. However, these results do not apply to our objective function due to the discontinuity in the ℓ_0-norm. In Theorem 4.1, we establish a new result which shows that cyclic CD (Algorithm 4.1) converges at an asymptotic linear rate, and we present a characterization of the corresponding solution. Theorem 4.1 is established under the following assumption:

Assumption 4.1. *Problem* (4.3) *satisfies at least one of the following conditions:*

1. *Strong convexity of the continuous regularizer, i.e.,* $\lambda_2 > 0$.

2. *Restricted Strong Convexity: For some* $u \in [p]$*, the function* $\beta_S \mapsto g(\beta_S)$ *is strongly convex for every* $S \subseteq [p]$ *such that* $|S| \leq u$*. Moreover,* λ_0 *and the initial solution* β^0 *are chosen such that* $P(\beta^0) < u\lambda_0$.

Theorem 4.1. *Let* $\{\beta^l\}$ *be the sequence of iterates generated by Algorithm 4.1. Suppose that Assumption 4.1 holds, then:*

1. *The support of* β^l *stabilizes in a finite number of iterations, i.e., there exists an integer* N *and support* $S \subset [p]$ *such that* $Supp(\beta^l) = S$ *for all* $l \geq N$.

2. *Let* S *be the support as defined in Part 1. The sequence* $\{\beta^l\}$ *converges to a solution* β^* *with support* S*, satisfying:*

$$\begin{aligned} & \beta^*_S \in \arg\min_{\beta_S} G(\beta_S) \\ & |\beta^*_i| \geq \sqrt{\frac{2\lambda_0}{\hat{L}_i + 2\lambda_2}} \quad \text{for } i \in S \\ \text{and} \quad & |\nabla_i g(\beta^*)| - \lambda_1 \leq \sqrt{2\lambda_0(\hat{L}_i + 2\lambda_2)} \quad \text{for } i \in S^c. \end{aligned} \tag{4.11}$$

3. *Let* S *be the support as defined in Part 1. Let us define* $H(\beta_S) \stackrel{\text{def}}{=} g(\beta_S) + \lambda_2 \|\beta_S\|_2^2$. *Let* σ_S *be the strong convexity parameter of* $\beta_S \mapsto H(\beta_S)$*, and let* L_S *be the Lipschitz constant of* $\beta_S \mapsto \nabla_S H(\beta_S)$*. Denote* $\hat{L}_{\max} = \max_{i \in S} \hat{L}_i$ *and* $\hat{L}_{\min} = \min_{i \in S} \hat{L}_i$*. Then, there exists an integer* N' *such that the following holds for all* $t \geq N'$*:*

$$P(\beta^{(t+1)p}) - P(\beta^*) \leq \left(1 - \frac{\sigma_S}{\gamma}\right) \left(P(\beta^{tp}) - P(\beta^*)\right), \tag{4.12}$$

where $\gamma^{-1} = 2\hat{L}_{\max}(1 + |S|L_S^2 \hat{L}_{\min}^{-2})$.

We provide a proof of Theorem 4.1 in the appendix. The proof is different from that of CD for ℓ_0-regularized regression [86] since we use inexact minimization for every coordinate update, whereas [86] use exact minimization. At a high level, the proof proceeds as follows. In Part

1, we prove a sufficient decrease condition which establishes that the support stabilizes in a finite number of iterations. In Part 2, we show that under Assumption 4.1, the objective function is strongly convex when restricted to the stabilized support. After restriction to the stabilized support, we obtain convergence from standard results on cyclic CD [22]. In Part 3, we show an asymptotic linear rate of convergence for Algorithm 4.1. To establish this rate, we extend the linear rate of convergence of cyclic CD for smooth strongly convex functions by [18] to our objective function (note that due to the presence of the ℓ_1-norm, our objective is not smooth even after support stabilization).

Stationary Points of CD versus IHT: The conditions in (4.11) describe a fixed point of the cyclic CD algorithm and are necessary optimality conditions for Problem (4.3). We now show that the stationary conditions (4.11) are strictly contained within the class of stationary points arising from the IHT algorithm [36, 16, 24]. Recall that IHT can be interpreted as a proximal gradient algorithm, whose updates for Problem (4.3) are given by:

$$\beta^{l+1} \in \arg\min_{\beta} \left\{ \frac{1}{2\tau} \|\beta - (\beta^l - \tau \nabla g(\beta^l))\|_2^2 + \psi(\beta) \right\}, \tag{4.13}$$

where $\tau > 0$ is a step size. Let L be the Lipschitz constant of $\beta \mapsto \nabla g(\beta)$, and let $\hat{L}$ be any constant satisfying $\hat{L} > L$. Update (4.13) is guaranteed to converge to a stationary point if $\tau = 1/\hat{L}$ (e.g., see [112, 86]). Note that $\tilde{\beta}$ is a fixed point for (4.13) if it satisfies (4.11) with $\hat{L}_i$ replaced with $\hat{L}$. The component-wise Lipschitz constant L_i always satisfies $L_i \leq L$. For high-dimensional problems, we may have $L_i \ll L$ (see discussions in [16, 86] for problems where L grows with p but L_i is constant). Hence, the CD optimality conditions in (4.11) are more restrictive than IHT—justifying a part of the hierarchy mentioned in (4.4). An important practical consequence of this result is that CD may lead to solutions of higher quality than IHT.

Remark 4.1. *A solution β^* that satisfies the CD or IHT stationarity conditions is a local minimizer in the traditional sense used in nonlinear optimization.*[4] *Thus, we use the terms stationary point and local minimizer interchangeably in our exposition.*

[4]That is, given a stationary point β^*, there is a small $\epsilon > 0$ such that any β lying in the set $\|\beta - \beta^*\|_2 \leq \epsilon$ will have an objective that is at least as large as the current objective value $P(\beta^*)$.

4.2.2 Local Combinatorial Search

We propose a local combinatorial search algorithm to improve the quality of solutions obtained by Algorithm 4.1. Given a solution from Algorithm 4.1, the idea is to perform small perturbations to its support in an attempt to improve the objective. This approach has been recently shown to be very effective (e.g, in terms of statistical performance) for ℓ_0-regularized regression [86], especially under difficult statistical settings (high feature correlations or n is small compared to p). Here, we extend the approach of [86] to general loss functions, discussed in Section 4.2.1. As we consider a general loss function, performing exact local minimization becomes computationally expensive—we thus resort to an approximate minimization scheme. This makes our approach different from the least squares setting considered in [86].

Our local search algorithm is iterative. It performs the following two steps at every iteration t:

1. **Coordinate Descent**: We run cyclic CD (Algorithm 4.1) initialized from the current solution, to obtain a solution β^t with support S.

2. **Combinatorial Search**: We attempt to improve β^t by making a change to its current support S via a *swap* operation. In particular, we search for two subsets of coordinates $S_1 \subset S$ and $S_2 \subset S^c$, each of size at most m, such that removing coordinates S_1 from the support, adding S_2 to the support, and then optimizing over the coefficients in S_2, improves the current objective value.

To present an optimization formulation for the combinatorial search step (discussed above), we introduce some notation. Let U^S denote a $p \times p$ matrix whose i-th row is e_i^T if $i \in S$ and zero otherwise. Thus, for any $\beta \in \mathbb{R}^p$, $(U^S\beta)_i = \beta_i$ if $i \in S$ and $(U^S\beta)_i = 0$ if $i \notin S$. The combinatorial search step solves the following optimization problem:

$$\min_{S_1,S_2,\beta} \quad P(\beta^t - U^{S_1}\beta^t + U^{S_2}\beta) \qquad \text{s.t.} \qquad S_1 \subset S, S_2 \subset S^c, |S_1| \leq m, |S_2| \leq m, \tag{4.14}$$

where the optimization variables are the subsets S_1, S_2 and the coefficients of β restricted

to S_2. If there is a feasible solution $\hat{\beta}$ to (4.14) satisfying $P(\hat{\beta}) < P(\beta^t)$, then we move to $\hat{\beta}$. Otherwise, the current solution β^t cannot be improved by swapping subsets of coordinates, and the algorithm terminates. We summarize the algorithm below.

Algorithm 4.2: CD with Local Combinatorial Search

- **Input:** Initialization $\hat{\beta}^0$ and swap subset size m.
- **Repeat for** $t = 1, 2, \ldots$**:**
  1. $\beta^t \leftarrow$ Output of cyclic CD initialized from $\hat{\beta}^{t-1}$. Let $S \leftarrow \text{Supp}(\beta^t)$.
  2. Find a feasible solution $\hat{\beta}$ to (4.14) satisfying $P(\hat{\beta}) < P(\beta^t)$.
  3. If Step 2 succeeds, then set $\hat{\beta}^t \leftarrow \hat{\beta}$. Otherwise, if Step 2 fails, **terminate**.

Theorem 4.2 shows that Algorithm 4.2 terminates in a finite number of iterations and provides a description of the resulting solution.

Theorem 4.2. *Let $\{\beta^t\}$ be the sequence of iterates generated by Algorithm 4.2. Then, under Assumption 4.1, β^t converges in finitely many steps to a solution β^* (say). Let $S = Supp(\beta^*)$. Then, β^* satisfies the stationary conditions in* (4.11) *(see Theorem 4.1). In addition, for every $S_1 \subset S$ and $S_2 \subset S^c$ with $|S_1| \leq m$, $|S_2| \leq m$, the solution β^* satisfies the following condition:*

$$P(\beta^*) \leq \min_{\beta} \quad P(\beta^* - U^{S_1}\beta^* + U^{S_2}\beta). \tag{4.15}$$

Algorithm 4.2 improves the solutions obtained from Algorithm 4.1. This observation along with the discussion in Section 4.2.1, establishes the hierarchy of local minima in (4.4). The choice of m in Algorithm 4.2 controls the quality of the local minima returned—larger values of m will lead to solutions with better objectives. For a sufficiently large value of m, Algorithm 4.2 will deliver a global minimizer of Problem (4.3). The computation time of solving Problem (4.14) increases with m. We have observed empirically that small choices of m (e.g., $m = 1$) can lead to a global minimizer of Problem (4.3) even for some challenging high-dimensional problems where the features are highly correlated (see the experiments in Section 4.5).

Problem (4.14) can be formulated using MIP—this is discussed in Section 4.3, where we also present methods to solve it for large problems. The MIP-based framework allows us to (i) obtain good feasible solutions (if they are available) or (ii) certify (via dual bounds) that the current solution cannot be improved by swaps corresponding to size m. Note that due to the restricted search space, solving (4.14) for small values of m can be much easier than solving Problem (4.3) using MIP solvers. A solution β^* obtained from Algorithm 4.2 has an appealing interpretation: being a fixed point of (4.15), β^* cannot be improved by locally perturbing its support. This serves as a certificate describing the quality of the current (locally optimal) solution β^*.

In what follows, we present a fast method to obtain a good solution to Problem (4.14) for the special case of $m = 1$.

Speeding up Combinatorial Search when $m = 1$: For Problem (4.14), we first check if removing variable i, without adding any new variables to the support, improves the objective, i.e., $P(\beta^t - e_i\beta_i^t) < P(\beta^t)$. If the latter inequality holds, we declare a success in Step 2 (Algorithm 2). Otherwise, we find a feasible solution to Problem (4.14) by solving

$$\min_{\beta_j} \quad G(\beta^t - e_i\beta_i^t + e_j\beta_j) \tag{4.16}$$

for every pair $S_1 = \{i\}$ and $S_2 = \{j\}$. Performing the full minimization in (4.16) for every (i, j) can be expensive—so we propose an approximate scheme that is found to work relatively well in our numerical experience. We perform a *few* proximal gradient updates by applying the thresholding operator defined in (4.10) with the choice $\hat{L}_j = L_j$, $\lambda_0 = 0$, and using $\beta_j = 0$ as an initial value, to approximately minimize (4.16)—this helps us identify if the inclusion of coordinate j leads to a success in Step 2.

The method outlined above requires approximately solving Problem (4.16) for $|S|(p - |S|)$ many (i, j)-pairs (in the worst case). This cost can be further reduced if we select j from a small subset of coordinates outside the current support S, i.e., $j \in J \subset S^c$, where $|J| < p - |S|$. We choose J so that it corresponds to the q largest (absolute) values of the gradient $|\nabla_j g(\beta^t - e_i\beta_i^t)|$, $j \in S^c$. As explained in Section 4.A, this choice of J ensures that we

Algorithm 4.3: Fast Heuristic for Local Search when $m = 1$

- **Input**: Restricted set size $q \in \mathbb{Z}_+$ such that $q \leq p - |S|$.
- **For every** $i \in S$:
  1. If $P(\beta^t - e_i\beta_i^t) < P(\beta^t)$ then **terminate** and return $\beta^t - e_i\beta_i^t$.
  2. Compute $\nabla_{S^c} g(\beta^t - e_i\beta_i^t)$ and let J be the set of indices of the q components with the largest values of $|\nabla_j g(\beta^t - e_i\beta_i^t)|$ for $j \in S^c$.
  3. For every $j \in J$:
     Solve $\hat{\beta}_j \in \arg\min_{\beta_j \in \mathbb{R}} G(\beta^t - e_i\beta_i^t + e_j\beta_j)$ by iteratively applying the thresholding operator in (4.10) (with $\hat{L}_i = L_i$ and $\lambda_0 = 0$). If $P(\beta^t - e_i\beta_i^t + e_j\hat{\beta}_j) < P(\beta^t)$, **terminate** and return $\beta^t - e_i\beta_i^t + e_j\hat{\beta}_j$.

search among coordinates $j \in S^c$ that lead to the maximal decrease in the current objective with one step of a proximal coordinate update initialized from $\beta_j = 0$. We summarize the proposed method in Algorithm 4.3.

The cost of applying the thresholding operator in step 3 of Algorithm 4.3 is $\mathcal{O}(n)$.[5] For squared error loss, the cost can be improved to $\mathcal{O}(1)$ by reusing previously computed quantities from CD (see [86] for details). Our numerical experience suggests that using the above heuristic with values of $q \approx 0.05 \times p$ often leads to the same solutions returned by full exhaustive search. Moreover, when some of the features are highly correlated, Algorithm 4.2 with the heuristic above (or solving Problem 4.14 exactly) performs better in terms of variable selection and prediction performance compared to state-of-the-art sparse learning algorithms (see Section 4.5.2 for numerical results).

4.2.3 Solutions for the Cardinality Constrained Formulation

Algorithms 4.1 and 4.2 deliver good solutions for the ℓ_0-penalized problem (4.3). We now discuss how they can be used to obtain solutions to the cardinality constrained version:

$$\min_{\beta \in \mathbb{R}^p} G(\beta) \quad \text{s.t.} \quad \|\beta\|_0 \leq k, \tag{4.17}$$

[5] Assuming that the derivative of $f(\cdot, v)$ w.r.t the first argument, can be computed in $\mathcal{O}(1)$, which is the case for common loss functions that we consider here.

where k controls the support size of β. While the unconstrained formulation (4.3) is amenable to fast CD-based algorithms, some support sizes are often skipped as λ_0 is varied. For example, if we decrease λ_0 to λ_0', the support size of the new solution can differ by more than one, even if λ_0' is taken to be arbitrarily close to λ_0. On the other hand, Formulation (4.17), can typically return a solution with any desired support size,[6] and it may be preferable over the unconstrained formulation in some applications due to its explicit control of the support size.

Suppose we wish to obtain a solution to Problem (4.17) for a support size k, that is not available from a sequence of solutions from (4.3). We propose to apply the IHT algorithm on Problem (4.17), initialized by a solution from Algorithm 4.1 or 4.2. This leads to the following update sequence

$$\beta^{l+1} \leftarrow \underset{\|\beta\|_0 \leq k}{\arg\min} \left\{ \frac{1}{2\tau} \left\| \beta - \left(\beta^l - \tau \nabla g(\beta^l) \right) \right\|_2^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2 \right\}, \tag{4.18}$$

for $l \geq 0$, with initial solution β^0 (available from the ℓ_0-penalized formulation) and $\tau > 0$ is a step size (e.g., see Theorem 4.3).

[16] has shown that greedy CD-like algorithms perform better than IHT for a class of problems similar to (4.17) (without ℓ_1-regularization). However, it is computationally expensive to apply greedy CD methods to (4.17) for the problem sizes we study here. Note that since we initialize IHT with a solution from Problem (4.3), it converges rapidly to a high-quality solution for Problem (4.17).

Theorem 4.3, which follows from [115], establishes that IHT is guaranteed to converge, and provides a characterization of its fixed points.

Theorem 4.3. *Let $\{\beta^l\}$ be a sequence generated by the IHT algorithm updates* (4.18)*. Let L be the Lipschitz constant of $\nabla g(\beta)$ and let $\hat{L} > L$. Then, the sequence $\{\beta^l\}$ converges for*

[6] Exceptions can happen if for a subset $T \subset [p]$ of size k, the minimum of $\beta_T \mapsto G(\beta_T)$ has some coordinates in β_T exactly set to zero.

a step size $\tau = 1/\hat{L}$. *Moreover,* β^* *with support* S *is a fixed point of* (4.18) *iff* $\|\beta^*\|_0 \leq k$,

$$\beta^*_S \in \arg\min_{\beta_S} \; G(\beta_S) \quad \text{and} \quad |\nabla_i g(\beta^*)| \leq \delta_{(k)} \quad \text{for } i \in S^c,$$

where $\delta_j = |\hat{L}\beta^*_j - \nabla_j g(\beta^*)|$ *for* $j \in [p]$ *and* $\delta_{(k)}$ *is the* k*th largest value of* $\{\delta_j\}_1^p$.

Lemma 4.1 shows that if a solution obtained by Algorithm 4.1 or 4.2 has a support size k, then it is a fixed point for the IHT update (4.18).

Lemma 4.1. *Let* $\gamma > 1$ *be a constant and* β^* *with support size* k *be a solution for Problem* (4.3) *obtained by using Algorithm 4.1 or 4.2 with* $\hat{L}_i = \gamma L_i$. *Set* $\hat{L} = \gamma L$ *and* $\tau = 1/\hat{L}$ *in* (4.18). *Then,* β^* *is a fixed point of the update* (4.18).

The converse of Lemma 4.1 is not true, i.e., a fixed point of update (4.18) may not be a fixed point for Algorithm 4.1 or 4.2 (see our earlier discussion around hierarchy 4.4).[7] Thus, we can generally expect the solutions returned by Algorithm 4.1 or 4.2 to be of higher quality than those returned by IHT.

The following summarizes our procedure to obtain a path of solutions for Problem (4.17) (here, we assume that (λ_1, λ_2) are fixed and λ_0 varies):

1. Run Algorithm 4.1 or 4.2 for a sequence of λ_0 values to obtain a regularization path.

2. To obtain a solution to Problem (4.17) with a support size (say k) that is not available in Step 1, we run the IHT updates (4.18). The IHT updates are initialized with a solution from Step 1 having a support size smaller than k.

As the above procedure uses high-quality solutions obtained by Algorithm 4.1 or 4.2 as advanced initializations for IHT, we expect to obtain significant performance benefits as compared to using IHT alone for generating the regularization path. Also, note that if a support size k is available from Step 1, then there is no need to run IHT (see Lemma 4.1).

[7] Assuming that we obtain the same support sizes for both the constrained and unconstrained formulations.

4.2.4 L0Learn: A Fast Toolkit for ℓ_0-regularized Learning

We implemented the algorithms discussed above in `L0Learn`: a fast sparse learning toolkit written in C++ along with an R interface. We currently support the logistic and squared-hinge loss functions,[8] but the toolkit can be expanded and we intend to incorporate additional loss functions in the future. Following [69, 86], we used several computational tricks to speed up the algorithms and improve the solution quality—these include: warm starts, active sets, correlation screening, a (partially) greedy heuristic to cycle through the coordinates, and efficient methods for updating the gradients by exploiting sparsity. Note that Problem (4.3) can lead to the same solution if two values of λ_0 are close to one another. To avoid this issue, we dynamically select a sequence of λ_0-values—this is an extension of an idea appearing in [86] for the least squares loss function.

4.3 Mixed Integer Programming Algorithms

We present MIP formulations and a new scalable algorithm: IGA, for solving Problem (4.3) to optimality. Compared to off-the-shelf MIP solvers (e.g., the commercial solver Gurobi), IGA leads to certifiably optimal solutions in significantly reduced computation times. IGA also applies to the local search problem of Section 4.2.2. We remind the reader that while Algorithm 4.1 (and Algorithm 4.2 for small values of m) leads to good feasible solutions quite fast, it does not deliver certificates of optimality (via dual bounds). The MIP framework can be used to certify the quality of solutions obtained by Algorithm 4.1 or 4.2 and potentially improve upon them.

4.3.1 MIP Formulations

Problem (4.3) admits the following MIP formulation:

$$\min_{\beta,z} \quad \left\{ G(\beta) + \lambda_0 \sum_{i=1}^{p} z_i \right\} \quad \text{s.t.} \qquad |\beta_i| \leq \mathcal{M} z_i, \;\; z_i \in \{0,1\}, \;\; i \in [p] \tag{4.19}$$

[8] This builds upon our earlier functionality for least squares loss [86].

where $\mathcal{M}$ is a constant chosen large enough so that some optimal solution β^* to (4.2) satisfies $\|\beta^*\|_\infty \leq \mathcal{M}$. Algorithms 4.1 and 4.2 can be used to obtain good estimates of $\mathcal{M}$—see [24, 115] for possible alternatives on choosing $\mathcal{M}$. In (4.19), the binary variable z_i controls whether β_i is set to zero or not. If $z_i = 0$ then $\beta_i = 0$, while if $z_i = 1$ then $\beta_i \in [-\mathcal{M}, \mathcal{M}]$ is allowed to be 'free'. For the hinge loss with $q = 1$, the problem is a Mixed Integer Linear Program (MILP). For $q = 2$ and the hinge loss function, (4.19) becomes a Mixed Integer Quadratic Program (MIQP). Similarly, the squared hinge loss leads to a MIQP (for both $q \in \{1, 2\}$). MILPs and MIQPs can be solved by state-of-the-art MIP solvers (e.g., Gurobi and CPLEX) for small-to-moderate sized instances. For other nonlinear convex loss functions such as logistic loss, two approaches are possible: (i) nonlinear branch and bound [20] or (ii) outer-approximation in which a sequence of MILPs is solved until convergence (for example, see [25, 28, 153]).[9] We refer the reader to [105] for a review of related approaches.

The local combinatorial search problem in (4.14) can be cast as a variant of (4.19) with additional constraints; and is given by the following MIP:

$$\min_{\beta, z, \theta} \quad G(\theta) + \lambda_0 \sum_{i=1}^{p} z_i \tag{4.20a}$$

$$\text{s.t.} \quad \theta = \beta^t - \sum_{i \in S} e_i \beta_i^t (1 - z_i) + \sum_{i \in S^c} e_i \beta_i \tag{4.20b}$$

$$|\beta_i| \leq \mathcal{M} z_i, \quad i \in S^c \tag{4.20c}$$

$$\sum_{i \in S} z_i \geq |S| - m \tag{4.20d}$$

$$\sum_{i \in S^c} z_i \leq m \tag{4.20e}$$

$$z_i \in \{0, 1\}, \quad i \in [p] \tag{4.20f}$$

where $S = \text{Supp}(\beta^t)$. The binary variables $z_i, i \in [p]$ perform the role of *selecting* the subsets $S_1 \subset S$ and $S_2 \subset S^c$ (described in (4.14)). Particularly, for $i \in S$, $z_i = 0$ means that $i \in S_1$—i.e., variable i should be removed from the current support. Similarly, for

[9] In this method, one obtains a convex piece-wise linear lower bound to a nonlinear convex problem. In other words, this leads to a polyhedral outer approximation (aka outer approximation) to the epigraph of the nonlinear convex function.

$i \in S^c$, $z_i = 1$ means that $i \in S_2$—i.e., variable j should be added to the support in (4.20). Constraints (4.20d) and (4.20e) enforce $|S_1| \leq m$ and $|S_2| \leq m$, respectively. The constraint in (4.20b) forces the variable θ to be equal to $\beta^t - U^{S_1}\beta^t + U^{S_2}\beta$.

Note that (4.20) has a smaller search space compared to the full formulation in (4.19)—there are additional constraints in (4.20d) and (4.20e), and the number of free continuous variables is $|S^c|$ (as opposed to p in (4.19)). This reduced search space usually leads to notable reductions in the runtime compared to solving Problem (4.19).

4.3.2 Scaling up the MIP via Integrality Generation

While state-of-the-art MIP solvers and outer-approximation based MILP approaches [25, 153] lead to impressive improvements over earlier MIP-based approaches, they often have long run times when solving high-dimensional classification problems with large p and small n (e.g., $p = 50,000$ and $n = 1000$). Our proposed algorithm, IGA, can solve (4.19) to *global* optimality high-dimensional instances that appear to be beyond the capabilities of current MIP-based approaches. Loosely speaking, IGA solves a sequence of MIP-based relaxations or subproblems of Problem (4.19); and exits upon obtaining a global optimality certificate for (4.19). These MIP subproblems are obtained by relaxing a subset of the binary variables z_i to lie within $[0, 1]$, while the remaining variables are retained as binary. Upon solving the relaxed problem and examining the integrality of the continuous z_i's, we create another (tighter) relaxation by allowing more variables to be binary—we continue in this fashion till convergence. The algorithm is formally described below.

The first step in our algorithm is to obtain a good upper bound for Problem (4.19)—this can be obtained by Algorithm 4.1 or 4.2. Let $\mathcal{I}$ denote the corresponding support of this solution. We then consider a relaxation of Problem (4.19) by allowing the binary variables

in $\mathcal{I}^c$ to be continuous:

$$\min_{\beta,z} \quad G(\beta) + \lambda_0 \sum_{i=1}^{p} z_i \tag{4.21a}$$

$$\text{s.t.} \quad |\beta_i| \leq \mathcal{M} z_i, \quad i \in [p] \tag{4.21b}$$

$$z_i \in [0,1], \quad i \in \mathcal{I}^c \tag{4.21c}$$

$$z_i \in \{0,1\}, \quad i \in \mathcal{I}. \tag{4.21d}$$

The relaxation (4.21) is a MIP (and thus nonconvex). The optimal objective of Problem (4.21) is a lower bound to Problem (4.19). In formulation (4.21), we place integrality constraints on $z_i, i \in \mathcal{I}$—all remaining variables z_i, $i \in \mathcal{I}^c$ are continuous. Let β^u, z^u be the solution obtained from the uth iteration of the algorithm. Then, in iteration $(u+1)$, we set $\mathcal{I} \leftarrow \mathcal{I} \cup \{i \mid z_i^u \neq 0\}$ and solve Problem (4.21) (with warm-starting enabled). If at some iteration u, the vector z^u is integral, then solution β^u, z^u must be optimal for Problem (4.19) and the algorithm terminates. We note that along the iterations, we obtain tighter lower bounds on the optimal objective of Problem (4.19). Depending on the available computational budget, we can decide to terminate the algorithm at an early stage with a corresponding lower bound or dual certificate. The algorithm is summarized below:

Algorithm 4.4: Integrality Generation Algorithm (IGA)

- Initialize $\mathcal{I}$ to the support of a solution obtained by Algorithm 4.1 or 4.2.
- **For** $u = 1, 2, \ldots$ perform the following steps till convergence:
  1. Solve the relaxed MIP (4.21) to obtain a solution β^u, z^u.
  2. Update $\mathcal{I} \leftarrow \mathcal{I} \cup \{i \mid z_i^u \neq 0\}$.

As we demonstrate in Section 4.5, Algorithm 4.4 can lead to significant speed-ups: it reduces the time to solve several sparse classification instances from the order of *hours* to *seconds*. This allows us to solve instances with $p \approx 50,000$ and $n \approx 1000$ within reasonable computation times. These instances are much larger than what has been reported in the literature prior to this work (see for example, [153, 25]). A main reason behind the success of Algorithm 4.4 is that Problem (4.21) leads to a solution z with very few nonzero coordinates.

Hence, a small number of indices are added to $\mathcal{I}$ in step 2 of the algorithm. Since $\mathcal{I}$ is typically small, (4.21) can be usually solved significantly faster than the full MIP in (4.19)—the branch-and-bound algorithm has a smaller number of variables to branch on. Lemma 4.2 provides some intuition on why the solutions of (4.21) are sparse.

Lemma 4.2. *Problem* (4.21) *can be equivalently written as:*

$$\min_{\beta, z_{\mathcal{I}}} \quad G(\beta) + \frac{\lambda_0}{\mathcal{M}} \sum_{i \in \mathcal{I}^c} |\beta_i| + \lambda_0 \sum_{i \in \mathcal{I}} z_i \tag{4.22a}$$

$$s.t. \quad |\beta_i| \leq \mathcal{M} z_i, \quad i \in \mathcal{I} \tag{4.22b}$$

$$|\beta_i| \leq \mathcal{M}, \quad i \in \mathcal{I}^c \tag{4.22c}$$

$$z_i \in \{0, 1\}, \quad i \in \mathcal{I}. \tag{4.22d}$$

We have observed empirically that in (4.22), the ℓ_1-regularization term $\sum_{i \in \mathcal{I}^c} |\beta_i|$ encourages sparse solutions, i.e., many of the components $\beta_i, i \in \mathcal{I}^c$ are set to zero. Consequently, the corresponding z_i's in (4.21) are mostly zero at an optimal solution. The sparsity level is controlled by the regularization parameter $\lambda_0/\mathcal{M}$—larger values will lead to more z_i's being set to zero in (4.21). Thus, we expect Algorithm 4.4 to work well when λ_0 is set to a sufficiently large value (to obtain sufficiently sparse solutions). Note that Problem (4.22) is different from the Lasso as it involves additional integrality constraints.

Optimality Gap and Early Termination: Each iteration of Algorithm 4.4 provides us with an improved lower bound to Problem (4.19). This lower bound, along with a good feasible solution (e.g., obtained from Algorithm 4.1 or 4.2), leads to an optimality gap: given an upper bound UB and a lower bound LB, the MIP optimality gap is defined by (UB - LB)/LB. This optimality gap serves as a certificate of (global) optimality. In particular, an early termination of Algorithm 4.4 leads to a solution with an associated certificate of optimality.

Choice of $\mathcal{I}$: The performance of Algorithm 4.4 depends on the initial set $\mathcal{I}$. If the initial $\mathcal{I}$ is close to the support of an optimal solution to (4.19), then our numerical experience suggests

that Algorithm 4.4 can terminate within a few iterations. Moreover, our experiments (see Section 4.5.2) suggest that Algorithm 4.2 can obtain an optimal or a near-optimal solution to Problem (4.19) quickly, leading to a high-quality initialization for $\mathcal{I}$.

In practice, if at iteration u of Algorithm 4.4, the set $\{i \mid z_i^u \neq 0\}$ is large, then we add only a small subset of it to $\mathcal{I}$ (in our implementation, we choose the 10 largest fractional z_i's). Alternatively, while expanding $\mathcal{I}$, we can use a larger cutoff for the fractional z_i's, i.e., we can take the indices $\{i \mid z_i^u \geq \tau\}$ for some value of $\tau \in (0, 1)$. This usually helps in maintaining a small size in $\mathcal{I}$, which allows for solving the MIP subproblem (4.21) relatively quickly.

IGA for Local Combinatorial Search: While the discussion above was centered around the full MIP (4.19)—the IGA framework (and in particular, Algorithm 4.4) extends, in principle, to the local combinatorial search problem in (4.20) for $m \geq 2$.

4.4 Statistical Properties: Error Bounds

We derive non-asymptotic upper bounds on the coefficient estimation error for a family of ℓ_0-constrained classification estimators (this includes the loss functions discussed in Section 4.1.2, among others). For our analysis, we assume that: $(x_i, y_i), i \in [n]$ are i.i.d. draws from an unknown distribution $\mathbb{P}$. Using the notation of Section 4.2, we consider a loss function f and define its population risk $\mathcal{L}(\beta) = \mathbb{E}\left(f\left(\langle x, \beta\rangle; y\right)\right)$, where the expectation is w.r.t. the (population) distribution $\mathbb{P}$. We let β^* denote a minimizer of the risk, that is:

$$\beta^* \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\ \mathcal{L}(\beta) := \mathbb{E}\left(f\left(\langle x, \beta\rangle; y\right)\right). \tag{4.23}$$

In the rest of this section, we let $k = \|\beta^*\|_0$ and $R = \|\beta^*\|_2$—i.e., the number of nonzeros and the Euclidean norm (respectively) of β^*. We assume $R \geq 1$. To estimate β^*, we consider the following estimator:[10]

$$\hat{\beta} \in \underset{\substack{\beta \in \mathbb{R}^p \\ \|\beta\|_0 \leq k,\ \|\beta\|_2 \leq 2R}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} f\left(\langle x_i, \beta\rangle; y_i\right), \tag{4.24}$$

[10]We drop the dependence of $\hat{\beta}$ on R, k for notational convenience.

which minimizes the empirical loss with a constraint on the number of nonzeros in β and a bound on the ℓ_2-norm of β. The ℓ_2-norm constraint in (4.24) makes β^* feasible for Problem (4.24); and ensures that $\hat{\beta}$ lies in a bounded set (which is useful for the technical analysis).

Section 4.4.1 presents the assumptions we need for our analysis—see the works of [142], [150] and [19] for related assumptions in the context of ℓ_1-based classification and quantile regression procedures. In Section 4.4.2, we establish a high probability upper bound on $\|\hat{\beta} - \beta^*\|_2^2$.

4.4.1 Assumptions

We first present some assumptions for establishing the error bounds.

Loss Function. We list our basic assumptions on the loss function.

Assumption 4.2. *The function $t \mapsto f(t; y)$ is non-negative, convex, and Lipschitz continuous with constant L, that is $|f(t_1; y) - f(t_2; y)| \leq L|t_1 - t_2|,\ \forall t_1, t_2$.*

We let $\partial f(t; y)$ denote a subgradient of $t \mapsto f(t; y)$—i.e., $f(t_2; y) - f(t_1; y) \geq \partial f(t_1; y)(t_2 - t_1),\ \forall t_1, t_2$. Note that the hinge loss satisfies Assumption 2 with $L = 1$; and has a subgradient given by $\partial f(t; y) = 1\,(1 - yt \geq 0)\, y$. The logistic loss function satisfies Assumption 2 with $L = 1$; and its subgradient coincides with the gradient.

Differentiability of the Population Risk. The following assumption is on the uniqueness of β^* and differentiability of the population risk $\mathcal{L}$.

Assumption 4.3. *Problem* (4.23) *has a unique minimizer. The population risk $\beta \mapsto \mathcal{L}(\beta)$ is twice continuously differentiable, with gradient $\nabla\mathcal{L}(\beta)$ and Hessian $\nabla^2\mathcal{L}(\beta)$. In particular, the following holds:*

$$\nabla\mathcal{L}(\beta) = \mathbb{E}\left(\partial f\left(\langle x, \beta\rangle; y\right) x\right). \tag{4.25}$$

When f is the hinge loss, [103] discuss conditions under which Assumption 4.3 holds. In particular, Assumption (A1) in [103] requires that the conditional density functions of x

for the two classes are continuous and have finite second moments. Under this assumption, the Hessian $\nabla^2\mathcal{L}(\beta)$ is well defined and continuous in β. Under Assumption (A4) [103], the Hessian is positive definite at an optimal solution, therefore (4.23) has a unique solution. We refer the reader to [150, 168] for discussions pertaining to logistic regression.

Restricted Eigenvalue Conditions. Assumption 4.4 is a *restricted eigenvalue condition* similar to that used in regression problems [29, 44]. For an integer $\ell > 0$, we assume that the quadratic forms associated with the Hessian matrix $\nabla^2\mathcal{L}(\beta^*)$ and the covariance matrix $n^{-1}X^TX$ are respectively lower-bounded and upper-bounded on the set of 2ℓ-sparse vectors.

Assumption 4.4. *Let $\ell > 0$ be an integer. Assumption 4.4(ℓ) is said to hold if there exists constants $\kappa(\ell), \lambda(\ell) > 0$ such that almost surely the following holds:*

$$\kappa(\ell) \leq \inf_{z\neq 0, \|z\|_0 \leq 2\ell} \left\{ \frac{z^T\nabla^2\mathcal{L}(\beta^*)z}{\|z\|_2^2} \right\} \qquad \text{and} \qquad \lambda(\ell) \geq \sup_{z\neq 0, \|z\|_0 \leq 2\ell} \left\{ \frac{\|Xz\|_2^2}{n\|z\|_2^2} \right\}.$$

In the rest of this section, we consider Assumption 4.4 with $\ell = k$. Assumption (A4) in [142] for linear SVM is similar to our Assumption 4.4. For logistic regression, related assumptions appear in the literature, e.g., Assumptions A1 and A2 in [150] (in the form of a dependency and an incoherence condition on the population Fisher information matrix).

Growth condition. As β^* minimizes the population risk, we have $\nabla\mathcal{L}(\beta^*) = 0$. Under the above regularity assumptions and when Assumption 4.4(k) is satisfied, the population risk is lower-bounded by a quadratic function in a neighborhood of β^*. By continuity, we let $r(k)$ denote the maximal radius for which the following lower bound holds:

$$r(k) = \max\left\{ r > 0 \;\middle|\; \mathcal{L}(\beta^* + z) \geq \mathcal{L}(\beta^*) + \frac{1}{4}\kappa(k)\|z\|_2^2 \quad \forall z \quad \text{s.t.} \quad \|z\|_0 \leq 2k, \|z\|_2 \leq r \right\}. \tag{4.26}$$

Below we make an assumption on the growth condition.

Assumption 4.5. *Let $\delta \in (0, 1/2)$. We say that Assumption 4.5(δ) holds if the parameters*

n, p, k, R satisfy:

$$\frac{24L}{\kappa(k)}\sqrt{\frac{\lambda(k)}{n}\left(k\log\left(Rp/k\right)+\log\left(1/\delta\right)\right)} \leq r(k),$$

and the following holds: $(k/n)\log(p/k) \leq 1$ *and* $7ne \leq 3L\sqrt{\lambda(k)}p\log\left(p/k\right)$.

Assumption 4.5 is similar to the scaling conditions in [150][Theorem 1] for logistic regression. [19] also makes use of a growth condition for their analysis in ℓ_1-sparse quantile regression problems. Note that although Assumption 4.5 is not required to prove Theorem 4.4 (which will be presented in Section 4.4.2), we will need it to derive the error bound of Theorem 4.5. We now proceed to derive an upper bound on coefficient estimation error.

4.4.2 Main Result

We first show (in Theorem 4.4) that the loss function f satisfies a form of restricted strong convexity [131] around β^*, a minimizer of (4.23).

Theorem 4.4. *Let* $h = \hat{\beta} - \beta^*$, $\delta \in (0, 1/2)$, *and* $\tau = 6L\sqrt{\frac{\lambda(k)}{n}\left(k\log\left(Rp/k\right)+\log\left(1/\delta\right)\right)}$. *If Assumptions 4.2, 4.3, 4.4(k) and 4.5(δ) are satisfied, then with probability at least* $1-\delta$, *the following holds:*

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} f\left(\langle x_i, \beta^* + h\rangle; y_i\right) - \frac{1}{n}\sum_{i=1}^{n} f\left(\langle x_i, \beta^*\rangle; y_i\right) \\ \geq \frac{1}{4}\kappa(k)\left\{\|h\|_2^2 \wedge r(k)\|h\|_2\right\} - \tau\|h\|_2 \vee \tau^2.\end{aligned} \tag{4.27}$$

Theorem 4.4 is used to derive the error bound presented in Theorem 4.5, which is the main result in this section.

Theorem 4.5. *Let* $\delta \in (0, 1/2)$ *and assume that Assumptions 4.2, 4.3, 4.4(k) and 4.5(δ) hold. Then the estimator* $\hat{\beta}$ *defined as a solution of Problem* (4.24) *satisfies with probability at least* $1-\delta$*:*

$$\|\hat{\beta} - \beta^*\|_2^2 \lesssim L^2\lambda(k)\widetilde{\kappa}^2\left(\frac{k\log(Rp/k)}{n} + \frac{\log(1/\delta)}{n}\right) \tag{4.28}$$

where, $\widetilde{\kappa} = \max\{1/\kappa(k), 1\}$.

In (4.28), the symbol "$\lesssim$" stands for "$\leq$" up to a universal constant. The proof of Theo-

rem 4.5 is presented in Appendix 4.A. The rate appearing in Theorem 4.5, of the order of $k/n \log(p/k)$, is the best known rate for a (sparse) classifier; and this coincides with the optimal scaling in the case of regression. In comparison, [142] and [150] derived a bound scaling as $k/n \log(p)$ for ℓ_1-regularized SVM (with hinge loss) and logistic regression, respectively. Note that the bound in Theorem 4.5 holds for any sufficiently small $\delta > 0$. Consequently, by integration, we obtain the following result in expectation.

Corollary 4.1. *Suppose Assumptions 4.2, 4.3, 4.4(k) hold true and Assumption 4.5(δ) is true for δ small enough, then:*

$$\mathbb{E}\|\hat{\beta} - \beta^*\|_2^2 \lesssim L^2 \lambda(k) \widetilde{\kappa}^2 \frac{k \log(Rp/k)}{n},$$

where $\widetilde{\kappa}$ is defined in Theorem 4.5.

4.5 Experiments

In this section, we compare the statistical and computational performance of our proposed algorithms versus the state of the art, on both synthetic and real data sets.

4.5.1 Experimental Setup

Data Generation. For the synthetic data sets, we generate a multivariate Gaussian data matrix $X_{n \times p} \sim \text{MVN}(0, \Sigma)$ and a sparse vector of coefficients $\beta^\dagger$ with $k^\dagger$ nonzero entries, such that $\beta_i^\dagger = 1$ for $k^\dagger$ equi-spaced indices $i \in [p]$. Every coordinate y_i of the outcome vector $y \in \{-1, 1\}^n$ is then sampled independently from a Bernoulli distribution with success probability: $P(y_i = 1|x_i) = (1 + \exp(-s\langle \beta^\dagger, x_i \rangle))^{-1}$, where x_i denotes the i-th row of X, and s is a parameter that controls the signal-to-noise ratio. Specifically, smaller values of s increase the variance in the response y, and when $s \to \infty$ the generated data becomes linearly separable.

Algorithms and Tuning. We compare our proposal, as implemented in our package `L0Learn`, with: (i) ℓ_1-regularized logistic regression (`glmnet` package [69]), (ii) MCP-regularized

logistic regression (`ncvreg` package [41]), and (iii) two packages for sparsity constrained minimization (based on hard thresholding): `GraSP` [13] and `NHTP` [192]. For `GraSP` and `NHTP`, we use cardinality constrained logistic regression with ridge regularization—that is, we optimize Problem (4.17) with $\lambda_1 = 0$. Tuning is done on a separate validation set under the fixed design setting, i.e., we use the same features used for training but a new outcome vector y' (independent of y). The tuning parameters are selected to minimize the loss function on the validation set (e.g., for regularized logistic regression, we minimize the unregularized negative log-likelihood).

We use ℓ_0-ℓ_q as a shorthand to denote the penalty $\lambda_0\|\beta\|_0 + \lambda_q\|\beta\|_q^q$ for $q \in \{1, 2\}$. For all penalties that involve 2 tuning parameters—i.e., ℓ_0-ℓ_q (for $q \in \{1, 2\}$), MCP, `GraSP`, and `NHTP`, we sweep the parameters over a two-dimensional grid. For our penalties, we choose 100 λ_0 values as described in Section 4.2.4. For `GraSP` and `NHTP`, we sweep the number of nonzeros between 1 and 100. For ℓ_0-ℓ_1, and we choose a sequence of 10 λ_1-values in $[a, b]$, where a corresponds to a zero solution and $b = 10^{-4}a$. Similarly, for ℓ_0-ℓ_2, `GraSP`, and `NHTP`, we choose 10 λ_2 values between 10^{-4} and 100 for the experiment in Section 4.5.2; and between 10^{-8} and 10^{-4} for that in Section 4.5.3. For MCP, the sequence of 100 λ values is set to the default values selected by `ncvreg`, and we vary the second parameter γ over 10 values between 1.5 and 25. For the ℓ_1-penalty, the grid of 100 λ values is set to the default sequence chosen by `glmnet`.

Performance Measures. We use the following measures to evaluate the performance of an estimator $\hat{\beta}$:

- **AUC**: The area under the curve of the ROC plot.
- **Recovery F1 Score**: This is the F1 score for support recovery, i.e., it is the harmonic mean of precision and recall: F1 Score $= 2PR/(P + R)$, where P is the precision given by $|\text{Supp}(\hat{\beta}) \cap \text{Supp}(\beta^\dagger)|/|\text{Supp}(\hat{\beta})|$, and R is the recall given by $|\text{Supp}(\hat{\beta}) \cap \text{Supp}(\beta^\dagger)|/|\text{Supp}(\beta^\dagger)|$. An F1 Score of 1 implies full support recovery; and a value of zero implies that the supports of the true and estimated coefficients have no overlap.

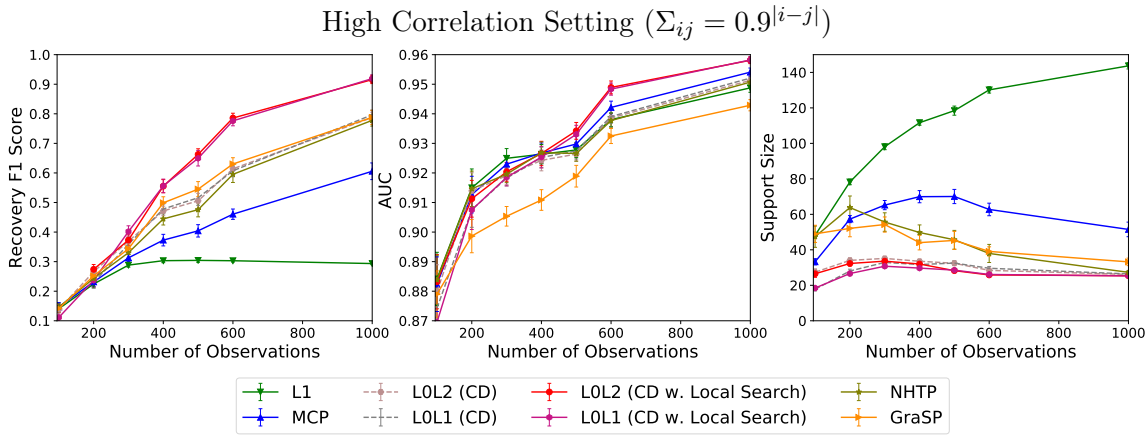


Figure 4-1: Performance for varying $n \in [100, 10^3]$ and $\Sigma_{ij} = 0.9^{|i-j|}, p = 1000, k^\dagger = 25, s = 1$. In this high-correlation setting, our proposed algorithm (using local search) for the ℓ_0-ℓ_q (for both $q \in \{1, 2\}$) penalized estimators—denoted as L0L2 (CD w. Local Search) and L0L1 (CD w. Local Search) in the figure—seems to outperform state-of-the-art methods (MCP, ℓ_1, `GraSP` and `NHTP`) in terms of both variable selection and prediction. The variable selection performance improvement (higher F1 score and smaller support size) is more notable than AUC. Local search with CD shows benefits compared to its variants that do not employ local search (denoted as L0L1 (CD) and L0L2 (CD) in the figure).

- **Support Size**: The number of nonzeros in $\hat{\beta}$.
- **False Positives**: This is equal to $|\text{Supp}(\hat{\beta}) \setminus \text{Supp}(\beta^\dagger)|$.

4.5.2 Performance for Varying Sample Sizes

In this experiment, we fix p and vary the number of observations n to study its effect on the performance of the different algorithms and penalties. We hypothesize that when the statistical setting is difficult (e.g., features are highly correlated and/or n is small), good optimization algorithms for ℓ_0-regularized problems lead to estimators that can significantly outperform estimators obtained from convex regularizers and common (heuristic) algorithms for nonconvex regularizers. To demonstrate our hypothesis, we perform experiments on the following data sets:

- **High Correlation**: $\Sigma_{ij} = 0.9^{|i-j|}, p = 1000, k^\dagger = 25, s = 1$
- **Medium Correlation**: $\Sigma_{ij} = 0.5^{|i-j|}, p = 1000, k^\dagger = 25, s = 1$

For each of the above settings, we use the logistic loss function and consider 20 random

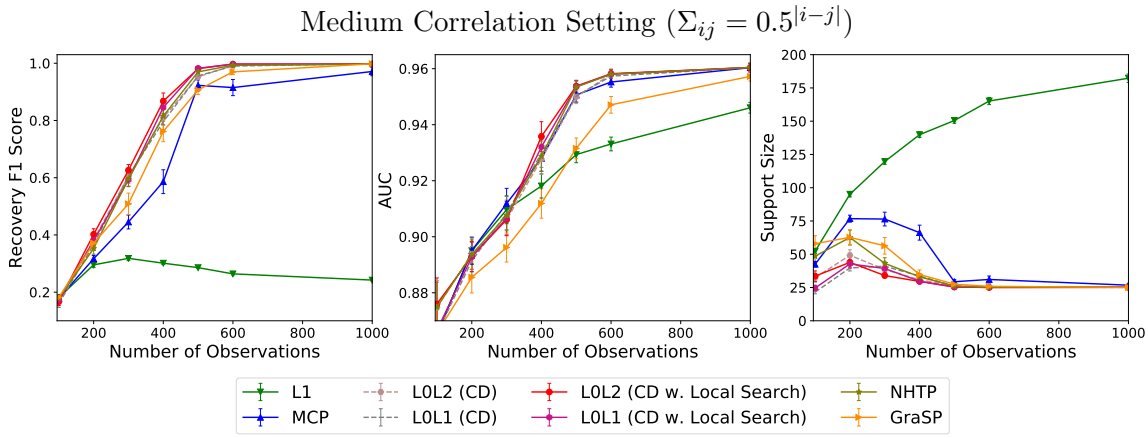


Figure 4-2: Performance for varying n and $\Sigma_{ij} = 0.5^{|i-j|}, p = 1000, k^{\dagger} = 25, s = 1$. In contrast to Figure 4-1, this is a medium-correlation setting. Once again Algorithm 4.2 (CD with local search) for the ℓ_0-ℓ_1 and ℓ_0-ℓ_2 penalties seems to perform quite well in terms of variable selection and prediction performance, though its improvement over the other methods appears to be less prominent compared to the high-correlation setting in Figure 4-1. In this example, local search does not seem to offer much improvement over pure CD (i.e., Algorithm 4.1).

repetitions. We report the averaged results for the high correlation setting in Figure 4-1 and for the medium correlation setting in Figure 4-2.

Figure 4-1 shows that in the high correlation setting, Algorithm 4.2 (with $m = 1$) for the ℓ_0-ℓ_2 penalty and the ℓ_0-ℓ_1 penalty—denoted by L0L2 (CD w. Local Search) and L0L1 (CD w. Local Search) (respectively) in the figure—achieves the best support recovery and AUC across different sample sizes n. We note that in this example, the best F1 score falls below 0.8 for n smaller than 600—suggesting that none of the algorithms can do full support recovery. For larger values of n, the difference between Algorithm 4.2 and others become more pronounced in terms of F1 score, suggesting an important edge in terms of variable selection performance. Moreover, our algorithms select the smallest support size (i.e., the most parsimonious model) for all n. In contrast, the ℓ_1 penalty (i.e., L1) selects significantly larger support sizes (exceeding 100 in some cases) and suffers in terms of support recovery. It is also worth mentioning that the other ℓ_0-based algorithms, i.e., Algorithm 4.1, `NHTP` and `GraSP`, outperform MCP and L1 in terms of support recovery (F1 score) and support sizes. The performance of `NHTP` and `GraSP` appears to be similar in terms of support sizes and F1 score, though `NHTP` appears to be performing better compared to `GraSP` in terms of AUC.

In Figure 4-2, for the medium correlation setting, we see that Algorithms 4.1 and 4.2 perform similarly: local search (with CD) performs similar to CD (without local search)—Algorithm 4.1 can recover the correct support with around 500 observations. In this case, the performance of MCP becomes similar to the ℓ_0-ℓ_q penalized estimators in terms of AUC, though there are notable differences in terms of variable selection properties. Compared to Figure 4-1, we observe that ℓ_1 performs better in this example, but still appears to be generally outperformed by other algorithms in terms of all measures. We also observe that `NHTP`'s performance is comparable to the best methods in terms of F1 score and AUC. However, it results in models that are more dense compared to L0L1 and L0L2 (both Algorithms 4.1 and 4.2). That being said, we will see in Section 4.5.5 that in terms of running time, Algorithm 4.1 has a significant edge over `NHTP`. Furthermore, for larger problem instances (Section 4.5.3) the performance of `NHTP` suffers in terms of variable selection properties compared to the methods we propose in this chapter. `NHTP` appears to perform better than `GraSP` across all metrics in this setting.

Figures 4-1 and 4-2 suggest that when the correlations are high, local search with ℓ_0 can significantly outperform competing methods (especially, in terms of variable selection). When the correlations are medium to low, the differences across the different nonconvex methods become less pronounced. The performance of nonconvex penalized estimators (especially, the ℓ_0-based estimators) is generally better than ℓ_1-based estimators in these examples.

4.5.3 Performance on Larger Instances

We now study the performance of the different algorithms for some large values of p under the following settings:

- **Setting 1**: $\Sigma = I, n = 1000, p = 50,000, s = 1000$, and $k^\dagger = 30$
- **Setting 2**: $\Sigma_{ij} = 0.3$ for $i \neq j$, $\Sigma_{ii} = 1$, $n = 1000, p = 10^5, s = 1000$, and $k^\dagger = 20$.

Table 4.2 reports the results available from Algorithms 4.1 and 4.2 ($m = 1$) versus the other methods, for different loss functions. In Settings 1 and 2 above, all methods achieve an AUC of 1 (approximately), and the main differences across the methods lie in variable selection.

	Setting 1		Setting 2	
Penalty/Loss	FP	$\|\hat{\beta}\|_0$	FP	$\|\hat{\beta}\|_0$
ℓ_0-ℓ_2/Logistic (Algorithm 4.1)	0.0 ± 0.0	30.0 ± 0.0	21.6 ± 0.9	26.2 ± 0.5
ℓ_0-ℓ_1/Logistic (Algorithm 4.1)	0.0 ± 0.0	30.0 ± 0.0	11.2 ± 0.7	14.8 ± 0.3
ℓ_0-ℓ_2/Logistic (Algorithm 4.2)	0.0 ± 0.0	30.0 ± 0.0	11.5 ± 0.4	14.6 ± 0.3
ℓ_0-ℓ_1/Logistic (Algorithm 4.2)	0.0 ± 0.0	30.0 ± 0.0	11.2 ± 0.4	14.5 ± 0.3
ℓ_0-ℓ_2/Sq. Hinge (Algorithm 4.1)	2.8 ± 0.3	32.8 ± 0.3	23.9 ± 0.7	27.0 ± 0.4
ℓ_1/Logistic	617.2 ± 8.3	647.2 ± 8.3	242.2 ± 4.8	256.9 ± 5.3
MCP/Logistic	0.0 ± 0.0	30.0 ± 0.0	80.1 ± 10.9	91.5 ± 12.1
NHTP/Logistic	44.7 ± 5.6	74.7 ± 5.6	92.5 ± 0.6	100.0 ± 0.0
GraSP/Logistic	50.5 ± 5.6	80.5 ± 5.6	45.3 ± 3.0	54.4 ± 2.8

Table 4.2: Variable selection performance for different penalty and loss combinations, under high-dimensional settings. FP refers to the number of false positives. We consider ten repetitions and report the averages (and standard errors) across the repetitions.

For Setting 1, both Algorithms 4.1 and 4.2 applied to the logistic loss; and `ncvreg` for the MCP penalty (with logistic loss) correctly recover the support. In contrast, ℓ_1 captures a large number of false positives, leading to large support sizes. Both `NHTP` and `GraSP` have a considerable number of false positives and result in large support sizes.

In Setting 2, none of the algorithms correctly recovered the support. This setting is more difficult than Setting 1 as it has higher correlation and a larger p. In Setting 2, we observe that CD with local search can have an edge over plain CD (see, logistic loss with ℓ_0-ℓ_2 penalty). In terms of small FP and compactness of model size, CD with local search appears to be the winner, with Algorithm 4.1 being quite close. Both Algorithms 4.1 and 4.2 appear to work better compared to earlier methods in this setting. We note that in Setting 2, our proposed algorithms select supports with roughly 3 times fewer nonzeros than the MCP penalized problem, and 10 times fewer nonzeros than those delivered by the ℓ_1-penalized problem. For both settings, our proposed methods offer important improvements over `NHTP` and `GraSP` as well.

4.5.4 Performance on Real Data Sets

We compare the performance of ℓ_1 with ℓ_0-ℓ_q ($q \in \{1, 2\}$) regularization, using the logistic loss function.[11] We consider the following three binary classification data sets taken from the NIPS 2003 Feature Selection Challenge [77]:

- **Arcene**: This data set is used to identify cancer vs non-cancerous patterns in mass-spectrometric data. The data matrix is dense with $p = 10,000$ features. We used 140 observations for training and 40 observations for testing.

- **Dorothea**: This data set is used to distinguish active chemical compounds in a drug. The data matrix is sparse with $p = 100,000$ features. We used 805 observations for training and 230 observations for testing.

- **Dexter:** The task here is to identify text documents discussing corporate acquisitions. The data matrix is sparse with $p = 20,000$ features. We used 420 observations for training and 120 observations for testing.

We obtained regularization paths for ℓ_1 using `glmnet` and for ℓ_0-ℓ_2 and ℓ_0-ℓ_1 using both Algorithm 4.1 and Algorithm 4.2 (with $m = 1$). In Figure 4-3, we plot the support size versus the test AUC for ℓ_1 and Algorithm 4.2 (with $m = 1$) using ℓ_0-ℓ_2. To avoid overcrowded plots, the results for our other algorithms and penalties are presented in Appendix 4.B. For the Arcene data set, ℓ_0-ℓ_2 with $\lambda_2 = 1$ (i.e., the red curve) outperforms ℓ_1 (the green curve) for most of the support sizes, and it reaches a peak AUC of 0.9 whereas ℓ_1 does not exceed 0.84. The other choices of λ_2 also achieve a higher peak AUC than ℓ_1 but do not uniformly outperform it. A similar pattern occurs for the Dorothea data set, where ℓ_0-ℓ_2 with $\lambda_2 = 1$ achieves a peak AUC of 0.9 at around 100 features, whereas ℓ_1 needs around 160 features to achieve the same peak AUC. For the Dexter data set, ℓ_0-ℓ_2 with $\lambda_2 = 10^{-2}$ (the blue curve) achieves a peak AUC of around 0.98 using less than 10 features, whereas ℓ_1 requires

[11] We tried MCP regularization using `ncvreg`, however it ran into convergence issues and did not terminate in over an hour. We also tried NHTP, but it ran into convergence issues and took more than 2 hours to solve for a single solution in the regularization path. Also, note that based on our experiment in Section 4.5.5, GraSP is slower than NHTP and cannot handle such high-dimensional problems. Therefore, we do not include MCP, NHTP and GraSP in this experiment.

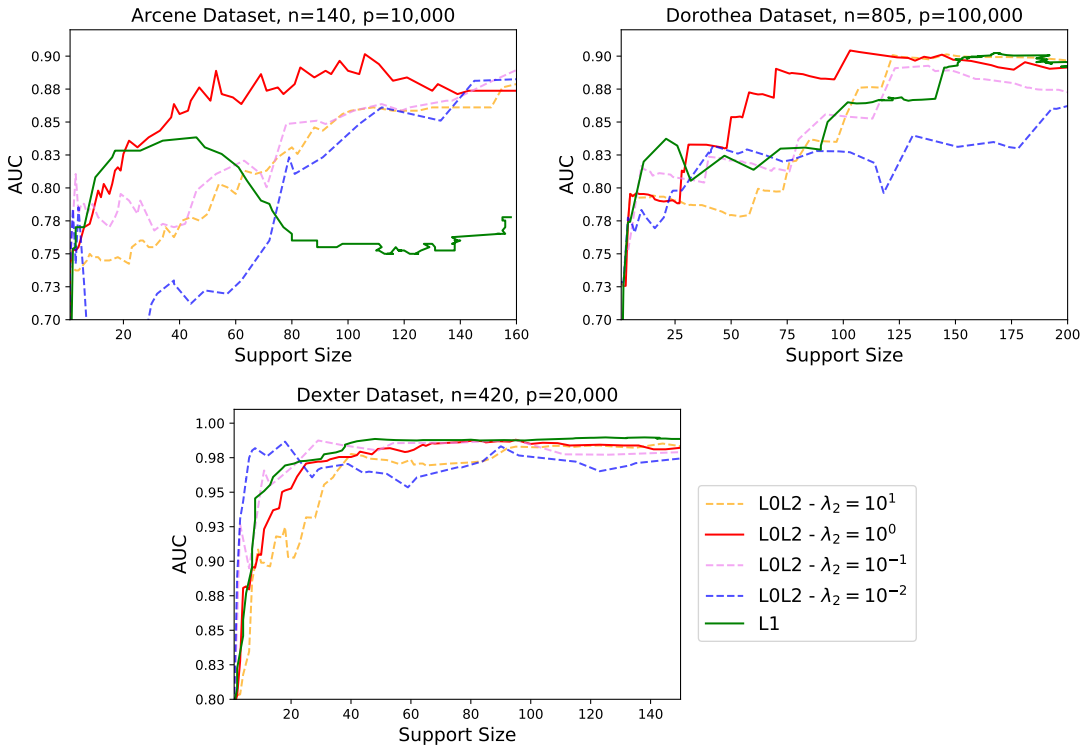


Figure 4-3: Plots of the AUC versus the support size for the Arcene, Dorothea, and Dexter data sets. The green curves correspond to logistic regression with ℓ_1 regularization. The other curves correspond to logistic regression with ℓ_0-ℓ_2 regularization (using Algorithm 4.2 with $m = 1$) for different values of λ_2 (see legend).

around 40 features to achieve a similar AUC. In this case, larger choices of λ_2 do not lead to any significant gains in AUC. This phenomenon is probably due to a higher signal in this data set compared to the previous two. Overall, we conclude that for all the three data sets, Algorithm 4.2 for ℓ_0-ℓ_2 can achieve higher AUC-values with (much) smaller support sizes.

4.5.5 Timings

In this section, we compare the running time of our algorithms versus several state-of-the-art algorithms for sparse classification.

Obtaining good solutions: Upper Bounds

We study the running time of fitting sparse logistic regression models, using the following packages: our package `L0Learn`, `glmnet`, `ncvreg`, and `NHTP`.[12] In `L0Learn`, we consider both Algorithm 4.1 (denoted by L0Learn 1) and Algorithm 4.2 with $m = 1$ (denoted by L0Learn 2). We generate synthetic data as described in Section 4.5.1, with $n = 1000, s = 1, \Sigma = I$, $k^\dagger = 5$, and we vary p. For all toolkits except `NHTP`,[13] we set the convergence threshold to 10^{-6} and solve for a regularization path with 100 solutions. For `L0Learn`, we use a grid of λ_0-values varying between $\lambda_0^{\max}$ (the value which sets all coefficients to zero) and $0.001\lambda_0^{\max}$. For `L0Learn` and `NHTP`, we set $\lambda_2 = 10^{-7}$. For `glmnet` and `ncvreg`, we compute a path using the default choices of tuning parameters. The experiments were carried out on a machine with a 6-core Intel Core i7-8750H processor and 16GB of RAM, running macOS 10.15.7, R 4.0.3 (with vecLib's BLAS implementation), and MATLAB R2020b. The running times are reported in Figure 4-4.

Figure 4-4 indicates that `L0Learn` (Algorithm 4.1) and `glmnet` achieve the fastest runtimes across the p range. For $p > 40,000$, `L0Learn` (Algorithm 4.1) runs slightly faster `glmnet`. `L0Learn` (Algorithm 4.2) is slower due to the local search, but it can obtain solutions in reasonable times, e.g., less than a minute for 100 solutions at $p = 10^5$. It is important to note that the toolkits are optimizing for different objective functions, and the speed-ups in `L0Learn` are partly due to the nature of ℓ_0-regularization which selects fewer nonzeros compared to those available from ℓ_1 or MCP regularization.

Timings for MIP Algorithms: Global Optimality Certificates

We compare the running time to solve Problem (4.19) by Algorithm 4.4 (IGA) versus solving (4.19) directly i.e., without using IGA. We consider the hinge loss and take $q = 2$. We use Gurobi's MIP solver for our experiments. We set $\Sigma = I$ and $k^\dagger = 5$ as described in Section 4.5.1. We then generate $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ and set $y_i = \text{sign}(x_i'\beta^\dagger + \epsilon_i)$,

[12] We also considered `GraSP`, but we found it to be several orders of magnitude slower than the other toolkits, e.g., it takes 3701 seconds at $p = 10^5$. All the other toolkits require less than 70 seconds at $p = 10^5$. Therefore, we did not include `GraSP` in our timing experiments.

[13] `NHTP` does not have a parameter to control the tolerance for convergence.

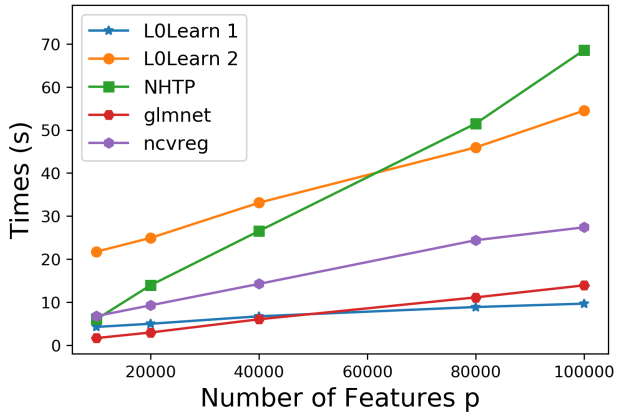


Figure 4-4: Runtimes to obtain good feasible solutions: Time (s) for obtaining a regularization path (with 100 solutions) for different values of p (as discussed in Section 4.5.5). L0Learn 1 runs Algorithm 4.1 (CD), while L0Learn 2 runs Algorithm 4.2 (CD with Local Search).

Tookit/p	10000	20000	30000	40000	50000
IGA (our proposal)	35	80	117	169	297*
Gurobi	4446	21817	-	-	-

Table 4.3: Runtimes to certify global optimality: Time(s) for solving an ℓ_0-regularized problem with a hinge loss function, to optimality. "-" denotes that the algorithm does not terminate in a day. "*" indicates that the algorithm is terminated with a 0.05% optimality gap.

where the signal-to-noise ratio SNR=$\text{Var}(X\beta^\dagger)/\sigma^2 = 10$. We set $n = 1000$ and vary $p \in \{10^4, 2\times10^4, 3\times10^4, 4\times10^4, 5\times10^4\}$. For a fixed $\lambda_2 = 10$, λ_0 is chosen such that the final (optimal) solution has 5 nonzeros. The parameter $\mathcal{M}$ is set to $1.2\|\tilde{\beta}\|_\infty$, where $\tilde{\beta}$ is the warm start obtained from Algorithm 4.2. We note that in all cases, the optimal solution recovers the support of the true solution $\beta^\dagger$.

For this experiment, we use a machine with a 12-core Intel Xeon E5 @ 2.7 GHz and 64GB of RAM, running OSX 10.13.6 and Gurobi v8.1. The timings are reported in Table 4.3. The results indicate significant speed-ups. For example, for $p = 20,000$ our algorithm terminates in 80 seconds whereas Gurobi takes 6 hours to solve (4.19) (to global optimality). For larger values of p, Gurobi cannot terminate in a day, while our algorithm can terminate to optimality in few minutes. It is worth noting that, empirically we observe IGA can attain low running times when sparse solutions are desired (< 30 nonzeros in our experience). This observation also applies to the state-of-the-art MIP solvers for sparse regression, e.g., see

[24, 28, 90]. For denser solutions, IGA can still be useful for obtaining optimality gaps, but certifying optimality with a very small optimality gap is expected to take longer.

4.6 Conclusion

We considered the problem of linear classification regularized with a combination of the ℓ_0 and ℓ_q (for $q \in \{1, 2\}$) penalties. We developed both approximate and exact algorithms for this problem. Our approximate algorithms are based on coordinate descent and local combinatorial search. We established convergence guarantees for these algorithms and demonstrated empirically that they can run in times comparable to the fast ℓ_1-based solvers. Our exact algorithm can solve to optimality high-dimensional instances with $p \approx 50,000$. This scalability is achieved through the novel idea of integrality generation, which solves a sequence of mixed integer programs with a small number of binary variables, until converging to a globally optimal solution. We also established new estimation error bounds for a class of ℓ_0-regularized classification problems and showed that these bounds compare favorably with the best known bounds for ℓ_1 regularization. We carried out experiments on both synthetic and real datasets with p up to 10^5. The results demonstrate that our ℓ_0-based combinatorial algorithms can have a significant statistical edge (in terms of variable selection and prediction) compared to state-of-the-art methods for sparse classification, such as those based on ℓ_1 regularization or simple greedy procedures for ℓ_0 regularization.

4.A Appendix: Proofs and Technical Details

Proof of Theorem 4.1

We first present Lemma 4.3, which establishes that after updating a single coordinate, there is a sufficient decrease in the objective function. The result of this Lemma will be used in the proof of the theorem.

Lemma 4.3. *Let $\{\beta^l\}$ be the sequence of iterates generated by Algorithm 4.1. Then, the*

following holds for any l and $i = 1 + (l \mod p)$:

$$P(\beta^l) - P(\beta^{l+1}) \geq \frac{\hat{L}_i - L_i}{2}(\beta_i^{l+1} - \beta_i^l)^2. \tag{4.29}$$

Proof. Consider some $l \geq 0$ and fix $i = 1 + (l \mod p)$ (this corresponds to the coordinate being updated via CD). Applying (4.6) to β^l and β^{l+1} and adding $\psi(\beta^{l+1})$ to both sides, we have:

$$P(\beta^{l+1}) \leq g(\beta^l) + (\beta_i^{l+1} - \beta_i^l)\nabla_i g(\beta^l) + \frac{L_i}{2}(\beta_i^{l+1} - \beta_i^l)^2 + \psi(\beta^{l+1}).$$

By writing $\frac{L_i}{2}(\beta_i^{l+1} - \beta_i^l)^2$ as the sum of two terms: $\frac{L_i - \hat{L}_i}{2}(\beta_i^{l+1} - \beta_i^l)^2 + \frac{\hat{L}_i}{2}(\beta_i^{l+1} - \beta_i^l)^2$ and regrouping the terms we get:

$$P(\beta^{l+1}) \leq \widetilde{P}_{\hat{L}_i}(\beta^{l+1}; \beta^l) + \frac{L_i - \hat{L}_i}{2}(\beta_i^{l+1} - \beta_i^l)^2, \tag{4.30}$$

where, in the above equation, we used the definition of $\widetilde{P}$ from (4.8). Recall from the definition of Algorithm 4.1 that $\beta_i^{l+1} \in \arg\min_{\beta_i} \widetilde{P}_{\hat{L}_i}(\beta_1^l, \ldots, \beta_i, , \ldots, \beta_p^l; \beta^l)$ which implies that $\widetilde{P}_{\hat{L}_i}(\beta^l; \beta^l) \geq \widetilde{P}_{\hat{L}_i}(\beta^{l+1}; \beta^l)$. But $\widetilde{P}_{\hat{L}_i}(\beta^l; \beta^l) = P(\beta^l)$ (this follows directly from (4.8)). Therefore, we have $P(\beta^l) \geq \widetilde{P}_{\hat{L}_i}(\beta^{l+1}; \beta^l)$. Plugging the latter inequality into (4.30) and rearranging the terms, we arrive to the result of the lemma. □

Next, we present the proof of Theorem 4.1.

Proof.

- **Part 1**: We will show that $\text{Supp}(\beta^l) \neq \text{Supp}(\beta^{l+1})$ cannot happen infinitely often. Suppose for some l we have $\text{Supp}(\beta^l) \neq \text{Supp}(\beta^{l+1})$. Then, for $i = 1 + (l \mod p)$, one of the following two cases must hold: (I) $\beta_i^l = 0 \neq \beta_i^{l+1}$ or (II) $\beta_i^l \neq 0 = \beta_i^{l+1}$. Let us consider case (I). Recall that the minimization step in Algorithm 4.1 is done using the thresholding operator defined in (4.10). Since $\beta_i^{l+1} \neq 0$, (4.10) implies that $|\beta_i^{l+1}| \geq \sqrt{\frac{2\lambda_0}{\hat{L}_i + 2\lambda_2}}$. Plugging the latter bound into the result of Lemma 4.3, we arrive to

$$P(\beta^l) - P(\beta^{l+1}) \geq \frac{\hat{L}_i - L_i}{\hat{L}_i + 2\lambda_2}\lambda_0, \tag{4.31}$$

where the r.h.s. of the above is positive, due to the choice $\hat{L}_i > L_i$. The same argument can be used for case (II) to arrive to (4.31). Thus, whenever the support changes, (4.31) applies, and consequently the objective function decreases by a constant value. Therefore, the support cannot change infinitely often (as $P(\beta)$ is bounded below).

- **Part 2**: First, we will show that under Assumption 4.1, the function $\beta_S \mapsto G(\beta_S)$ is strongly convex. This holds for the first case of Assumption 4.1, i.e., when $\lambda_2 > 0$. Next, we will consider the second case of Assumption 4.1 which states that $P(\beta^0) < \lambda_0 u$ and $g(\beta)$ is strongly convex when restricted to a support of size at most u. Since Algorithm 4.1 is a descent algorithm, we get $P(\beta^l) < \lambda_0 u$ for any $l \geq 0$. This implies $\|\beta^l\|_0 < u$ for any $l \geq 0$, and thus $|S| < u$. Consequently, the function $\beta_S \mapsto g(\beta_S)$ is strongly convex (and so is $\beta_S \mapsto G(\beta_S)$).

  After support stabilization (from Part 1), the iterates generated by Algorithm 4.1 are the same as those generated by CD for minimizing the strongly convex function $G(\beta_S)$, which is guaranteed to converge (e.g., see [22]). Therefore, we conclude that $\{\beta^l\}$ converges to a stationary solution β^* satisfying $\beta^*_S \in \arg\min_{\beta_S} G(\beta_S)$ and $\beta^*_{S^c} = 0$.

  Finally, we will prove the two inequalities in (4.11). Fix some $i \in S$. Then, from the definition of the thresholding operator in (4.10), the following holds for every $l > N$ (i.e., after support stabilization) at which coordinate i is updated:

$$|\beta^l_i| \geq \sqrt{\frac{2\lambda_0}{\hat{L}_i + 2\lambda_2}}. \tag{4.32}$$

  Taking the limit as $l \to \infty$ we arrive to the first inequality in (4.11), which also implies that $\text{Supp}(\beta^*) = S$. Now let us fix some $i \in S^c$. For every $l > N$ at which coordinate i is updated, we have $\beta^l_i = 0$ and the following condition holds by the definition of the thresholding operator in (4.10):

$$\frac{\hat{L}_i}{\hat{L}_i + 2\lambda_2}\Big(\Big|\frac{\nabla_i g(\beta^l)}{\hat{L}_i}\Big| - \frac{\lambda_1}{\hat{L}_i}\Big) < \sqrt{\frac{2\lambda_0}{\hat{L}_i + 2\lambda_2}}. \tag{4.33}$$

Taking the limit as $l \to \infty$ in the above and simplifying, we arrive to the second inequality in (4.11).

- **Part 3**: The support stabilization on S (from Part 1) and the fact that $\beta^l \to \beta^*$ where $\text{Supp}(\beta^*) = S$ (from Part 2), directly imply that $\text{sign}(\beta^l_S)$ stabilizes in a finite number of iterations. That is, there exists an integer N' and a vector $t \in \{-1, 1\}^{|S|}$ such that $\text{sign}(\beta^l_S) = t$ for all $l \geq N'$. Therefore, for $l \geq N'$, the iterates of Algorithm 4.1 are the same as those generated by running coordinate descent to minimize the continuously differentiable function:

$$g(\beta) + \lambda_0|S| + \lambda_1(\sum_{i:t_i>0} \beta_i - \sum_{i:t_i<0} \beta_i) + \lambda_2\|\beta\|_2^2. \tag{4.34}$$

  [18] established a linear rate of convergence for the case of CD applied to a class of strongly convex and continuously differentiable functions (which includes 4.34). Applying [18]'s result to (4.34) leads to (4.12).

□

Proof of Theorem 4.2

Proof. First, we will show that the algorithm terminates in a finite number of iterations. Suppose the algorithm does not terminate after T iterations. Then, we have a sequence $\{\beta^t\}_0^T$ of stationary solutions (since these are the outputs of Algorithm 4.1—see Theorem 4.1). Let $S_t = \text{Supp}(\beta^t)$. Each stationary solution β^t is a minimizer of the convex function $\beta_{S_t} \mapsto G(\beta_{S_t})$, and consequently $P(\beta^t) = \min\{P(\beta) \mid \beta \in \mathbb{R}^p, \text{Supp}(\beta) = S_t\}$. Moreover, the definition of Step 2 and the descent property of cyclic CD imply $P(\beta^T) < P(\beta^{T-1}) < \cdots < P(\beta^0)$. Therefore, the same support cannot appear more than once in the sequence $\{\beta^t\}_0^T$, and we conclude that the algorithm terminates in a finite number of iterations with a solution β^*. Finally, we note that β^* is the output of Algorithm 4.1 so it must satisfy the characterization given in Theorem 4.1. Moreover, the search in Step 2 must fail at β^* (otherwise, the algorithm does not terminate), and thus (4.15) holds. □

Proof of Lemma 4.1

Proof. First, we recall that $\delta_j = |\hat{L}\beta_j^* - \nabla_j g(\beta^*)|$ for any $j \in [p]$. Let $S = \text{Supp}(\beta^*)$ and fix some $j \in S$. By Theorem 4.1, we have $\beta_S^* \in \arg\min_{\beta_S} G(\beta_S)$, which is equivalent to $0 \in \partial G(\beta_S^*)$. The zero subgradient condition directly implies that $\nabla_j g(\beta_S^*) = -\lambda_1 \operatorname{sign}(\beta_j^*) - 2\lambda_2|\beta_j^*|$. Substituting this expression into δ_j, we get:

$$\begin{aligned}\delta_j &= |(\hat{L} + 2\lambda_2)\beta_j^* + \lambda_1 \operatorname{sign}(\beta_j^*)| \\ &= (\hat{L} + 2\lambda_2)|\beta_j^*| + \lambda_1 \\ &\geq \sqrt{2\lambda_0(\hat{L} + 2\lambda_2)} + \lambda_1, \end{aligned} \tag{4.35}$$

where (4.35) follows from inequality $|\beta_j^*| \geq \sqrt{\frac{2\lambda_0}{\hat{L}_j + 2\lambda_2}}$ (due to Theorem 4.1) and the fact that $\hat{L} \geq \hat{L}_j$. Now fix some $i \notin S$. Using Theorem 4.1 and $\hat{L} \geq \hat{L}_i$:

$$\delta_i = |\nabla_i g(\beta^*)| \leq \sqrt{2\lambda_0(\hat{L}_i + 2\lambda_2)} + \lambda_1 \leq \sqrt{2\lambda_0(\hat{L} + 2\lambda_2)} + \lambda_1. \tag{4.36}$$

Inequalities (4.35) and (4.36) imply that $\delta_j \geq \delta_i$ for any $j \in S$ and $i \notin S$. Since $\|\beta^*\|_0 = k$, we have $\delta_{(k)} \geq \delta_i$ for any $i \notin S$, which combined with the fact that $\beta_S^* \in \arg\min_{\beta_S} G(\beta_S)$ (from Theorem 4.1) implies that β^* satisfies the fixed point conditions for IHT stated in Theorem 4.3. □

Choice of J: Sorted Gradients

Let β_j^1 denote the solution obtained after the first application of the thresholding operator to minimize the function in (4.16). Note that we are interested in coordinates with nonzero β_j^1 (because in step 1 of Algorithm 4.3, we check whether β_j should be zero). Coordinates with nonzero β_j^1 must satisfy $|\nabla_j g(\beta^t - e_i\beta_i^t)| - \lambda_1 > 0$ (this follows from (4.10) with $\lambda_0 = 0$). Using (4.6), it can be readily seen that we have the following lower bound on the improvement in the objective if $|\nabla_j g(\beta^t - e_i\beta_i^t)| - \lambda_1 > 0$:

$$P(\beta^t - e_i\beta_i^t) - P(\beta^t - e_i\beta_i^t + e_j\beta_j^1) \geq \frac{1}{2(L_j + 2\lambda_2)}(|\nabla_j g(\beta^t - e_i\beta_i^t)| - \lambda_1)^2 - \lambda_0. \tag{4.37}$$

The choices of $j \in S^c$ with larger $|\nabla_j g(\beta^t - e_i \beta_i^t)|$ have a larger r.h.s. in inequality (4.37) and thus are expected to have lower objectives. Therefore, instead of searching across all values of $j \in S^c$ in Step 2 of Algorithm 4.2, we restrict $j \in J$.

Proof of Lemma 4.2

Proof. Problem (4.21) can be rewritten as

$$\begin{aligned} \min_{\beta, z_I} \Big[\min_{z_{\mathcal{I}^c}} \quad & G(\beta) + \lambda_0 \sum_{i=1}^{p} z_i \Big] \\ & |\beta_i| \leq \mathcal{M} z_i, \quad i \in [p] \\ & z_i \in [0,1], \quad i \notin \mathcal{I} \\ & z_i \in \{0,1\}, \quad i \in \mathcal{I}. \end{aligned}$$

Solving the inner minimization problem leads to $z_i = |\beta_i|/\mathcal{M}$ for every $i \in \mathcal{I}^c$. Plugging the latter solution into the objective function, we arrive to the result of the lemma. □

Proof of Theorem 4.4

A useful proposition

Proposition 4.1 (stated below) is an essential step in the proof of Theorem 4.4. This allows us to control the supremum of a random variable (of interest) over a bounded set of $2k$ sparse vectors.

Proposition 4.1. *Let* $\delta \in (0, 1/2)$, $\tau = 6L\sqrt{\frac{\lambda(k)}{n}\left(k \log\left(Rp/k\right) + \log\left(1/\delta\right)\right)}$ *and define*

$$\Delta(z) = \frac{1}{n} \sum_{i=1}^{n} f\left(\langle x_i, \beta^* + z \rangle; y_i\right) - \frac{1}{n} \sum_{i=1}^{n} f\left(\langle x_i, \beta^* \rangle; y_i\right), \quad \forall z. \tag{4.38}$$

If Assumptions 4.2, 4.4(k) and 4.5(δ) hold then:

$$\mathbb{P}\left(\sup_{\substack{z\in\mathbb{R}^p \\ \|z\|_0\leq 2k,\ \|z\|_2\leq 3R}} \left\{|\Delta(z)-\mathbb{E}\left(\Delta(z)\right)|-\tau\|z\|_2\vee\tau^2\right\}\geq 0\right)\leq\delta.$$

Proof. We divide the proof into 3 steps. First, we upper-bound the quantity $|\Delta(z)-\mathbb{E}\left(\Delta(z)\right)|$ for any $2k$ sparse vector with Hoeffding inequality. Second, we extend the result to the maximum over an ϵ-net. We finally control the maximum over the compact set and derive our proposition.

Step 1: We fix $z\in\mathbb{R}^p$ such that $\|z\|_0\leq 2k$ and introduce the random variables $Z_i,\forall i$ as follows

$$Z_i = f\left(\langle x_i,\beta^*+z\rangle;y_i\right)-f\left(\langle x_i,\beta^*\rangle;y_i\right).$$

Assumption 4.2 guarantees that $f(.;y)$ is Lipschitz with constant L, which leads to:

$$|Z_i|\leq L\left|\langle x_i,z\rangle\right|.$$

Note that $\Delta(z)=\frac{1}{n}\sum_{i=1}^n Z_i$. We introduce a small quantity $\eta>0$ later explicited in the proof. Using Hoeffding's inequality and Assumption 4.4(k) it holds: $\forall t>0$,

$$\begin{aligned}
\mathbb{P}\left(|\Delta(z)-\mathbb{E}\left(\Delta(z)\right)|\geq t\left(\|z\|_2\vee\eta\right)\middle|X\right) &\leq 2\exp\left(-\frac{2n^2t^2\left(\|z\|_2\vee\eta\right)^2}{\sum_{i=1}^n L^2\langle x_i,z\rangle^2}\right)\\
&= 2\exp\left(-\frac{2n^2t^2\left(\|z\|_2^2\vee\eta^2\right)}{L^2\|Xz\|_2^2}\right)\\
&\leq 2\exp\left(-\frac{2nt^2\left(\|z\|_2^2\vee\eta^2\right)}{L^2\lambda(k)\|z\|_2^2}\right)\\
&\leq 2\exp\left(-\frac{2nt^2}{L^2\lambda(k)}\right).
\end{aligned}\tag{4.39}$$

Note that in the above display, the r.h.s. bound does not depend upon X (the conditioning event in the l.h.s. of display (4.39)).

Step 2: We consider an ϵ-net argument to extend the result to any $2k$ sparse vector satisfying $\|z\|_2\leq 3R$. We recall that an ϵ-net of a set $\mathcal{I}$ is a subset $\mathcal{N}$ of $\mathcal{I}$ such that each

element of $\mathcal{I}$ is at a distance at most ϵ of $\mathcal{N}$.

Lemma 1.18 in [151] proves that for any value $\epsilon \in (0,1)$, the ball $\left\{z \in \mathbb{R}^d : \ \|z\|_2 \leq 3R\right\}$ has an ϵ-net of cardinality $|\mathcal{N}| \leq \left(\frac{6R+1}{\epsilon}\right)^d$. Consequently, we can fix an ϵ-net $\mathcal{N}_{k,R}$ of the set $\mathcal{I}_{k,R} = \{z \in \mathbb{R}^p : \ \|z\|_0 = 2k \ ; \ \|z\|_2 \leq 3R\}$ with cardinality $\binom{p}{2k}\left(\frac{6R+1}{\epsilon}\right)^{2k}$. This along with equation (4.39) leads to: $\forall t > 0$,

$$\begin{aligned} \mathbb{P}\left(\sup_{z \in \mathcal{N}_{k,R}} \left(|\Delta(z) - \mathbb{E}\left(\Delta(z)\right)| - t\left(\|z\|_2 \vee \eta\right) \right) \geq 0 \right) \\ \leq \binom{p}{2k}\left(\frac{6R+1}{\epsilon}\right)^{2k} 2\exp\left(-\frac{2nt^2}{L^2\lambda(k)}\right). \end{aligned} \tag{4.40}$$

Step 3: We finally extend the result to any vector in $\mathcal{I}_{k,R}$. This is done by expressing the supremum of the random variable $|\Delta(z) - \mathbb{E}\left(\Delta(z)\right)|$ over the entire set $\mathcal{I}_{k,R}$ with respect to its supremum over the ϵ-net $\mathcal{N}_{k,R}$.

For $z \in \mathcal{I}_{k,R}$, there exists $z_0 \in \mathcal{N}_{k,R}$ such that $\|z - z_0\|_2 \leq \epsilon$. With Assumption 4.2 and Cauchy-Schwartz inequality, we obtain:

$$|\Delta(z) - \Delta(z_0)| \leq \frac{1}{n}\sum_{i=1}^{n} L\left|\langle x_i, z - z_0\rangle\right| \leq \frac{1}{\sqrt{n}} L\|X(z - z_0)\|_2 \leq L\sqrt{\lambda(k)}\epsilon. \tag{4.41}$$

For a fixed value of t, we define the operator

$$f_t(z) = |\Delta(z) - \mathbb{E}\left(\Delta(z)\right)| - t\left(\|z\|_2 \vee \eta\right), \ \forall z.$$

Using the (reverse) triangle inequality, it holds that:

$$|\Delta(z) - \Delta(z_0)| \geq |\Delta(z) - \mathbb{E}\left(\Delta(z)\right)| - |\Delta(z_0) - \mathbb{E}\left(\Delta(z_0)\right)| - \left|\mathbb{E}\left(\Delta(z)\right) - \mathbb{E}\left(\Delta(z_0)\right)\right|. \tag{4.42}$$

In addition, note that:

$$\left(\|z - z_0\|_2 \vee \eta\right) \quad + \quad \left(\|z\|_2 \vee \eta\right) \quad \geq \quad \|z_0\|_2 \vee \eta. \tag{4.43}$$

Using (4.42) and (4.43) in $f_t(z)$ we have:

$$f_t(z_0) \geq f_t(z) - |\Delta(z) - \Delta(z_0)| - |\mathbb{E}(\Delta(z) - \Delta(z_0))| - t(\|z - z_0\|_2 \vee \eta). \tag{4.44}$$

Now using (4.41) and Jensen's inequality, we have:

$$\begin{aligned} |\Delta(z) - \Delta(z_0)| &\leq L\sqrt{\lambda(k)}\epsilon \\ |\mathbb{E}(\Delta(z) - \Delta(z_0))| &\leq L\sqrt{\lambda(k)}\epsilon. \end{aligned} \tag{4.45}$$

Applying (4.45) and $\|z - z_0\|_2 \leq \epsilon$ to the right hand side of (4.44), we have:

$$f_t(z_0) \geq f_t(z) - 2L\sqrt{\lambda(k)}\epsilon - t(\epsilon \vee \eta). \tag{4.46}$$

Suppose we choose

$$\begin{aligned} \eta &= 4L\sqrt{\frac{\lambda(k)}{n}k\log(p/k)} \\ \epsilon = \frac{\eta^2}{4L\sqrt{\lambda(k)}} \quad &= \sqrt{\lambda(k)}\frac{4Lk\log(p/k)}{n} \\ t &\geq \eta/4. \end{aligned}$$

This implies that:

$$\epsilon \leq \eta \qquad (\text{using } k/n\log(p/k) \leq 1 \text{ by Assumption 5}),$$

$$L\sqrt{\lambda(k)}\epsilon \leq t\eta \qquad (\text{using } t \geq \eta/4 \implies t\eta \geq \frac{\eta^2}{4} = L\sqrt{\lambda(k)}\epsilon).$$

Using the above, we obtain a lower bound to the r.h.s. of (4.46). This leads to the following chain of inequalities:

$$\begin{aligned} f_t(z) - 2L\sqrt{\lambda(k)}\epsilon - t(\epsilon \vee \eta) &\geq f_t(z) - 3t\eta \\ &= |\Delta(z) - \mathbb{E}(\Delta(z))| - t(\|z\|_2 \vee \eta) - 3t\eta \\ &\geq |\Delta(z) - \mathbb{E}(\Delta(z))| - 4t(\|z\|_2 \vee \eta) \\ &= f_{4t}(z). \end{aligned} \tag{4.47}$$

Consequently, Equations (4.46) and (4.47) lead to:

$$\forall t \geq \eta/4,\ \forall z \in \mathcal{I}_{k,R},\ \exists z_0 \in \mathcal{N}_{k,R} :\ f_{4t}(z) \leq f_t(z_0),$$

which can be equivalently written as:

$$\sup_{z \in \mathcal{I}_{k,R}} f_t(z) \leq \sup_{y \in \mathcal{N}_{k,R}} f_{t/4}(y),\ \forall t \geq \eta. \tag{4.48}$$

Lemma 2.7 in [151] gives the relation $\binom{p}{2k} \leq \left(\frac{pe}{2k}\right)^{2k}$. Using equation (4.48) along with the union-bound (4.40), it holds that: $\forall t \geq \eta$,

$$\begin{aligned}
\mathbb{P}\left(\sup_{z \in \mathcal{I}_{k,R}} f_t(z) \geq 0\right) &\leq \mathbb{P}\left(\sup_{z \in \mathcal{N}_{k,R}} f_{t/4}(z) \geq 0\right) \\
&\leq \binom{p}{2k}\left(\frac{6R+1}{\epsilon}\right)^{2k} 2\exp\left(-\frac{2nt^2}{16L^2\lambda(k)}\right) \\
&\leq \left(\frac{pe}{2k}\right)^{2k}\left(\frac{6R+1}{\epsilon}\right)^{2k} 2\exp\left(-\frac{2nt^2}{16L^2\lambda(k)}\right) \\
&\leq 2\left(\frac{pe}{2k}\ \frac{7R}{\epsilon}\right)^{2k} \exp\left(-\frac{2nt^2}{16L^2\lambda(k)}\right).
\end{aligned} \tag{4.49}$$

By Assumption 4.5 we have $7en \leq 3L\sqrt{\lambda(k)}p\log(p/k)$; and using the definition of ϵ (above), it follows that:

$$\frac{7e}{2\epsilon} \leq \frac{p}{k} \leq \frac{Rp}{k},$$

where in the last inequality we used the assumption $R \geq 1$. Using this in (4.49) we have:

$$\mathbb{P}\left(\sup_{z \in \mathcal{I}_{k,R}} f_t(z) \geq 0\right) \leq 2\left(\frac{Rp}{k}\right)^{4k} \exp\left(-\frac{2nt^2}{16L^2\lambda(k)}\right),\ \forall t \geq \eta. \tag{4.50}$$

We want the right-hand side of Equation (4.50) to be smaller than δ. To this end, we need to select $t^2 \geq \frac{16L^2\lambda(k)}{2n}\left[4k\log\left(\frac{Rp}{k}\right) + \log(2) + \log\left(\frac{1}{\delta}\right)\right]$ and $t \geq \eta$. A possible choice for t that we use is:

$$\tau = 6L\sqrt{\frac{\lambda(k)}{n}\left(k\log(Rp/k) + \log(1/\delta)\right)}.$$

We conclude that with probability at least $1-\delta$:

$$\sup_{z\in\mathcal{I}_{k,R}} f_\tau(z) \leq 0.$$

Note that

$$f_\tau(z) = |\Delta(z) - \mathbb{E}\left(\Delta(z)\right)| - \tau\left(\|z\|_2 \vee \eta\right) \geq |\Delta(z) - \mathbb{E}\left(\Delta(z)\right)| - \tau\|z\|_2 \vee \tau^2.$$

It then holds with probability at least $1-\delta$ that:

$$\sup_{\substack{z\in\mathbb{R}^p \\ \|z\|_0\leq 2k,\ \|z\|_2\leq 3R}} \left\{|\Delta(z) - \mathbb{E}\left(\Delta(z)\right)| - \tau\|z\|_2 \vee \tau^2\right\} \leq 0,$$

which concludes the proof. □

Proof of Theorem 4.4

Proof. The $2k$ sparse vector $h = \hat{\beta} - \beta^*$ satisfies: $\|h\|_2 \leq \|\hat{\beta}\|_2 + \|\beta^*\|_2 \leq 3R$. Thus applying Proposition 4.1 to $\Delta(h)$ (see the definition in Equation 4.38) it holds with probability at least $1-\delta$ that:

$$\begin{aligned}
\Delta(h) &\geq \mathbb{E}\left(\Delta(h)\right) - \tau\|h\|_2 \vee \tau^2 \\
&= \frac{1}{n}\mathbb{E}\left(\sum_{i=1}^n f\left(\langle x_i, \beta^* + h\rangle; y_i\right) - \sum_{i=1}^n f\left(\langle x_i, \beta^*\rangle; y_i\right)\right) - \tau\|h\|_2 \vee \tau^2 \\
&= \mathbb{E}\left[f\left(\langle x_i, \beta^* + h\rangle; y_i\right) - f\left(\langle x_i, \beta^*\rangle; y_i\right)\right] - \tau\|h\|_2 \vee \tau^2 \\
&= \mathcal{L}(\beta^* + h) - \mathcal{L}(\beta^*) - \tau\|h\|_2 \vee \tau^2.
\end{aligned} \tag{4.51}$$

To control the difference $\mathcal{L}(\beta^* + h) - \mathcal{L}(\beta^*)$ we consider the following two cases.

Case 1 ($\|h\|_2 \leq r(k)$): If $\|h\|_2 \leq r(k)$ then by definition of $r(k)$ in (4.26) it holds that

$$\mathcal{L}(\beta^* + h) - \mathcal{L}(\beta^*) \geq \frac{1}{4}\kappa(k)\|h\|_2^2. \tag{4.52}$$

Case 2: We consider the case when $\|h\|_2 > r(k)$. Since $\beta \mapsto \mathcal{L}(\beta)$ is convex, the one-dimensional function $t \to \mathcal{L}(\beta^* + tz)$ is convex and it holds:

$$\begin{aligned}
\mathcal{L}(\beta^* + h) - \mathcal{L}(\beta^*) &\geq \frac{\|h\|_2}{r(k)} \left\{ \mathcal{L}\left(\beta^* + \frac{r(k)}{\|h\|_2} h\right) - \mathcal{L}(\beta^*) \right\} \\
&\geq \frac{\|h\|_2}{r(k)} \inf_{\substack{z:\ \|z\|_0 \leq 2k \\ \|z\|_2 = r(k)}} \{\mathcal{L}(\beta^* + z) - \mathcal{L}(\beta^*)\} \\
&\geq \frac{\|h\|_2}{r(k)} \frac{1}{4}\kappa(k) r(k)^2 \\
&= \frac{1}{4}\kappa(k) r(k) \|h\|_2.
\end{aligned} \tag{4.53}$$

In (4.53), the first inequality follows by observing that the one-dimensional function $x \mapsto (g(a+x) - g(a))/x$ defined on $x > 0$ is increasing for any one-dimensional convex function $x \mapsto g(x)$. The last inequality in display (4.53) follows from the definition of $r(k)$ as in (4.26).

Combining Equations (4.51), (4.52) and (4.53), we obtain the following restricted strong convexity property (which holds with probability at least $1 - \delta$):

$$\Delta(h) \geq \frac{1}{4}\kappa(k) \left\{ \|h\|_2^2 \wedge r(k)\|h\|_2 \right\} - \tau \|h\|_2 \vee \tau^2.$$

□

Proof of Theorem 4.5

Proof. Using the observation that $\hat{\beta}$ is a minimizer of Problem (4.24); and the representation $\hat{\beta} = \beta^* + h$, it holds that:

$$\frac{1}{n} \sum_{i=1}^{n} f\left(\langle x_i, \beta^* + h \rangle; y_i\right) \leq \frac{1}{n} \sum_{i=1}^{n} f\left(\langle x_i, \beta^* \rangle; y_i\right).$$

This relation is equivalent to saying that $\Delta(h) \leq 0$. Consequently, by combining this relation with (4.27) (see Theorem 4.4), it holds with probability at least $1 - \delta$:

$$\frac{1}{4}\kappa(k) \left\{ \|h\|_2^2 \wedge r(k)\|h\|_2 \right\} \leq \tau \|h\|_2 \vee \tau^2. \tag{4.54}$$

We now consider two cases.

Case 1: Let $\|h\|_2 \leq \tau$. Then we have that

$$\|h\|_2 \leq \tau = 6L\sqrt{\frac{\lambda(k)}{n}\left(k\log\left(Rp/k\right) + \log\left(1/\delta\right)\right)}.$$

Case 2: We consider the case where $\|h\|_2 > \tau$. With probability $1-\delta$ it holds (from Inequality 4.54) that:

$$\frac{1}{4}\kappa(k)\left\{\|h\|_2^2 \wedge r(k)\|h\|_2\right\} \leq \tau\|h\|_2,$$

which can be simplified to:

$$\|h\|_2 \wedge r(k) \leq \frac{4\tau}{\kappa(k)} = \frac{24L}{\kappa(k)}\sqrt{\frac{\lambda(k)}{n}\left(k\log\left(Rp/k\right) + \log\left(1/\delta\right)\right)}. \tag{4.55}$$

Note that by Assumption $5(\delta)$, the r.h.s. of the above is smaller than $r(k)$. The l.h.s. of (4.55) is the minimum of two terms $\|h\|_2$ and $r(k)$; and since this is lower than $r(k)$, we conclude that

$$\|h\|_2 \leq \frac{24L}{\kappa(k)}\sqrt{\frac{\lambda(k)}{n}\left(k\log\left(Rp/k\right) + \log\left(1/\delta\right)\right)}.$$

Combining the results from Cases 1 and 2, we conclude that with probability at least $1-\delta$, the following holds:

$$\|h\|_2 \leq CL\max\left\{1, \frac{1}{\kappa(k)}\right\}\sqrt{\frac{\lambda(k)}{n}\left(k\log\left(Rp/k\right) + \log\left(1/\delta\right)\right)},$$

where C is an universal constant. This concludes the proof of the theorem.

□

Proof of Corollary 4.1:

Proof. Let us define the random variable:

$$W = \frac{1}{L^2 \widetilde{\kappa}^2 \lambda(k)} \|\hat{\beta} - \beta^*\|_2^2.$$

Since the difference $\hat{\beta} - \beta^*$ is bounded, then W is upper-bounded by a constant. In addition, because Assumption 4.5 is satisfied for $\delta > 0$ small enough, Theorem 4.5 leads to the existence of a constant C such that

$$\mathbb{P}\left(W \leq \frac{C}{n}\left(k \log(Rp/k) + \log(1/\delta)\right)\right) \geq 1 - \delta, \ \forall \delta \in (0,1).$$

This relation can be equivalently expressed as

$$\mathbb{P}\left(W/C \geq \frac{k \log(p/k)}{n} + t\right) \leq e^{-nt}, \ \forall t \geq 0.$$

Let us define $H = (k/n)\log(p/k)$. By integration it holds:

$$\begin{aligned} \mathbb{E}(W) &= \int_0^{+\infty} C\mathbb{P}\left(|W|/C \geq t\right) dt \\ &\leq \int_0^{+\infty} C\mathbb{P}\left(|W|/C \geq t + H\right) dt + CH \\ &\leq \int_0^{+\infty} Ce^{-nt} dt + CH = \frac{C}{n} + CH \leq 2CH \end{aligned} \tag{4.56}$$

We finally conclude:

$$\mathbb{E}\|\hat{\beta} - \beta^*\|_2^2 \lesssim L^2 \lambda(k) \widetilde{\kappa}^2 \frac{k \log(p/k)}{n}.$$

□

4.B Appendix: Additional Experiments

Here we present additional experimental results complementing the real data set experiments presented in Section 4.5.4. We compare the performance of our proposed algorithm with ℓ_1-regularization. In particular, Figure 4-5 considers the ℓ_0-ℓ_2 penalty with CD (Algorithm 4.1); Figure 4-6 considers the ℓ_0-ℓ_1 penalty with local search (Algorithm 4.2 with $m = 1$); and Figure 4-7 considers the ℓ_0-ℓ_1 penalty with CD (Algorithm 4.1).

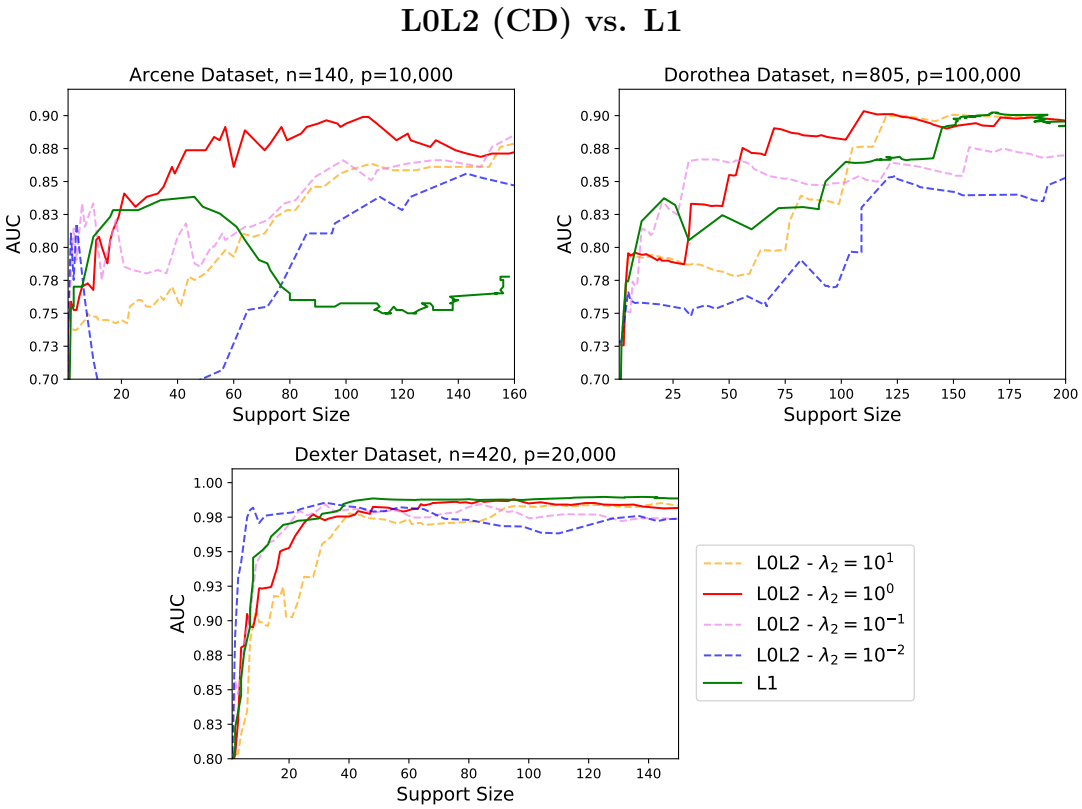


Figure 4-5: Plots of AUC versus corresponding support sizes for the Arcene, Dorothea, and Dexter data sets. The green curves correspond to logistic regression with ℓ_1 regularization. The other curves correspond to logistic regression with ℓ_0-ℓ_2 regularization using Algorithm 4.1 for different values of λ_2 (see legend).

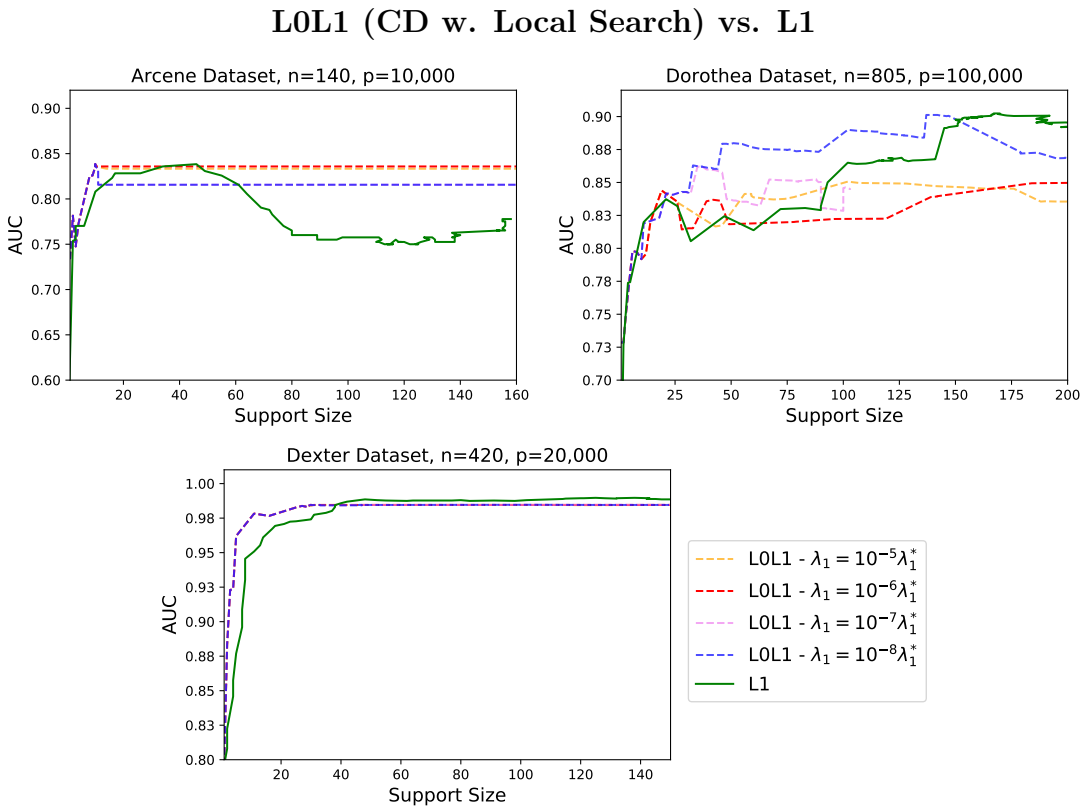


Figure 4-6: Plots of the AUC versus the support size for the Arcene, Dorothea, and Dexter data sets. The green curves correspond to logistic regression with ℓ_1 regularization. The other curves correspond to logistic regression with ℓ_0-ℓ_1 regularization using Algorithm 4.2 (with $m = 1$) for different values of λ_2 (see legend).

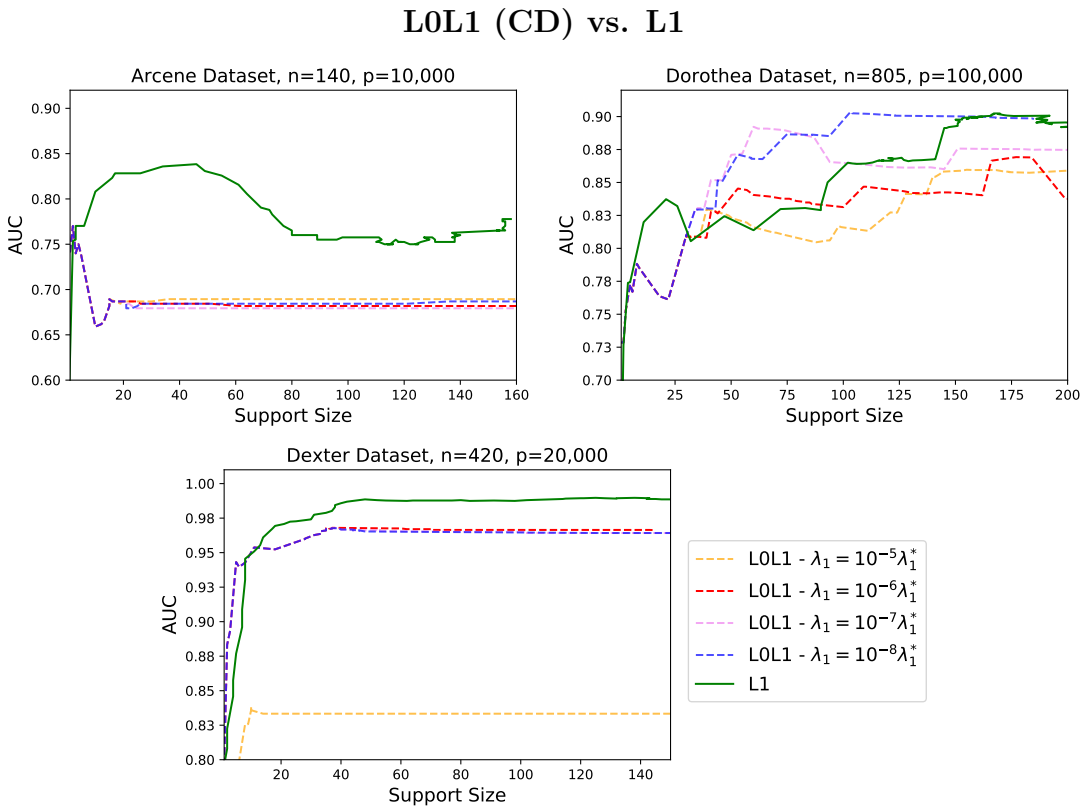


Figure 4-7: Plots of the AUC versus the support size for the Arcene, Dorothea, and Dexter data sets. The green curves correspond to logistic regression with ℓ_1 regularization. The other curves correspond to logistic regression with ℓ_0-ℓ_1 regularization using Algorithm 4.1 for different values of λ_2 (see legend).

Chapter 5

Grouped Variable Selection with Discrete Optimization: Computational and Statistical Perspectives

This chapter is based on [89]. It is a joint work with Rahul Mazumder and Peter Radchenko.

5.1 Introduction

Sparsity plays a ubiquitous role in modern statistical regression, especially when the number of predictors is large relative to the number of observations. In this chapter, we focus on the case where predictors have a natural group structure. Typical examples where such a structure appears are models with multilevel categorical predictors and models that represent nonlinear effects of continuous variables using basis functions [44, 182, 85]. Grouping may also arise from scientifically meaningful prior knowledge about the collection of the predictor variables. More specifically, we consider the usual linear regression framework with response $\mathbf{y}_{n\times 1}$ and model matrix $\mathbf{X}_{n\times p} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$. We suppose that the p predictors are divided into q pre-specified, non-overlapping groups. For a given $\boldsymbol{\beta} \in \mathbb{R}^p$ and each $g \in \{1, ..., q\}$, we

denote by $\boldsymbol{\beta}_g$ the sub-vector of $\boldsymbol{\beta}$ whose coefficients correspond to the predictors in group g. Following the traditional approach in high-dimensional regression, we assume that few of the regression coefficients are nonzero, i.e., the model is sparse. This leads to a natural generalization of the classical best subset selection problem in linear regression [122, 24] to the group setting:

$$\min_{\boldsymbol{\beta}} \quad \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_0 \sum_{g=1}^{q} \mathbf{1}(\boldsymbol{\beta}_g \neq \mathbf{0}), \tag{5.1}$$

where $\mathbf{1}(\cdot)$ is the indicator function, and λ_0 is a non-negative regularization parameter that controls the number of nonzero groups selected. We will refer to Problem (5.1) as the Group ℓ_0 problem.

Problem (5.1) is NP-Hard [128] and poses computational challenges. A rich body of prior work explores sparsity-inducing methods to obtain approximate solutions to (5.1). Popular methods include: convex optimization based procedures, such as Group Lasso [182], which is a generalization of the Lasso approach [162] to the grouped setting, and local solutions to nonconvex optimization problems arising from group-nonconvex regularizers, such as SCAD, MCP and others [185, 96]. Despite the appeal of these approaches, the statistical and computational aspects of *optimal* solutions to (5.1) remain to be understood at a deeper level. To this end, we aim to advance the computational frontiers of Problem (5.1) using novel tools from discrete optimization. Our proposed combinatorial optimization-based algorithms are scalable. In particular, they can deliver optimal solutions to (5.1) for instances that are much larger than state-of-the-art approaches. We also develop a better understanding of the statistical properties of Problem (5.1) both theoretically and empirically.

Computation. We propose new algorithms based on combinatorial optimization for solving Problem (5.1) and its variants. First we present approximate algorithms: they deliver high-quality solutions using a combination of cyclic coordinate descent and local combinatorial optimization [86]. These algorithms have runtimes comparable to popular approaches for grouped variable selection (for example, Group Lasso or MCP), but deliver solutions with considerably improved statistical performance (for example, in terms of prediction and variable selection), as we demonstrate in our experiments. Our approximate algorithms de-

liver good-quality feasible solutions to (5.1) but are unable to certify (global) optimality of solutions via matching lower bounds on the optimal objective value of (5.1). Certifying optimality is not only important from a methodological perspective but can also be beneficial in practice for mission-critical applications. For example, having certifiably optimal solutions can engender trust and provide transparency in consequential applications such as healthcare. Thus, we propose a new tailored branch-and-bound based optimization framework for solving (5.1) to certifiable optimality.

In our exact (global optimization) framework, we formulate the Group ℓ_0 problem as a Mixed Integer Program (MIP). However, in a departure from earlier work [24, 28], we propose a custom branch-and-bound (BnB) algorithm to solve the MIP. Indeed, MIP-based techniques have gained considerable traction recently to solve to (near) optimality the best subset selection problem, where all groups are of size one [24, 28, 118, 115, 86, 180, 90]. All these works, with the exception of [90], leverage capabilities of powerful commercial MIP solvers such as Gurobi and CPLEX. These solvers have gained wide adoption in the past two decades due to major advances in algorithms and software development [32, 101]. However, these general-purpose solvers may take several hours to certify optimality on small instances (for example, with $p = 1000$). In contrast, our custom BnB algorithm exploits problem-specific structure to scale to much larger instances. For example, it can solve to optimality instances with $p = 5 \times 10^6$ – this is 1000 times larger than what can be handled using Gurobi's MIP-solver. Our BnB algorithm generalizes to the grouped setting the approach of [90] developed for the best subset selection problem.

Statistical properties. Statistical properties of Group Lasso have been extensively studied, and it has been shown, both empirically and theoretically, that it performs well in sparse high-dimensional settings [52, 9, 127, 98, 176, 111, 134], under certain assumptions on the data. However, Group Lasso also has its shortcomings, similar to those of Lasso in high dimensional linear regression [24, 44, 86]. More specifically, depending on the penalty weight, the resulting model may either be very dense or, alternatively, comes with overly shrunk nonzero coefficients. This problem is aggravated when the groups are correlated with each other, as Group Lasso tends to bring in all of the correlated groups in lieu of searching for a

more parsimonious model. For further discussions of these issues in the special case of Lasso see, for example, [186, 117, 44, 24], and the references therein. In this chapter, we demonstrate, both empirically and theoretically, that the Group ℓ_0 methodology has advantages over its Group Lasso counterpart in a variety of regimes. In particular, as a consequence of directly controlling the sparsity level in the optimization problem, our framework leads to substantially sparser models under similar data fidelity. Moreover, in many scenarios where the predictors are highly correlated, our approach performs better in terms of both estimation and prediction.

Additive models with ℓ_0-sparsity. In addition to linear models, we also study an important example of regression with group structure that arises in high-dimensional sparse additive modeling [85, 82]. Here, we estimate a nonparametric multivariate regression function in q covariates, $(x_1, \ldots, x_q)$, which we model as a sparse additive sum of the form $\sum_{j \in S} f_j(x_j)$, where $S \subset \{1, \ldots, q\}$. In this setting, each group generally corresponds to the basis representation of a given additive component, one for each of the q predictors. Because the groups are allowed to be large, additional regularization needs to be imposed, typically in the form of a roughness type penalty on the regression functions. A number of successful Group Lasso-based approaches have been proposed and analyzed in this setting – see, for example, [120, 149, 97, 102, 145, 183] and the references therein. To our knowledge, this is the first work to explore statistical and computational aspects of Group ℓ_0-based formulations in the context of sparse additive modeling. We show theoretically and empirically that Group ℓ_0 based methods enjoy certain statistical advantages when compared to the Group Lasso-based counterparts.

Contributions. The focus of this chapter is on Problem (5.1) and the sparse additive modeling problem (which can be formulated as a variant of Problem (5.1), as we discuss in Section 5.2). Our main contributions for these two problems can be summarized as follows:

- We develop fast approximate algorithms, based on first-order and local combinatorial optimization. We establish convergence guarantees for these algorithms and provide

useful characterizations of the corresponding local minima. Our experiments indicate that these algorithms can have an edge in terms of statistical performance over popular alternatives for grouped variable selection.

- We present mixed integer second order cone program (MISOCP) formulations for the Group ℓ_0-based estimators; and design a novel specialized, nonlinear branch-and-bound (BnB) framework for solving the MISOCP to global optimality. Our custom BnB solver can handle instances with 5×10^6 variables – more than a 1000 times larger than what can be handled by state-of-the-art commercial MISOCP solvers.

- We establish non-asymptotic prediction and estimation error bounds for our proposed estimators, for both the high-dimensional linear regression and sparse additive modeling problems. We show that under the assumption of sparsity, these error bounds compare favorably with the ones for Group Lasso.

- We demonstrate empirically that our approach appears to outperform the state of the art (for example, Group Lasso and available algorithms for nonconvex penalized estimators) in a variety of high-dimensional regimes and under different statistical metrics (for example, prediction, estimation, and variable selection).

Organization. In Section 5.2, we present formulations for the Group ℓ_0 and sparse additive modeling problems. Section 5.3 presents approximate algorithms based on first-order and local combinatorial optimization algorithms. Then, in Section 5.4, we present our exact MIP algorithm. Statistical properties of our approach are investigated in Section 5.5. Section 5.6 presents computational experiments. Technical proofs and additional computational details are provided in the appendix.

Notation. For any non-negative integer k, we denote the set $\{1, ..., k\}$ by $[k]$. The complement of a set A is denoted by A^c. We denote the index sets corresponding to the q groups of predictors by $\mathcal{G}_g$, for $g \in [q]$ so that $\cup_{g=1}^{q} \mathcal{G}_g = [p]$ and $\mathcal{G}_g \cap \mathcal{G}_\ell = \emptyset$ for all $g \neq \ell$. For a vector $\boldsymbol{\theta}$, we use the notation $\text{Supp}(\boldsymbol{\theta})$ to denote the group support, i.e., $\text{Supp}(\boldsymbol{\theta}) = \{g \mid \boldsymbol{\theta}_g \neq 0, g \in [q]\}$. We also define a measure of ℓ_0-group sparsity (i.e., number

of nonzero groups): $G(\boldsymbol{\theta}) := \sum_{g=1}^{q} \mathbf{1}(\boldsymbol{\theta}_g \neq \mathbf{0})$. We denote the gradient of a scalar-valued function, say $J(\boldsymbol{\theta})$, by $\nabla J(\boldsymbol{\theta})$. Moreover, we use the notation $\nabla_{\boldsymbol{\theta}_g} J(\boldsymbol{\theta})$ to refer to the sub-vector of $\nabla J(\boldsymbol{\theta})$ corresponding to the variables in $\boldsymbol{\theta}_g$. Vectors and matrices are denoted in boldface.

5.2 Optimization Problems Considered

In this section, we present optimization formulations for the Group ℓ_0 approach (and its variants), as well as the ℓ_0-sparse additive function estimation approach.

5.2.1 Group ℓ_0 with Ridge Regularization

The algorithms discussed in this chapter apply to the Group ℓ_0 estimator (5.1) with an optional ridge regularization term:

$$\min_{\boldsymbol{\beta}} \quad \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_0 \sum_{g=1}^{q} \mathbf{1}(\boldsymbol{\beta}_g \neq \mathbf{0}) + \lambda_2 \|\boldsymbol{\beta}\|_2^2, \tag{5.2}$$

where $\lambda_0 > 0$ controls the number of selected groups, and $\lambda_2 \geq 0$ controls the strength of the ridge regularization. Our proposed algorithms apply to *both* settings: $\lambda_2 = 0$ and $\lambda_2 > 0$ in Problem (5.2). The choice of the ridge term in (5.2) is motivated by earlier work in the context of best-subset selection [115, 86], which suggest that when the signal-to-noise ratio (SNR) is low, additional ridge regularization can improve the prediction performance of best-subset selection (both theoretically and empirically). Additionally, as discussed in Section 5.4.2, the choice $\lambda_2 > 0$, allows for deriving stronger MIP formulations by appealing to perspective formulations [67, 74].

5.2.2 Nonparametric Additive Models with ℓ_0-sparsity

In the multivariate setting, estimating the conditional mean function $\mathbb{E}(y|\mathbf{x}) = f(x_1, \ldots, x_q)$ becomes notoriously difficult, due to curse of dimensionality. To overcome this problem, additive approximation schemes [82] are commonly used as an effective methodology: $f(\mathbf{x}) =$

$\sum_{j=1}^{q} f_j(x_j)$. A popular approach [see, for example, 171] is to choose f_j from some smooth functional class $\mathcal{C}_j$, such as the class of twice continuously differentiable functions. Given the observations $(y_i, \mathbf{x}_i)$, $i \in [n]$, the additive model $f(\mathbf{x})$ can be estimated by solving the following optimization problem:

$$\min_{f} \quad \sum_{i=1}^{n} (y_i - \sum_{j=1}^{q} f_j(x_{ij}))^2 + \lambda \sum_{j=1}^{q} \text{Pen}(f_j), \tag{5.3}$$

where $\text{Pen}(f_j)$ is a roughness penalty that controls the amount of smoothness in function f_j.

A key ingredient in the additive function fitting framework is the estimation of a univariate smooth regression function based on observations $(y_i, u_i), i \in [n]$. Suppose, for simplicity, that the u_is are distinct and $u_i \in [0, 1]$ for all i. For illustration, let us take $\text{Pen}(g) = \int_0^1 (g''(u))^2 du$. Then, the solution to the corresponding (infinite dimensional) univariate problem is of the form: $g(u) = \alpha_0 + \alpha_1 u + \sum_{j=1}^{n} \gamma_j N_j(u)$, where $N_j(u)$ are some cubic spline basis functions, such as truncated power series functions, natural cubic splines or the B-spline basis functions, with knots chosen at the distinct data points $u_i, i \in [n]$. Note that $\int_0^1 (g''(u))^2 du = \boldsymbol{\gamma}' \boldsymbol{\Omega} \boldsymbol{\gamma}$, where $\boldsymbol{\Omega}$ is an $n \times n$ positive definite matrix with the elements $\omega_{ij} = \int_0^1 N_i''(u) N_j''(u) du$. If we refer to the corresponding functional class as $\mathcal{C}$, define the elements of $\mathbf{g}$ as $g_i := g(u_i)$, for $i = 1, \ldots, n$, and let $\|\mathbf{g}\|_{\mathcal{C}}^2 := \boldsymbol{\gamma}' \boldsymbol{\Omega} \boldsymbol{\gamma}$, then the univariate optimization problem is equivalent to

$$(\hat{g}_1, \ldots, \hat{g}_m) = \hat{\mathbf{g}} \in \arg\min \ \|\mathbf{y} - \mathbf{g}\|_2^2 + \lambda \|\mathbf{g}\|_{\mathcal{C}}^2. \tag{5.4}$$

Problem (5.4) is a generalized least squares problem in $(\alpha_0, \alpha_1, \boldsymbol{\gamma})$. A direct extension to the additive model setting is given by the following formulation:

$$\min \quad \|\mathbf{y} - \sum_{j=1}^{q} \mathbf{f}_j\|_2^2 + \lambda \sum_{j=1}^{q} \|\mathbf{f}_j\|_{\mathcal{C}_j}^2, \tag{5.5}$$

where $f_j \in \mathcal{C}_j$ and $\mathbf{f}_j = (f_j(x_{ij}), \ldots, f_j(x_{nj}))$.

We wish to impose sparsity on the additive components f_j, $j \in [q]$, which naturally leads to the following optimization problem:

$$\min \quad \|\mathbf{y} - \sum_{j=1}^{q} \mathbf{f}_j\|_2^2 + \lambda_0 \sum_{j=1}^{q} \mathbf{1}(\mathbf{f}_j \neq 0) + \lambda \sum_{j=1}^{q} \|\mathbf{f}_j\|_{\mathcal{C}_j}^2. \tag{5.6}$$

We note that the choice $\text{Pen}(f_j) = \sqrt{\int (f_j''(u))^2 du}$ leads to the optimization problem

$$\min \quad \|\mathbf{y} - \sum_{j=1}^{q} \mathbf{f}_j\|_2^2 + \lambda_0 \sum_{j=1}^{q} \mathbf{1}(\mathbf{f}_j \neq 0) + \lambda \sum_{j=1}^{q} \|\mathbf{f}_j\|_{\mathcal{C}_j}. \tag{5.7}$$

Problems (5.6) and (5.7) are close cousins and result in similar estimators. The terms $\sum_j \|\mathbf{f}_j\|_{\mathcal{C}_j}$ and $\sum_j \|\mathbf{f}_j\|_{\mathcal{C}_j}^2$ encourage smoothness in each of the additive components, while the sum of indicators directly controls the number of included predictors. In Section 5.5, we establish theoretical error bounds for the estimator that corresponds to Problem (5.7).

Connections with Group Lasso-type penalization schemes. For Grouped Lasso-type penalization schemes, the choice of the penalty becomes rather subtle. Problem (5.3) with $\text{Pen}(f_j) = \|\mathbf{f}_j\|_{\mathcal{C}_j}^2$ does *not* induce sparsity in $\|\mathbf{f}_j\|_{\mathcal{C}_j}$'s for finite λ. Alternatively, the choice $\text{Pen}(f_j) = \|\mathbf{f}_j\|_{\mathcal{C}_j}$ does result in several components $\|\mathbf{f}_j\|_{\mathcal{C}_j}$ being set to zero when λ is large. Note, however, that $\|\mathbf{f}_j\|_{\mathcal{C}_j} = 0$ does not imply $f_j = 0$. This is because $\|\mathbf{f}_j\|_{\mathcal{C}_j}$ is a seminorm that is not affected by the linear components of f_j. To set $f_j = 0$ one needs to include the linear components into the penalty. To overcome these limitations, alternatives have been proposed – here we mention some penalization schemes that are used to encourage selection and smoothness. One possible choice [120] is $\text{Pen}(f_j) = \sqrt{\|\mathbf{f}_j\|_2^2 + \lambda' \|\mathbf{f}_j\|_{\mathcal{C}_j}^2}$, where $\|\mathbf{f}_j\|_2$ denotes the usual ℓ_2 norm of the vector $\mathbf{f}_j$. The corresponding penalization term is $\lambda \sum_j \text{Pen}(f_j)$, and, hence, the parameters λ and λ' jointly control smoothness and sparsity. The sum of $\|\mathbf{f}_j\|_2^2$ and $\lambda' \|\mathbf{f}_j\|_{\mathcal{C}_j}^2$ leads to double penalization, thereby potentially resulting in unwanted shrinkage that may interfere with variable selection. Similar issues arise with the choices $\text{Pen}(f_j) = \|\mathbf{f}_j\|_2 + \lambda' \|\mathbf{f}_j\|_{\mathcal{C}_j}$, considered in [44], and $\text{Pen}(f_j) = \sqrt{\|\mathbf{f}_j\|_2^2 + \lambda' \|\mathbf{f}_j\|_{\mathcal{C}_j}^2} + \tilde{\lambda} \|\mathbf{f}_j\|_{\mathcal{C}_j}^2$, which appears in [120].

Thus, the choice of $\text{Pen}(f_j)$ plays an important role in obtaining sparsity for Lasso-type regularization methods. In contrast, the levels of smoothness and sparsity are controlled separately in the ℓ_0-formulations: Problems (5.6) and (5.7). Group Lasso-type penalization schemes may be interpreted as convex relaxations of the ℓ_0-penalty appearing in Problem (5.7), as discussed in Appendix 5.A.

Other choices of smooth function classes. We note that the above framework, where each additive component is taken to be a cubic spline, can be generalized to more flexible smooth nonparametric models, depending upon the choice of $\text{Pen}(\cdot)$ and the functional classes $\mathcal{C}_j$s. For example, one may consider the class of functions that are τ times continuously differentiable, together with the choice $\text{Pen}(f_j) = \int f_j^{(\tau)}(u)du$, where $f^{(\tau)}$ denotes the τth derivative of f_j – solutions to these problems are given by splines of order τ [171].

Another popular paradigm pursued in several works [102, 109, 148] is the Reproducing Kernel Hilbert Space (RKHS) framework, wherein every $\mathcal{C}_j$ is taken to be a Hilbert space encouraging some form of smoothness on f_j. Here, $\text{Pen}(f_j) = \|\mathbf{f}_j\|_{K_j}$ is an appropriate Hilbert space norm.

5.2.3 General Problem Formulation

Our focus in this chapter is on Problem (5.2) and the sparse additive modeling problems defined in (5.6) and (5.7). These three problems can all be formulated as follows:

$$\min_{\boldsymbol{\beta}} \quad \boldsymbol{\beta}'\mathbf{P}\boldsymbol{\beta} + \langle \mathbf{a}, \boldsymbol{\beta} \rangle + \lambda_0 G(\boldsymbol{\beta}) + \lambda_1 \sum_{g=1}^{q} \|\mathbf{P}_g \boldsymbol{\beta}_g\|_2, \tag{5.8}$$

for suitable choices of $\boldsymbol{a}$, $\mathbf{P} \succeq \mathbf{0}$, $\mathbf{P}_g \succ \mathbf{0}$, $g \in [q]$, where we recall that $G(\boldsymbol{\beta}) := \sum_{g=1}^{q} \mathbf{1}(\boldsymbol{\beta}_g \neq \mathbf{0})$. The term $\sum_{g=1}^{q} \|\mathbf{P}_g \boldsymbol{\beta}_g\|_2$ is only used for the sparse additive modeling problem in (5.7). Problems (5.1) and (5.6) can be obtained by setting $\lambda_1 = 0$ and choosing $\mathbf{P}$ and $\boldsymbol{a}$ appropriately.

To simplify the presentation, we apply a change of variable in Problem (5.8): $\boldsymbol{\theta}_g = \mathbf{P}_g^{\frac{1}{2}} \boldsymbol{\beta}_g$

for $g \in [q]$. This leads to the following equivalent problem:

$$\min_{\boldsymbol{\theta}} \quad h(\boldsymbol{\theta}) := \underbrace{\boldsymbol{\theta}'\mathbf{W}\boldsymbol{\theta} + \langle \mathbf{b}, \boldsymbol{\theta} \rangle}_{:=\ell(\boldsymbol{\theta})} \; + \; \underbrace{\lambda_0 G(\boldsymbol{\theta}) + \lambda_1 \sum_{g=1}^{q} \|\boldsymbol{\theta}_g\|_2}_{:=\Omega(\boldsymbol{\theta})}, \tag{5.9}$$

for appropriately defined[1] $\mathbf{W}$ and $\mathbf{b}$. Our algorithmic development will focus on (5.9).

Overview of our algorithms: Problem (5.9) is nonconvex due to the discontinuity in $G(\boldsymbol{\theta})$. In Section 5.3, we design fast algorithms that can obtain high-quality approximate solutions for this problem. In Section 5.4, we develop an exact algorithmic framework, based on a custom MIP solver, which obtains certifiably optimal solutions to (5.9). Our algorithm constructs: (i) a sequence of feasible solutions, whose objective values are valid upper bounds, and (ii) a sequence of lower bounds (a.k.a. dual bounds). As our BnB algorithm progresses, these upper and lower bounds converge towards the optimal objective of Problem (5.9). The solver terminates and certifies optimality when the upper and lower bounds match[2]. Our experiments indicate that high-quality initial solutions, as available from the algorithms presented in Section 5.3, can significantly speed up convergence and reduce memory requirements in our BnB algorithm.

5.3 Approximate Algorithms

In this section, we develop fast approximate algorithms to obtain high quality local minimizers for Problem (5.9). While these algorithms do not deliver certificates of optimality (via dual bounds), they attain nearly-optimal (and at times optimal) solutions to many statistically challenging instances, in running times comparable to group Lasso-based algorithms.

A main workhorse of our approximate algorithms is a nonstandard application of cyclic block coordinate descent (BCD) to the discontinuous objective function (5.9). We draw inspiration

[1] Let $\mathbf{D}_1 = \text{diag}(\mathbf{P}_1^{\frac{1}{2}}, \ldots, \mathbf{P}_q^{\frac{1}{2}})$ be a block diagonal matrix. Then $\mathbf{W} = \mathbf{D}_1^{-1}\mathbf{P}\mathbf{D}_1^{-1}$, and $\mathbf{b} = \mathbf{D}_1^{-1}\mathbf{a}$.

[2] In practice, MIP solvers terminate when the difference between the upper and lower bounds are below a small, user-defined threshold.

from the appealing scalability properties of coordinate descent in sparse learning problems [see, for example, 69, 16, 86]. Our second algorithm is based on local combinatorial search and is used to improve the quality of solutions obtained by BCD. We establish convergence guarantees for these two algorithms.

Our algorithms arise from studying necessary optimality conditions for Problem (5.9). To this end, we show that the quality of solutions obtained by BCD are of higher quality than local solutions corresponding to the popular proximal gradient descent (PGD) [139] algorithm[3]. The local minimizers corresponding to local combinatorial search form a smaller subset of those available from BCD. In this section, we establish the following hierarchy among the classes of local minima:

$$\text{Global Minima} \subseteq \text{Local Search Minima} \subseteq \text{BCD Minima} \subseteq \text{PGD Minima}. \tag{5.10}$$

Above, PGD minima correspond to the fixed points of the PGD algorithm; they include all the fixed points of our proposed BCD algorithm. As we move from right to left in the above hierarchy, the classes become smaller, i.e., they impose stricter necessary optimality conditions. At the top of the hierarchy we have the global minimizers of the problem, which can be obtained using our exact MIP-based framework (we discuss this in Section 5.4). Our approximate algorithms are inspired by recent work [86] on the sparse regression problem, but the approach presented here has notable differences. In particular, the coordinate descent algorithm in [86] performs exact minimization per coordinate, which can be computationally expensive when extended to the group setting. Thus, our proposed BCD algorithm performs inexact minimization per group. In addition, the presence of ℓ_2 norms in our objective function makes the analysis for the rate of convergence for our algorithm different.

[3]Though PGD is popularly used in the context of convex optimization problems, it also leads to useful algorithms for nonconvex sparse learning problems. In particular, PGD for our problem can be viewed as a generalization of the iterative hard thresholding (IHT) algorithm [35] to the group setting.

5.3.1 Block Coordinate Descent

We present a cyclic BCD algorithm to obtain good feasible solutions to Problem (5.9) and establish convergence guarantees. We first introduce a useful upper bound for $\ell(\boldsymbol{\theta})$. For every $g \in [q]$, we define $S_g = \{(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \mid \boldsymbol{\theta}_i = \tilde{\boldsymbol{\theta}}_i,\ \forall i \in [q] \text{ s.t. } i \neq g\}$. By the Block Descent Lemma [22], the following upper bound holds for every $g \in [q]$:

$$\ell(\boldsymbol{\theta}) \leq \ell(\tilde{\boldsymbol{\theta}}) + \langle \nabla_{\boldsymbol{\theta}_g} \ell(\tilde{\boldsymbol{\theta}}), \boldsymbol{\theta}_g - \tilde{\boldsymbol{\theta}}_g \rangle + \frac{L_g}{2} \|\boldsymbol{\theta}_g - \tilde{\boldsymbol{\theta}}_g\|_2^2, \qquad \forall (\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \in S_g, \tag{5.11}$$

where L_g is the "group-wise" Lipschitz constant of $\nabla \ell(\boldsymbol{\theta})$, i.e., L_g is a constant which satisfies: $\|\nabla_{\boldsymbol{\theta}_g} \ell(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}_g} \ell(\tilde{\boldsymbol{\theta}})\|_2 \leq L_g \|\boldsymbol{\theta}_g - \tilde{\boldsymbol{\theta}}_g\|_2$, for all $(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \in S_g$. Since $\ell(\boldsymbol{\theta})$ is a quadratic function, $L_g = 2\sigma_{\max}(\mathbf{W}_g)$, where $\mathbf{W}_g$ is the submatrix of $\mathbf{W}$ with columns and rows restricted to group g, and $\sigma_{\max}(\cdot)$ denotes the largest eigenvalue.

Cyclic BCD sequentially minimizes the objective of (5.9) with respect to one group of variables while the other groups are held fixed. Let $\boldsymbol{\theta}^l$ be the iterate obtained by the algorithm after the l-th iteration. Then, in iteration $l+1$, the variables in a group g (say), are updated while the other groups are held fixed. Specifically, we have $(\boldsymbol{\theta}^l, \boldsymbol{\theta}^{l+1}) \in S_g$. Using (5.11) with $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^l$, $\hat{L}_g > L_g$ and adding $\Omega(\boldsymbol{\theta})$ to both sides we get:

$$h(\boldsymbol{\theta}) \leq \tilde{g}(\boldsymbol{\theta}; \boldsymbol{\theta}^l) := \ell(\boldsymbol{\theta}^l) + \langle \nabla_{\boldsymbol{\theta}_g} \ell(\boldsymbol{\theta}^l), \boldsymbol{\theta}_g - \boldsymbol{\theta}_g^l \rangle + \frac{\hat{L}_g}{2} \|\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^l\|_2^2 + \Omega(\boldsymbol{\theta}). \tag{5.12}$$

Note that the left hand side of (5.12) is the objective function of Problem (5.9). We obtain $\boldsymbol{\theta}_g^{l+1}$ by minimizing the upper bound on our objective, $\tilde{g}(\boldsymbol{\theta}; \boldsymbol{\theta}^l)$, with respect to $\boldsymbol{\theta}_g$:

$$\boldsymbol{\theta}_g^{l+1} \in \arg\min_{\boldsymbol{\theta}_g} \tilde{g}(\boldsymbol{\theta}; \boldsymbol{\theta}^l) = \arg\min_{\boldsymbol{\theta}_g} \frac{\hat{L}_g}{2} \left\| \boldsymbol{\theta}_g - \left(\boldsymbol{\theta}_g^l - \frac{1}{\hat{L}_g} \nabla_{\boldsymbol{\theta}_g} \ell(\boldsymbol{\theta}^l) \right) \right\|_2^2 + \Omega(\boldsymbol{\theta}_g). \tag{5.13}$$

Although nonconvex, the minimization problem in (5.13) admits a closed-form solution,

which can be obtained via the operator $H : \mathbb{R}^u \to \mathbb{R}^u$ defined as follows:

$$H(\mathbf{z}; \boldsymbol{\lambda}; \hat{L}_g) = \begin{cases} \frac{\mathbf{z}}{\|\mathbf{z}\|_2}\left[\|\mathbf{z}\|_2 - \frac{\lambda_1}{\hat{L}_g}\right] & \text{if } \|\mathbf{z}\|_2 > \sqrt{\frac{2\lambda_0}{\hat{L}_g}} + \frac{\lambda_1}{\hat{L}_g} \\ 0 & \text{otherwise} \end{cases} \tag{5.14}$$

where $\boldsymbol{\lambda} = (\lambda_0, \lambda_1)$. It can be readily seen that an optimal solution of (5.13) is given by $H(\mathbf{z}; \boldsymbol{\lambda}; \hat{L}_g)$, where $\mathbf{z} = \boldsymbol{\theta}_g^l - \frac{1}{\hat{L}_g}\nabla_{\boldsymbol{\theta}_g}\ell(\boldsymbol{\theta}^l)$. Below we summarize our proposed cyclic BCD algorithm.

Algorithm 5.1: Cyclic Block Coordinate Descent (BCD)

- **Input:** Initialization $\boldsymbol{\theta}^0$ and $\hat{L}_g$ for every $g \in [q]$.
- **Repeat** Steps 1, 2 for $l = 0, 1, 2, \ldots$ until convergence:
  1. $g \leftarrow 1 + (l \mod q)$ and $\boldsymbol{\theta}_j^{l+1} \leftarrow \boldsymbol{\theta}_j^l$ for all $j \neq g$
  2. $\boldsymbol{\theta}_g^{l+1} \leftarrow H(\mathbf{z}; \boldsymbol{\lambda}; \hat{L}_g)$, where $\mathbf{z} = \boldsymbol{\theta}_g^l - (1/\hat{L}_g)\nabla_{\boldsymbol{\theta}_g}\ell(\boldsymbol{\theta}^l)$.

Convergence Analysis. To establish convergence of the sequence $\boldsymbol{\theta}^l$ in Algorithm 5.1, we make use of the following assumption.

Assumption 1. *At least one of the following conditions holds:*

(a) Strong Convexity: $\mathbf{W} \succ 0$.

(b) Restricted Strong Convexity: Let $\hat{\boldsymbol{\theta}}$ *be a (Group Lasso) solution defined as* $\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) + \lambda_1 \sum_{g=1}^q \|\boldsymbol{\theta}_g\|_2$. *Let* $k = \max_{\boldsymbol{\theta}}\{\|\boldsymbol{\theta}\|_0 \mid G(\boldsymbol{\theta}) \leq G(\hat{\boldsymbol{\theta}})\}$. *Every collection of* k *columns in* $\mathbf{W}$ *are linearly independent, and the initial solution* $\boldsymbol{\theta}^0$ *(in Algorithm 5.1) satisfies* $h(\boldsymbol{\theta}^0) \leq h(\hat{\boldsymbol{\theta}})$.

Assumption 1(a) holds if a ridge regularization term is used, i.e., it holds for Problem (5.2) with $\lambda_2 > 0$. Assumption 1(b) is less restrictive because we can have $\mathbf{W} \succeq \mathbf{0}$. Suppose that for some non-negative integer u, every set of u columns in $\mathbf{W}$ are linearly independent. Then, in the Group Lasso problem (defined in Assumption 1(b)), λ_1 can be chosen sufficiently large so that some Group Lasso solution $\hat{\boldsymbol{\theta}}$ satisfies $k \leq u$. If $\hat{\boldsymbol{\theta}}$ is used to initialize Algorithm 5.1,

then Assumption 1(b) is satisfied.

The following theorem establishes a linear convergence guarantee for the sequence generated by Algorithm 5.1.

Theorem 5.1. *Let $\{\boldsymbol{\theta}^l\}$ be the sequence generated by Algorithm 5.1 and suppose that Assumption 1 holds. Then,*

1. *The group support stabilizes after a finite number of iterations, i.e., there exists an integer K and a support $S \subseteq [q]$ such that $Supp(\boldsymbol{\theta}^l) = S$ for all $l \geq K$.*

2. *The sequence $\{\boldsymbol{\theta}^l\}$ converges to a solution $\boldsymbol{\theta}^*$, with $Supp(\boldsymbol{\theta}^*) = S$, satisfying:*

$$\boldsymbol{\theta}^*_S \in \arg\min_{\boldsymbol{\theta}_S} \quad \ell(\boldsymbol{\theta}_S) + \lambda_1 \sum_{g \in S} \|\boldsymbol{\theta}_g\|_2 \tag{5.15}$$

$$\|\boldsymbol{\theta}^*_g\|_2 \geq \sqrt{\frac{2\lambda_0}{\hat{L}_g}}, \quad \forall g \in S \tag{5.16}$$

$$\|\nabla_{\boldsymbol{\theta}_g} \ell(\boldsymbol{\theta}^*)\|_2 \leq \sqrt{2\lambda_0 \hat{L}_g} + \lambda_1, \quad \forall g \in S^c. \tag{5.17}$$

3. *The function $\boldsymbol{\theta}_S \mapsto \ell(\boldsymbol{\theta}_S)$ is strongly convex with a strong convexity parameter $\sigma_S > 0$. Let L_S be the Lipschitz constant of $\nabla_{\boldsymbol{\theta}_S} \ell(\boldsymbol{\theta}_S)$. Define $\hat{L}_{max} = \max_{g \in S} \hat{L}_g + 2\lambda_1$ and $\hat{L}_{min} = \min_{g \in S} \hat{L}_g + 2\lambda_1$. Then, for $l \geq K$, the following holds:*

$$h(\boldsymbol{\theta}^{(l+1)q}) - h(\boldsymbol{\theta}^*) \leq \left(1 - \frac{\sigma_S}{\eta}\right)\Big(h(\boldsymbol{\theta}^{lq}) - h(\boldsymbol{\theta}^*)\Big), \tag{5.18}$$

where $\eta = 2\hat{L}_{max}(1 + |S|(L_S + 2\lambda_1|S|)^2 \hat{L}_{min}^{-2})$.

The proof of Theorem 5.1 is in the appendix. We present here a high-level sketch of the proof. We establish part 1 by proving a sufficient decrease condition. For part 2, we show that the objective function restricted to the group support S is strongly convex, and thus convergence follows from standard results on cyclic BCD, e.g., [22]. To establish the linear rate of convergence in part 3 of the theorem, we extend the result of [18] who show that cyclic BCD can achieve a linear rate of convergence on smooth and strongly convex functions: note

that our objective function after support stabilization is *not* smooth due to the presence of the term $\sum_{g \in S} \|\boldsymbol{\theta}_g\|_2$.

Optimality conditions of BCD and PGD. The conditions in Theorem 5.1 (part 2) characterize a fixed point of Algorithm 5.1. These are necessary optimality conditions for Problem (5.9) since any global minimizer must be a fixed point for Algorithm 5.1. In what follows, we will show that the necessary optimality conditions imposed by PGD (which is a generalization of [35] to the group setting) are generally less restrictive compared to those imposed by Algorithm 5.1. Note that PGD is an iterative algorithm whose updates for Problem (5.9) are given by:

$$\boldsymbol{\theta}^{l+1} \in \arg\min_{\boldsymbol{\theta}} \left\{ \frac{1}{2\tau} \|\boldsymbol{\theta} - (\boldsymbol{\theta}^l - \tau \nabla \ell(\boldsymbol{\theta}^l))\|_2^2 + \Omega(\boldsymbol{\theta}) \right\}, \tag{5.19}$$

where $\tau > 0$ is a step size. Let L be the Lipschitz constant of $\nabla \ell(\boldsymbol{\theta})$. For a constant step size, the update in (5.19) converges if $\tau = 1/\hat{L}$ where $\hat{L}$ is a constant chosen such that $\hat{L} > L$ [see, for example, 86, 112]. For the choice $\tau = 1/\hat{L}$, it can be readily checked that any fixed point of PGD satisfies the three optimality conditions in Theorem 5.1 (part 2), but with $\hat{L}_g$ replaced by $\hat{L}$. The group-wise Lipschitz constant L_g satisfies $L_g \leq L$ (for any g). In many high-dimensional problems, we can have $L_g \ll L$ [see 16, 86]. Thus, Algorithm 5.1 generally imposes more restrictive necessary optimality conditions compared to PGD, which can lead to higher quality local minima in practice. This establishes a part of the hierarchy in (5.10).

5.3.2 Local Combinatorial Search

In this section, we introduce a local combinatorial search algorithm to improve the quality of solutions obtained by cyclic BCD (Algorithm 5.1). The algorithm performs the following two steps in the t-th iteration:

1. **Block Coordinate Descent:** We run Algorithm 5.1 initialized at the current solution $\boldsymbol{\theta}^{t-1}$ to obtain a solution $\boldsymbol{\theta}^t$. We denote the indices of the nonzero groups in $\boldsymbol{\theta}^t$ by $\text{Supp}(\boldsymbol{\theta}^t) = S$.

2. **Group Combinatorial Search:** We attempt to improve the solution $\boldsymbol{\theta}^t$ by *swapping* groups of variables from inside and outside the support S. In particular, we search for two subsets $S_1 \subseteq S$ and $S_2 \subseteq S^c$ such that removing S_1 from the support, adding S_2 to the support, and then optimizing over the groups in S_2, improves the current objective. To ensure that the local search problem is computationally feasible, we restrict our search to subsets satisfying $|S_1| \leq m$ and $|S_2| \leq m$, where m is a pre-specified integer that takes relatively small values (for example, in the range 1 to 10).

We present a formal description of the optimization problem in step 2 (above). We denote the standard basis of $\mathbb{R}^p$ by $\{\mathbf{e}_1, \ldots, \mathbf{e}_p\}$. Given a set $J \subseteq [q]$, we define the $p \times p$ matrix $\mathbf{U}^J$ as follows: the i-th column of $\mathbf{U}^J$ is $\mathbf{e}_i$ if $i \in \cup_{g \in J} \mathcal{G}_g$ and $\mathbf{0}$ otherwise. In other words, for any $\boldsymbol{\theta} \in \mathbb{R}^p$, we have $(\mathbf{U}^J \boldsymbol{\theta})_i = \theta_i$ if $i \in \cup_{g \in J} \mathcal{G}_g$ and 0 otherwise. The optimization problem in Step 2 is given by:

$$\min_{S_1, S_2, \boldsymbol{\theta}} \; h(\boldsymbol{\theta}^t - \mathbf{U}^{S_1} \boldsymbol{\theta}^t + \mathbf{U}^{S_2} \boldsymbol{\theta}) \qquad \text{s.t.} \qquad S_1 \subseteq S, S_2 \subseteq S^c, |S_1| \leq m, |S_2| \leq m, \tag{5.20}$$

where we recall that $S = \text{Supp}(\boldsymbol{\theta}^t)$. If there is a feasible solution $\hat{\boldsymbol{\theta}}$ to (5.20) satisfying $h(\hat{\boldsymbol{\theta}}) < h(\boldsymbol{\theta}^t)$, then we move to the improved solution $\hat{\boldsymbol{\theta}}$; otherwise, we terminate the algorithm. We summarize the algorithm below:

Algorithm 5.2: Local Combinatorial Search

- **Input:** Initial solution $\boldsymbol{\theta}^0$ and swap subset size m.
- **Repeat** Steps 1–3 for $t = 1, 2, \ldots$ until convergence:
  1. Run Algorithm 5.1 initialized from $\boldsymbol{\theta}^{t-1}$ to obtain a solution $\boldsymbol{\theta}^t$.
  2. Search for a feasible solution $\hat{\boldsymbol{\theta}}$ to (5.20) satisfying $h(\hat{\boldsymbol{\theta}}) < h(\boldsymbol{\theta}^t)$.
  3. If step 2 succeeds, $\boldsymbol{\theta}^t \leftarrow \hat{\boldsymbol{\theta}}$. Otherwise, terminate.

Theorem 5.2 establishes that Algorithm 5.2 converges in a finite number of iterations and characterizes the corresponding solution.

Theorem 5.2. *Let $\{\boldsymbol{\theta}^t\}$ be the sequence of iterates generated by Algorithm 5.2 and suppose Assumption 1 holds. Then, $\boldsymbol{\theta}^t$ converges in a finite number of iterations to a solution that*

we denote by $\boldsymbol{\theta}^\dagger$. *Let* $S = Supp(\boldsymbol{\theta}^\dagger)$. *Then,* $\boldsymbol{\theta}^\dagger$ *satisfies the necessary optimality conditions in part 2 of Theorem 5.1. In addition,* $\boldsymbol{\theta}^\dagger$ *satisfies:*

$$h(\boldsymbol{\theta}^\dagger) \leq \min_{S_1, S_2, \boldsymbol{\theta}} \; h(\boldsymbol{\theta}^\dagger - \mathbf{U}^{S_1}\boldsymbol{\theta}^\dagger + \mathbf{U}^{S_2}\boldsymbol{\theta}) \quad s.t. \quad S_1 \subseteq S, S_2 \subseteq S^c, |S_1| \leq m, |S_2| \leq m. \tag{5.21}$$

Theorem 5.2 shows that the solutions obtained by Algorithm 5.2 impose more restrictive necessary optimality conditions (in particular, condition (5.21)) compared to Algorithm 5.1, which justifies part of the hierarchy in (5.10). This is expected, as every iteration of Algorithm 5.2 improves over a solution obtained by Algorithm 5.1. The quality of solutions returned by Algorithm 5.2 depends on the swap subset size m. For a sufficiently large choice of m, the algorithm will return a global minimizer. Intuitively, the computational cost of the local search in step 2 of Algorithm 5.2 increases with m. In our experiments, we observe that small choices such as $m = 1$ can lead to significant improvements in solution quality compared to algorithms that do not incorporate combinatorial optimization. These improvements are most pronounced in settings where $n \ll p$ or the predictors across groups are highly correlated. In Section 5.4.1, we present a MIP formulation for the local search problem in Algorithm 5.2 for $m > 1$. For the special case of $m = 1$, we use our own custom implementation that is more efficient than using a MIP-based approach.

5.3.3 Algorithms for the Cardinality Constrained Formulation

Algorithms 5.1 and 5.2 provide solutions for the (penalized) formulation in (5.9). While this leads to a family of high-quality estimators across a range of model sizes, it does not allow for explicit control over the number of nonzero groups $G(\boldsymbol{\theta})$. To this end, we consider the cardinality constrained variant of problem (5.9):

$$\min_{\boldsymbol{\theta}} \quad E(\boldsymbol{\theta}) := \ell(\boldsymbol{\theta}) + \lambda_1 \sum_{g \in [q]} \|\boldsymbol{\theta}_g\|_2 \quad \text{s.t. } G(\boldsymbol{\theta}) \leq k. \tag{5.22}$$

In order to obtain a solution to (5.22) with a desired support size, we propose the following procedure. First, we run Algorithm 5.2 (say) over a grid of λ_0-values to obtain a sequence

of solutions. Then, if a desired support size, say k, is missing, we obtain it by applying proximal gradient descent (PGD) to Problem (5.22):

$$\boldsymbol{\theta}^{l+1} \in \underset{\boldsymbol{\theta}:\ G(\boldsymbol{\theta})\leq k}{\arg\min} \left\{ \frac{1}{2\tau}\|\boldsymbol{\theta} - (\boldsymbol{\theta}^l - \tau\nabla\ell(\boldsymbol{\theta}^l))\|_2^2 + \lambda_1 \sum_{g\in[q]} \|\boldsymbol{\theta}_g\|_2 \right\}, \tag{5.23}$$

where $\tau > 0$ is a step size and the initial solution $\boldsymbol{\theta}^0$ can be obtained from Algorithm 5.2 (for example, we take a solution with group support size closest to k).

The next proposition establishes the convergence of update (5.23) and describes its fixed points.

Proposition 5.1. *Let $\{\boldsymbol{\theta}^l\}$ be the sequence of iterates generated the PGD updates* (5.23)*. Let L be the Lipschitz constant of $\nabla\ell(\boldsymbol{\theta})$ and a scalar $\hat{L}$ such that $\hat{L} > L$. Then, $\{\boldsymbol{\theta}^l\}$ converges for a step size $\tau = 1/\hat{L}$. Moreover, a solution $\boldsymbol{\theta}^*$ with group support S is a fixed point of* (5.23) *iff $G(\boldsymbol{\theta}^*) \leq k$, and*

$$\boldsymbol{\theta}^*_S \in \underset{\boldsymbol{\theta}_S}{\arg\min}\ E(\boldsymbol{\theta}_S) \quad \text{and} \quad \|\nabla_{\boldsymbol{\theta}_g}\ell(\boldsymbol{\theta}^*)\|_2 \leq \gamma_{(k)} \quad \text{for } g \in S^c,$$

where $\gamma_g = \|\hat{L}\boldsymbol{\theta}^*_g - \nabla_{\boldsymbol{\theta}_g}\ell(\boldsymbol{\theta}^*)\|_2$, and $\gamma_{(k)}$ denotes the kth largest value in the sequence $\{\gamma_g\}_{g=1}^q$.

We omit the proof of Proposition 5.1 as it can be established by a simple extension to the standard results on the convergence of IHT [for example, those in 35, 16].

5.4 Mixed Integer Programming

In this section, we propose MIP formulations and algorithms to solve (5.9) and the combinatorial search problem in Algorithm 5.2. Section 5.4.1 introduces MIP formulations, and Section 5.4.2 presents a new BnB algorithm for solving the corresponding problems to optimality.

5.4.1 MIP Formulations

Formulations for Problem (5.9)

Below we present two MIP-formulations for (5.9).

Big-M Formulation: We first present a Big-M based MIP formulation for Problem (5.9):

$$\min_{\boldsymbol{\theta},\mathbf{z}} \quad \ell(\boldsymbol{\theta}) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\boldsymbol{\theta}_g\|_2 \tag{5.24a}$$

$$\text{s.t.} \quad \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}} z_g, \quad g \in [q] \tag{5.24b}$$

$$z_g \in \{0,1\}, \quad g \in [q] \tag{5.24c}$$

where, the optimization variables are $\boldsymbol{\theta}$ (continuous) and $\mathbf{z}$ (binary). Above, $\mathcal{M}_{\text{U}}$ is an a-priori specified constant (leading to the name "Big-M") such that some optimal solution, say $\boldsymbol{\theta}^*$, to (5.9) satisfies $\max_{g\in[q]} \|\boldsymbol{\theta}^*_g\|_2 \leq \mathcal{M}_{\text{U}}$. In (5.24), the binary variable z_g controls whether all the regression coefficients in group g are zero or not: $z_g = 0$ implies that $\boldsymbol{\theta}_g = \mathbf{0}$, and $z_g = 1$ implies that $\|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}$. Such Big-M formulations are commonly used in mixed integer programming to model relations between discrete and continuous variables, and have been recently used in ℓ_0-regularized regression [24, 180] (for example). Various techniques have been proposed to estimate the constant $\mathcal{M}_{\text{U}}$ in practice; see [24] for a discussion on estimating the Big-M in the context of linear regression. The constraints in (5.24b) are second order cones [40]. Moreover, the objective function in (5.24) can be written as a linear function, with additional second order cone constraints to express the quadratic function $\ell(\boldsymbol{\theta})$ and the terms $\|\boldsymbol{\theta}_g\|_2,\ g \in [q]$. Thus, Problem (5.24) can be reformulated as a Mixed Integer Second Order Cone Program (MISOCP), which can be modeled and solved (for small/moderate problem instances) with commercial MIP solvers such as Gurobi, CPLEX, and MOSEK. We present an efficient, standalone BnB algorithm for (5.24) in Section 5.4.2.

Perspective Reformulation: Recall that Problem (5.9) contains a ridge term in its objective. The ridge term can be used to derive stronger MIP formulations for (5.9) based on

the perspective formulation [67, 74]. As we discuss below, the perspective-based formulation differs from the Big-M formulation (5.24)—when $\lambda_2 > 0$, it usually leads to tighter convex relaxations and consequently, reduced MIP runtimes. First, we rewrite (5.9) as

$$\min_{\boldsymbol{\theta},\mathbf{z}} \quad \tilde{\ell}(\boldsymbol{\theta}) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\boldsymbol{\theta}_g\|_2 + \lambda_2 \sum_{g=1}^{q} \|\boldsymbol{\theta}_g\|_2^2 \qquad \text{s.t.} \quad (5.24\text{b}), (5.24\text{c}) \tag{5.25}$$

where $\ell(\boldsymbol{\theta}) = \tilde{\ell}(\boldsymbol{\theta}) + \lambda_2 \|\boldsymbol{\theta}\|_2^2$. Using the perspective reformulation [67, 74, 61] for the ridge term $\sum_{g\in[q]} \|\boldsymbol{\theta}_g\|_2^2$ in the objective, we can reformulate (5.25) as

$$\min_{\boldsymbol{\theta},\mathbf{z},\boldsymbol{s}} \quad \tilde{\ell}(\boldsymbol{\theta}) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\boldsymbol{\theta}_g\|_2 + \lambda_2 \sum_{g=1}^{q} s_g, \tag{5.26a}$$

$$\text{s.t.} \quad \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}} z_g, \ g \in [q] \tag{5.26b}$$

$$\|\boldsymbol{\theta}_g\|_2^2 \leq s_g z_g, \quad g \in [q] \tag{5.26c}$$

$$z_g \in \{0,1\}, s_g \geq 0, \quad g \in [q]. \tag{5.26d}$$

Compared to (5.25), formulation (5.26) uses additional auxiliary variables $s_g \in \mathbb{R}_{\geq 0}, \ g \in [q]$ and rotated second order cone constraints: $\|\boldsymbol{\theta}_g\|_2^2 \leq s_g z_g$ for $g \in [q]$. Each s_g takes the place of the term $\|\boldsymbol{\theta}_g\|_2^2$ in the objective function in (5.24). Specifically, any optimal solution $(\boldsymbol{\theta}^*, \mathbf{z}^*, \boldsymbol{s}^*)$ to (5.26) must satisfy $s_g^* = \|\boldsymbol{\theta}_g^*\|_2^2$.

Although the MIP formulations (5.26) and (5.25) are equivalent, their continuous relaxations are generally different. The following proposition states that the relaxation of (5.26) is generally tighter (i.e., has a higher objective) than the relaxation of (5.25).

Proposition 5.2. *Let v_1 and v_2 be the objective values of* (5.25) *and* (5.26) *upon relaxing the binary variable z_g to $[0,1]$ for all $g \in [q]$. Let $(\boldsymbol{\theta}^*, \mathbf{z}^*, \mathbf{s}^*)$ be an optimal solution to the relaxation corresponding to v_2. Then, the following holds:*

$$v_2 - v_1 \geq \lambda_2 \sum_{g\in[q]|z_g^*>0} \|\boldsymbol{\theta}_g^*\|_2^2 \Big((z_g^*)^{-1} - 1 \Big).$$

Proposition 5.2 implies that using formulation (5.26) (over formulation (5.24)) can lead to tighter lower bounds for the root node relaxation; and hence tighter dual bounds for the node relaxations in the BnB tree. This can result in improved runtimes in the overall BnB solver (as we demonstrate in our experiments). Thus, in our algorithmic framework in Section 5.4.2, we focus on formulation (5.26). To be clear, our BnB procedure applies even without the presence of a ridge term (i.e., $\lambda_2 = 0$). Specifically, if $\lambda_2 = 0$ in (5.26), the conic constraints (5.26c) can be removed and formulation (5.26) reduces to the Big-M formulation in (5.24).

MIP Formulation for Local Combinatorial Search

We present a MIP formulation for the local search problem[4] that arises in Algorithm 5.2. Problem (5.20) can be formulated using the following Big-M based MIP:

$$\begin{aligned}
\min_{\mathbf{u},\mathbf{z},\boldsymbol{\theta}} \quad & \ell(\mathbf{u}) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\mathbf{u}_g\|_2 \\
\text{s.t.} \quad & \mathbf{u} = \boldsymbol{\theta}^t - \sum_{g \in S} \mathbf{U}^g \boldsymbol{\theta}^t (1 - z_g) + \sum_{g \in S^c} \mathbf{U}^g \boldsymbol{\theta} && \text{(5.27a)} \\
& \|\mathbf{u}_g\|_2 \leq \mathcal{M}_{\text{U}} z_g, \ g \in S^c && \text{(5.27b)} \\
& \sum_{g \in S} z_g \geq |S| - m, \ \sum_{g \in S^c} z_g \leq m && \text{(5.27c)} \\
& z_g \in \{0, 1\}, \quad g \in [q]. && \text{(5.27d)}
\end{aligned}$$

In the formulation above, we assume that $\mathcal{M}_{\text{U}}$ is chosen sufficiently large so that some optimal solution to (5.20), say $\boldsymbol{\theta}^*$, satisfies $\|\boldsymbol{\theta}^*_g\|_2 \leq \mathcal{M}_{\text{U}}, \ g \in S^c$. As we discuss below, the objective in (5.27) represents $h(\mathbf{u})$ with $\mathbf{u} = \boldsymbol{\theta}^t - \mathbf{U}^{S_1}\boldsymbol{\theta}^t + \mathbf{U}^{S_2}\boldsymbol{\theta}$, where $h(\mathbf{u})$, S_1 and S_2 are as defined in (5.20). Note that the variable $\mathbf{u}$ is an auxiliary variable introduced to simplify the presentation. The binary variables $z_g, g \in [q]$ are used to select the subsets $S_1 \subseteq S$ and $S_2 \subseteq S^c$. In particular, for $g \in S$, $z_g = 0$ iff $g \in S_1$, and this is encoded by constraint (5.27a). On the other hand, for $g \in S^c$, $z_g = 1$ iff $g \in S_2$, and this is encoded by constraints (5.27a)

[4]We recommend the use of the MIP formulations when $m \geq 2$. When $m = 1$ a solution to the local search procedure can be computed efficiently from first principles.

and (5.27b). Therefore, $\sum_{g=1}^{q} z_g$ is equal to $G(\mathbf{u})$. The constraints (5.27c) enforce $|S_1| \leq m$ and $|S_2| \leq m$.

The local search MIP-formulation (5.27) has a *smaller* search space compared to the full problem (5.24). This is due to the additional constraints appearing in (5.27c). Furthermore, Problem (5.27) effectively uses $|S^c|$-many 'free' continuous group-variables—this is in contrast to $|S| + |S^c|$ continuous group-variables appearing in the full problem. Thus, for small values of m, Problem (5.27) can be typically solved faster than the MIP formulation of (5.8). While (5.27) is based on a Big-M formulation, in the presence of an additional ridge regularizer, one can also derive a perspective reformulation using ideas similar to (5.26).

5.4.2 A Custom Nonlinear Branch-and-Bound Algorithm

High-performance commercial MIP solvers, such as Gurobi and CPLEX, often deliver state-of-the-art performance for a variety of MIP problems. These solvers are based on a BnB framework, which can solve MIP problems to global optimality, typically without having to explicitly enumerate all (exponentially many) solutions in the search space. These solvers are general-purpose and do not take into account the specific structure of the problems we consider here. Therefore, their performance can suffer: we have empirically observed that they may require several hours to solve (to certifiable optimality) instances of (5.26) with $p \sim 10^3$, and larger problems can take much longer.

To address this lack of scalability in general-purpose MIP solvers, we propose a specialized, nonlinear BnB framework for solving (5.26) to certifiable optimality. Our framework takes into account problem structure to achieve scalability. As we demonstrate in the experiments section, our BnB can solve instances with $p \sim 5 \times 10^6$ to certifiable optimality in minutes to hours, whereas Gurobi takes prohibitively long (at least a day) for $p \sim 10^3$. An important feature of our proposal is an open-source, standalone implementation of the BnB solver, which does not rely on sophisticated and proprietary BnB-capabilities of commercial MIP solvers (e.g., Gurobi). We first give a high-level overview of our novel nonlinear BnB framework and then dive into specific technical details.

Overview of Nonlinear BnB: Nonlinear BnB is a general framework for solving mixed integer nonlinear programs [20]. This framework constructs a search tree to partition the set of feasible solutions of the given MIP (Problem (5.26) in our case). Instead of explicitly enumerating all the (exponentially many) feasible solutions, BnB uses intelligent enumeration and methods to prune parts of the tree by using lower bounds (dual bounds) on the optimal objective value. In what follows, we briefly describe how the tree is constructed and pruned. Starting at the root node, the algorithm solves a nonlinear convex relaxation of Problem (5.26), where all binary variables are relaxed to $[0, 1]$ – this is usually referred to as the *root relaxation*. Then, the algorithm chooses a *branching variable*, say z_g, and creates two child nodes (optimization subproblems): one with $z_g = 0$ and another with $z_g = 1$, where all other binary variables are relaxed to $[0, 1]$. The algorithm then proceeds recursively: for every unvisited node, it solves the corresponding optimization problem and checks if there is any fractional (i.e., non-binary) variable z_g. If there is any fractional z_g, the branching process must continue — to this end, the algorithm branches on one fractional z_g, generating two new child nodes. Thus, every node in the search tree corresponds to an optimization subproblem and every edge represents a branching decision.

While growing the search tree, BnB maintains an upper bound on the objective function (which can be obtained from any feasible solution to the problem). If the optimization subproblem at the current node leads to an objective value that exceeds the upper bound, then the node is pruned (i.e., no children are generated for this node), because none of its descendants can have a better objective value than the upper bound. Another case where BnB can safely prune a node is when the corresponding subproblem leads to an integral solution, i.e., a binary $\mathbf{z}$ (since there will be no variables to branch on). For further discussion on nonlinear BnB, see [20].

Specific Details: There are many delicate details in BnB that can critically affect its scalability: for example, the choice of the algorithm for solving the continuous node subproblems, obtaining upper bounds, branching, and tree-search strategies. We discuss our choices below:

- **Subproblem Solver:** The optimal solutions of the continuous optimization subproblems encountered in the course of BnB are typically sparse (see Section 5.4.2 for further discussions). To solve these subproblems, we propose an active-set algorithm, which exploits sparsity by considering a reduced problem restricted to a small subset of groups. Moreover, we share information on the active sets across the BnB tree to speed up convergence (see Section 5.4.2).

- **Upper Bounds:** Better upper bounds can lead to aggressive pruning in the search tree, which can reduce the overall runtime. We obtain the initial upper bound using the approximate algorithms of Section 5.3. As we demonstrate in the experiments, our approximate algorithms typically obtain optimal or near-optimal solutions, making them a good choice to initialize BnB. Moreover, at every node of BnB, we attempt to improve the upper bound by using the sparsity pattern of the solution to the current node's subproblem. More concretely, let $S \subseteq q$ denote the group support of the latter subproblem's solution. Then, we obtain a new upper bound, by restricting optimization to S, i.e., we solve:

$$\min_{\boldsymbol{\theta}} \quad \tilde{\ell}(\boldsymbol{\theta}) + \lambda_1 \sum_{g=1}^{q} \|\boldsymbol{\theta}\|_2 + \lambda_2 \|\boldsymbol{\theta}\|_2^2 \quad \text{s.t.} \quad \boldsymbol{\theta}_{S^c} = \mathbf{0},\ \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\mathrm{U}},\ g \in [q].$$

- **Branching and Search Strategies:** The branching strategy selects the next variable to branch on, while the search strategy decides which unexplored node in the search tree to visit next. Many elaborate strategies for branching and search have been proposed in the literature – see [124] for a survey. When the initial upper bound is of high quality, more aggressive pruning is possible, and simple strategies tend to work relatively well in practice [for example, see the discussion in 55]. Since our approximate algorithms typically return good upper bounds, we rely on simple strategies. For branching, we use maximum fractional branching [20, 124], which branches on the factional variable z_g whose value is closest to 0.5. For search, we use breadth-first search and switch to depth-first search if memory issues are encountered.

Our approach extends our recent work [90] for the best subset selection problem (with a group size of one). We note that there are important differences as the Group ℓ_0 problem involves a different and more challenging optimization formulation. Specifically, the Big-M constraints in (5.26b) translate to second order cones, instead of box-constraints that appear when the group sizes are one. Furthermore, in the group setup, we have a non-smooth term $\sum_{g\in[q]} \|\boldsymbol{\theta}_g\|_2$ in the objective of (5.26). The conic constraints and ℓ_2 norms in our problem require special care when developing the subproblem solver (for example, when reformulating the subproblems in Section 5.4.2 and designing the active set algorithm in Section 5.4.2). It is also worth mentioning that in the simplest case where $\lambda_1 = \lambda_2 = 0$, our solver solves a MISOCP, whereas [90] solves a mixed integer quadratic program.

Relaxation Reformulation

In this section, we study the convex relaxation arising at a node of the BnB search tree. We present a particular reformulation of this problem that leads to (i) useful insights about the sparsity in the solutions of the convex relaxation; and (ii) computational benefits. To simplify the presentation, we will first focus on the root relaxation of (5.26), which is obtained by relaxing all the binary variables in (5.26) to $[0, 1]$.

Note that the root relaxation involves the variables $(\boldsymbol{\beta}, \mathbf{z}, \mathbf{s})$. In Proposition 5.3, we show that the root relaxation can be reformulated in the $\boldsymbol{\beta}$ space, leading to a regularized least squares problem. The associated regularizer can be characterized in terms of the reverse Huber penalty [136] (see also [61]), which is a function $\mathcal{H} : \mathbb{R} \to \mathbb{R}$ defined as follows:

$$\mathcal{H}(t) = \begin{cases} |t| & \text{if } |t| \leq 1 \\ (t^2 + 1)/2 & \text{otherwise.} \end{cases} \tag{5.28}$$

Proposition 5.3. *The root relaxation obtained by relaxing the binary variables in* (5.26) *to*

$[0, 1]$ *is equivalent to:*

$$\min_{\boldsymbol{\theta}} \quad F(\boldsymbol{\theta}) := \tilde{\ell}(\boldsymbol{\theta}) + \sum_{g=1}^{q} \Psi(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) \;\; \textit{s.t.} \;\; \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}, \;\; g \in [q]. \tag{5.29}$$

where $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \lambda_2)$ *and*

$$\Psi(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) := \begin{cases} 2\lambda_0 \mathcal{H}(\sqrt{\lambda_2/\lambda_0}\|\boldsymbol{\theta}_g\|_2) + \lambda_1\|\boldsymbol{\theta}_g\|_2 & \textit{if } \sqrt{\lambda_0/\lambda_2} \leq \mathcal{M}_{\text{U}} \\ (\lambda_0/\mathcal{M}_{\text{U}} + \lambda_1 + \lambda_2\mathcal{M}_{\text{U}})\|\boldsymbol{\theta}_g\|_2 & \textit{if } \sqrt{\lambda_0/\lambda_2} > \mathcal{M}_{\text{U}}. \end{cases}$$

The reformulation in (5.29) eliminates the the conic and Big-M constraints from the root relaxation, at the expense of introducing the non-smooth penalty $\sum_{g=1}^{q} \Psi(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}})$ which is separable across the blocks $\{\boldsymbol{\theta}_g\}_1^q$. Depending on the choices of $\boldsymbol{\lambda}$ and $\mathcal{M}_{\text{U}}$, the penalty Ψ is either the ℓ_2 norm or a combination of the reverse Huber penalty and the ℓ_2 norm. In either case, the penalty is sparsity-inducing. In essence, Problem (5.29) is similar to the Group Lasso problem [182], with two exceptions: (i) Problem (5.29) has the additional constraints: $\|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}, \;\; g \in [q]$, and (ii) when $\sqrt{\lambda_0/\lambda_2} \leq \mathcal{M}_{\text{U}}$, the penalty involves the reverse Huber penalty.

Node relaxations within the BnB tree: The convex relaxation subproblem encountered at a node of the BnB search tree is similar to the root relaxation, except that some of the z_gs are fixed to 0 or 1. The fixed z_gs are determined by the branching decisions made starting from the root until reaching the node. The convex relaxation at a particular node can be reformulated in the $\boldsymbol{\beta}$-space similar to the reformulation of the root relaxation in (5.29), except that: (i) if $z_g = 0$ then the corresponding group should be removed from the objective function; and (ii) if $z_g = 1$, then the penalty $\Psi(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}})$ should be replaced with $\tilde{\Psi}(\boldsymbol{\theta}_g; \boldsymbol{\lambda}) := \lambda_1\|\boldsymbol{\theta}_g\|_2 + \lambda_2\|\boldsymbol{\theta}_g\|_2^2$. More precisely, let $\mathcal{Z}$ and $\mathcal{N}$ be the sets of indices of the z_gs that are fixed to 0 and 1, respectively. Then, the following subproblem is solved at the corresponding node:

$$\min_{\boldsymbol{\theta}} \;\; \tilde{\ell}(\boldsymbol{\theta}) + \sum_{g \in \mathcal{N}^c} \Psi(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) + \sum_{g \in \mathcal{N}} \tilde{\Psi}(\boldsymbol{\theta}_g; \boldsymbol{\lambda}) \;\; \text{s.t.} \;\; \boldsymbol{\theta}_{\mathcal{Z}} = \mathbf{0}, \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}, \;\; g \in [q]. \tag{5.30}$$

In the next section, we develop a scalable algorithm for solving Problem (5.29). The BnB subproblem (5.30) can be solved similarly after accounting for the fixed z_gs.

Active-Set Subproblem Solver

As discussed earlier, a solution to Problem (5.29) is expected to be sparse in $\boldsymbol{\theta}$ (this will be also true for the node sub-problems in the BnB tree). To exploit this sparsity, we use an active-set algorithm: We start by solving Problem (5.29) restricted to a small subset of groups (i.e., the *active set*). After convergence on the active set, we augment the active set with a collection of groups that violate the optimality conditions for the full problem (if any) and then resolve the problem restricted to the augmented active set. The algorithm keeps iterating between solving a reduced optimization problem and augmenting the active set, until the optimality conditions for the full problem are satisfied. Such active-set algorithms have proven to be effective in scaling up the solvers for group Lasso-type problems [for example, see 87]—our usage differs in that we use this active-set strategy within every node of the BnB tree.

Next, we describe our active-set algorithm more formally. Let $\mathcal{A} \subseteq [q]$ be the active set. The algorithm starts by solving (5.29) restricted to the active set, i.e.,

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \quad F(\boldsymbol{\theta}) \;\; \text{s.t.} \;\; \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}, \;\; g \in [q], \;\; \boldsymbol{\theta}_{\mathcal{A}^c} = \mathbf{0}. \tag{5.31}$$

After solving (5.31), we check if $\hat{\boldsymbol{\theta}}$ satisfies the optimality condition for the full problem. Equivalently, for every group $g \in \mathcal{A}^c$, we check if the following holds

$$\mathbf{0} \in \arg\min_{\boldsymbol{\theta}_g} \quad F(\hat{\boldsymbol{\theta}}_1, \ldots, \boldsymbol{\theta}_g, \ldots, \hat{\boldsymbol{\theta}}_q) \;\; \text{s.t.} \;\; \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}. \tag{5.32}$$

Since $\boldsymbol{\theta}_g = \mathbf{0}$ is in the interior of the feasible set, condition (5.32) is equivalent to the zero-subgradient condition: $\mathbf{0} \in \partial_{\boldsymbol{\theta}_g} F(\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_{g-1}, \mathbf{0}, \hat{\boldsymbol{\theta}}_{g+1}, \ldots, \hat{\boldsymbol{\theta}}_q)$, and can be checked in closed form.

We repeat the procedure of solving the restricted subproblem in (5.31) and augmenting

$\mathcal{A}$ with groups that violate (5.32), until there are no more violations. The algorithm is summarized below.

Algorithm 5.3: An Active-set Algorithm for (5.29)

- **Input:** Initial solution $\hat{\boldsymbol{\theta}}$ and initial active set $\mathcal{A}$.
- **Repeat** Steps 1—3 till convergence:
  1. Solve the restricted problem (5.31) to get a solution $\hat{\boldsymbol{\theta}}$.
  2. $\mathcal{V} \leftarrow \{g \in \mathcal{A}^c \mid (5.32) \text{ is violated}\}$.
  3. If $\mathcal{V}$ is empty **terminate**, otherwise[a], $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{V}$.

[a]In some cases, $|\mathcal{V}|$ can be large, which can slow down the solver in Step 1. Thus, if $\mathcal{V}$ has more than K groups, we augment $\mathcal{A}$ with the K groups in $\mathcal{V}$ that have the largest violation (instead of $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{V}$). In our experiments we set $K = 10$. We found this helpful to keep the size of the active set manageable during the course of the algorithm.

Algorithm 5.3 is guaranteed to converge to an optimal solution for Problem (5.29) in a finite number of steps, as there are finitely many groups.

Choice of the active set: The quality of the initial active set $\mathcal{A}$ can have a important effect on the number of iterations in Algorithm 5.3. Due to the choice of our branching rule, the parent and its two child nodes solve similar subproblems; the only difference between these subproblems is that a single z_g is fixed to 0 or 1 in the children. Thus, the solutions and supports of the parent and its children are unlikely to differ by much. We therefore initialize the active set of every node in the BnB tree (except the root) with the support of its parent. For the root node, we initialize the active set with the support of the warm start, obtained from the approximate algorithms that are discussed in Section 5.3.

Solving the restricted subproblem: The convex sub-problem (5.31) in Step 1 has a small active set and can be solved with a variety of optimization algorithms: for example, BCD, proximal gradient methods [22] or an interior point solver (as available in Gurobi). In our experiments, we use the latter due to its good performance in practice.

5.5 Statistical Theory

In this section we derive non-asymptotic prediction and estimation error bounds for the Group ℓ_0 estimators, and compare them to the bounds that have been established for the corresponding Group Lasso-based approaches. We focus on linear regression models in Section 5.5.1 and on nonparametric additive models in Section 5.5.2.

While the arguments used in our proofs extend naturally to the penalized case, we focus on the constrained specifications of the proposed estimators for concreteness. To simplify the presentation, we consider the setting where the model is correctly specified, so that the true regression function is a feasible solution to the corresponding optimization problem. However, our results can be generalized to allow for model misspecification.

We say that a constant is *universal* if it does not depend on other parameters, such as n, q or k. We use the notation $\gtrsim$ and $\lesssim$ to indicate that inequalities $\geq$ and $\leq$, respectively, hold up to positive universal multiplicative factors, and write $\asymp$ when the two inequalities hold simultaneously.

5.5.1 Linear Model

We assume that the observed data follows the model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}$, where $\mathbf{X}$ is deterministic and the elements of $\boldsymbol{\epsilon}$ are independent $N(0, \sigma^2)$ with $\sigma > 0$. We define $k_* = G(\boldsymbol{\beta}^*)$ and refer to $n^{-1}\|\mathbf{X}\widehat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2$ as the prediction error for estimator $\widehat{\boldsymbol{\beta}}$. For simplicity of the presentation we focus on the setting where each group has the same number of T features. Thus, the total number of features, p, is equal to qT. Given $\boldsymbol{\beta} \in \mathbb{R}^p$ and $J \subseteq [q]$, we write $\boldsymbol{\beta}_J$ for the sub-vector of $\boldsymbol{\beta}$ indexed by $\cup_{g\in J}\mathcal{G}_g$. Consider the following definition, in which we use the notation $\|\boldsymbol{\beta}\|_{2,1} = \sum_{g=1}^q \|\boldsymbol{\beta}_g\|_2$.

Definition 5.1. *Given a positive integer k and a constant $c \geq 1$, let*

$$\gamma_k = \min_{\boldsymbol{\beta}\neq\mathbf{0},\, G(\boldsymbol{\beta})\leq k} \frac{\sqrt{k}\|\mathbf{X}\boldsymbol{\beta}\|_2}{\sqrt{n}\|\boldsymbol{\beta}\|_{2,1}} \quad \textit{and} \quad \kappa_{k,c} = \min_{J\subseteq[q],|J|\leq k} \left\{ \min_{\boldsymbol{\beta}\neq\mathbf{0},\, \|\boldsymbol{\beta}_{J^c}\|_{2,1}\leq c\|\boldsymbol{\beta}_J\|_{2,1}} \frac{\sqrt{k}\|\mathbf{X}\boldsymbol{\beta}\|_2}{\sqrt{n}\|\boldsymbol{\beta}_J\|_{2,1}} \right\}.$$

The above definition is most meaningful under the scaling of the features where $\|\mathbf{x}_j\|_2 \asymp \sqrt{n}$ for all j. As we discuss below, constants $\kappa_{k_*,c}^{-1}$, with $c > 1$, appear in the prediction and estimation error bounds for the Group Lasso estimator, while $\gamma_{2k_*}^{-1}$ appears in the estimation error bound for the Group ℓ_0 estimator. The following result establishes a useful relationship for these quantities.

Proposition 5.4. *$\gamma_{2k} \geq \kappa_{k,c}/\sqrt{2}$, for all positive integers k and all $c \geq 1$.*

We study estimator $\widehat{\boldsymbol{\beta}}$, which solves the following optimization problem:

$$\min_{\boldsymbol{\beta}} \quad \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \sum_{g=1}^{q} \mathbf{1}(\boldsymbol{\beta}_g \neq \mathbf{0}) \leq k, \tag{5.33}$$

where k is a fixed parameter that controls the sparsity level. We note that (5.33) is a special case of the cardinality constrained problem considered in Section 5.3.3. Our first result provides the prediction error bound for $\widehat{\boldsymbol{\beta}}$, which holds without any assumptions on the design.

Theorem 5.3. *Let $\delta_0 \in (0,1)$ and suppose that $\widehat{\boldsymbol{\beta}}$ solves optimization problem (5.33) for $k \geq k_*$. Then,*

$$n^{-1}\|\mathbf{X}\widehat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2 \lesssim \sigma^2 k\Big[\frac{T + \log(q/k)}{n}\Big] + \sigma^2\Big[\frac{\log(1/\delta_0)}{n}\Big].$$

with probability at least $1 - \delta_0$.

Letting $\delta_0 = (k/q)^k$ and using Definition 5.1, we derive the following result.

Corollary 5.1. *If $k = k_*$, then*

$$\begin{aligned} n^{-1}\|\mathbf{X}\widehat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2 &\lesssim \sigma^2 k_*\Big[\frac{T + \log(q/k_*)}{n}\Big] \\ \|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_{2,1} &\lesssim \sigma k_*\Big[\frac{T + \log(q/k_*)}{n}\Big]^{1/2}\big[\gamma(2k_*)\big]^{-1} \end{aligned}$$

with probability at least $1 - (k_*/q)^{k_*}$.

We make several observations regarding the established error bounds, comparing them to

the bounds for the Group Lasso estimator, denoted by $\widehat{\boldsymbol{\beta}}_{\mathrm{GL}}$, which replaces the ℓ_0 constraint in Problem (5.33) with a penalty on $\|\boldsymbol{\beta}\|_{2,1}$. To simplify the comparison of the corresponding rates, we focus on the setting where $k = k_*$.

Remark 5.1. *The Group ℓ_0 prediction error rate provided in Corollary 5.1 matches the corresponding optimal prediction error rate established in [111]. The estimation error rate in Corollary 5.1 is also optimal provided that $\gamma_{2k_*}^{-1}$ is bounded by a universal constant under the aforementioned feature scaling $\|\mathbf{x}_j\|_2 \asymp \sqrt{n}$.*

Remark 5.2. *Let $\|\mathbf{x}_j\|_2 \asymp \sqrt{n}$ for all j and assume that $\kappa_{k_*,c}^{-1}$ is bounded by a universal constant for some $c > 1$. Then, the error bounds for the Group Lasso estimator [see, for example, Section 8.3 of 44] are*

$$n^{-1}\|\mathbf{X}\widehat{\boldsymbol{\beta}}_{GL} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2 \lesssim \sigma^2 k_* \Big[\frac{T + \log(q)}{n}\Big] \quad \textit{and} \quad \|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_{2,1} \lesssim \sigma k_* \Big[\frac{T + \log(q)}{n}\Big]^{1/2}. \tag{5.34}$$

The Group ℓ_0 rates discussed in Remark 5.1 are better than those in display (5.34), because they replace the $\log(q)$ term with $\log(q/k_*)$. Moreover, in view of Proposition 5.4, the assumption on γ_{2k_*} in Remark 5.1 is weaker than the Group Lasso assumption on $\kappa_{k_*,c}$. Finally, the Group ℓ_0 prediction error bound holds without any assumptions on the design.

The last observation represents an important non-trivial advantage of ℓ_0-based approaches over Lasso-type methods. [190] provide examples of design matrices in the usual linear regression context for which the Lasso prediction error is lower-bounded by a constant multiple of $1/\sqrt{n}$, generally leading to a much larger prediction error than the one for the ℓ_0-based method.[5]

Remark 5.3. *One advantage of estimator (5.33) is that tuning parameter k directly controls the sparsity of the proposed estimator. In particular, the $\widehat{\boldsymbol{\beta}}$ that achieves the bounds in Corollary 5.1 satisfies $G(\widehat{\boldsymbol{\beta}}) \leq k_*$. On the other hand, the $\widehat{\boldsymbol{\beta}}_{GL}$ that achieves bounds (5.34) is typically much more dense. The following inequality, which holds with high probability, is*

[5]The lower-bound applies to a wide class of coordinate-separable M-estimators, including local optima of nonconvex regularizers such as SCAD and MCP.

provided in [111]:

$$G(\widehat{\boldsymbol{\beta}}_{GL}) \leq \Big[\frac{64\phi_{\max}}{\kappa_{k_*,3}}\Big]k_*.$$

Here, $\phi_{\max}$ is the maximum eigenvalue of $\mathbf{X}^\top\mathbf{X}/n$. Thus, the right-hand side is at least $64k_*$.

Remark 5.4. *Theorem 5.3 and Corollary 5.1 can also apply to approximate solutions, obtained after an early termination of the MIP solver. In such settings, the solver provides the current lower and upper bounds, LB and UB, on the value of the objective. If the corresponding optimality gap satisfies $(UB - LB)/LB \lesssim \sigma^2 k_*[T + \log(q/k_*)]/n$, then the bounds in Corollary 5.1 also hold for the approximate solution.*

An attractive feature of Theorem 5.3 is that the uncertainty parameter δ_0 is independent of the tuning parameter k. This allows us to control the expected prediction error, as we demonstrate in the following result.

Corollary 5.2. *Under the conditions of Theorem 5.3,*

$$\mathbb{E}\|\mathbf{X}\widehat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2 \lesssim \sigma^2 k\big[T + \log(q/k)\big].$$

An application of Definition 5.1 yields a corresponding bound on the expected estimation error.

5.5.2 Nonparametric Additive Model

We study the performance of the proposed approach in the deterministic design setting. We write $\|\cdot\|_{L_2}$ for the L_2 norm of a real-valued function on $[0,1]$. Using the notation in Section 2.2, we let $\mathcal{C}_j = \mathcal{C}$ for all j and focus on the case where $\mathcal{C}$ is an L_2-Sobolev space:

$$\mathcal{C} = \big\{g : [0,1] \mapsto \mathbb{R},\ \|g\|_{L_2} + \|g^{(m)}\|_{L_2} < \infty\big\} \quad \text{and} \quad \text{Pen}(g) = \|g^{(m)}\|_{L_2}.$$

We define $\mathcal{C}_{\text{gr}} = \{f : [0,1]^q \mapsto \mathbb{R},\ f(\mathbf{x}) = \sum_{j=1}^q f_j(x_j),\ f_j \in \mathcal{C}\}$ as the corresponding space of additive functions. We associate each $f \in \mathcal{C}_{\text{gr}}$ with the vector $\mathbf{f} = \sum_{j=1}^q \mathbf{f}_j$, where

$\mathbf{f}_j = \big(f_j(x_{1j}), ..., f_j(x_{nj})\big)$, and let

$$G(f) = \sum_{j=1}^{q} \mathbf{1}(\mathbf{f}_j \neq \mathbf{0}), \qquad \text{Pen}_{\text{gr}}(f) = \sum_{j=1}^{q} \text{Pen}(f_j).$$

We focus on the estimator that solves the following optimization problem:

$$\min_{f \in \mathcal{C}_{\text{gr}}} \ \|\mathbf{y} - \mathbf{f}\|_n^2 + \lambda_n \text{Pen}_{\text{gr}}(f) \quad \text{s.t.} \quad G(f) \leq k, \tag{5.35}$$

where $\|\cdot\|_n$ denotes the Euclidean norm divided by $\sqrt{n}$.[6] To ensure identifiability of the representation $f(\mathbf{x}) = \sum_{j=1}^{q} f_j(x_j)$, additional restrictions are typically imposed. For example, a popular method is to separate out the constant term and require that $\sum_{i=1}^{n} f_j(x_{ij}) = 0$ for each j. Here we follow the approach of [158] and avoid specifying a particular set of restrictions. We treat every representation of f as equivalent, with the understanding that one particular representation is used when evaluating properties of the components, such as $\|\mathbf{f}_j\|_n$.

We are interested in comparing estimator (5.35), denoted by $\widehat{f}$, with the widely popular Group Lasso-based approach, which replaces the ℓ_0 constraint in Problem (5.35) with a penalty on $\sum_{j=1}^{q} \|\mathbf{f}_j\|_n$. Theoretical properties of the latter approach have been investigated extensively [see, for example, 120, 102, 148, 157, 183, 158, and the references therein]. To compare the error bounds for the two estimators, we need the following definition.

Definition 5.2. *Given a positive integer k, a constant $\xi \in (1, \infty]$ and an index set $J \subseteq [q]$, let*

$$A_{k,\xi} = \{f \in \mathcal{C}_{gr} : \ \sum_{j=1}^{q} \|\mathbf{f}_j\|_n \neq 0,\ G(f) \leq k,\ 2n^{-m/(2m+1)} \text{Pen}_{\text{gr}}(f) \leq (\xi - 1) \sum_{j=1}^{q} \|\mathbf{f}_j\|_n\}$$

$$B_{J,\xi} = \{f \in \mathcal{C}_{gr} : \ \sum_{j=1}^{q} \|\mathbf{f}_j\|_n \neq 0,\ \sum_{j \notin J} \|\mathbf{f}_j\|_n + n^{-m/(2m+1)} \text{Pen}_{\text{gr}}(f) \leq \xi \sum_{j \in J} \|\mathbf{f}_j\|_n\}$$

$$\psi(k, \xi) = \min_{f \in A_{k,\xi}} \frac{\sqrt{k}\|\mathbf{f}\|_n}{\sum_{j=1}^{q} \|\mathbf{f}_j\|_n} \quad \textit{and} \quad \phi(k, \xi) = \min_{J \subseteq [q], |J| \leq k} \left\{ \min_{f \in B_{J,\xi}} \frac{\sqrt{k}\|\mathbf{f}\|_n}{\sum_{j \in J} \|\mathbf{f}_j\|_n} \right\}.$$

[6] We acknowledge the notational inconsistency when $n \leq 2$.

As we discuss below, constants $\phi(2k,\xi)^{-1}$ appear in the error bounds for the Group Lasso-based approach, while constants $\psi(k,\xi)^{-1}$ appear in some of the bounds that we establish for $\widehat{f}$. The following result establishes a useful relationship for these quantities.

Proposition 5.5. *For all positive integers k and all $\xi \in (1,\infty]$, $\psi(2k,\xi) \geq \phi(k,\xi)/\sqrt{2}$.*

We assume that the observed data follows the model $\mathbf{y} = \mathbf{f}^* + \boldsymbol{\epsilon}$, where $f^* \in \mathcal{C}_{\text{gr}}$, and the elements of $\boldsymbol{\epsilon}$ are independent $N(0,\sigma^2)$ with $\sigma > 0$. We refer to $\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2$ as the prediction error for estimator $\widehat{f}$. We write $r_n = n^{-m/(2m+1)}$, suppressing the dependence on m for notational simplicity, noting that r_n^2 is the optimal prediction error rate in the univariate regression setting where $f^* \in \mathcal{C}$. For example, in the case where $\mathcal{C}$ is the second order Sobolev space, which corresponds to $m = 2$, the above rate is $r_n^2 = n^{-4/5}$. We define $\alpha = 1/(4m+2)$ and note that $\alpha = 1/10$ when $m = 2$. The next result, in which we treat $m \geq 1$ as a fixed integer, establishes prediction error bounds for the proposed approach.

Theorem 5.4. *Let $k_* = G(f^*)$ and consider optimization Problem (5.35) with $k \geq k_*$. There exists a universal constant c_1, such that if $\lambda_n \geq c_1\sigma\big[k^{2\alpha}r_n^2 + k^{\alpha}r_n\sqrt{\log(eq/k)/n}\big]$, then*

$$\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 \lesssim \sigma^2 k\Big[k^{2\alpha}r_n^2 + \frac{\log(eq/k)}{n}\Big] + \lambda_n \text{Pen}_{\text{gr}}(f^*) \tag{5.36}$$

with probability at least $1-(k/q)^k$. Furthermore, for every $\xi \in (1,\infty]$, there exists a finite constant c_2, which depends only on ξ, such that if $\lambda_n \geq c_2\sigma\big[r_n^2 + r_n\sqrt{\log(q)/n}\big]$, then

$$\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 \lesssim \sigma^2 k\Big[r_n^2 + \frac{\log(q)}{n}\Big]\big[\psi(2k,\xi)\big]^{-2} + \lambda_n \text{Pen}_{\text{gr}}(f^*) \tag{5.37}$$

with probability at least $1-1/q$.

We make the following observations regarding the established error bounds. To simplify the comparison of the error rates, we focus on the setting where $k = k_*$ and $\text{Pen}_{\text{gr}}(f^*) \asymp \sigma k_*$. The last relationship holds, for example, when the scaled roughness of each nonzero component, $\text{Pen}(f_j^*)/\sigma$, is bounded above and below by positive universal constants.

Remark 5.5. *The expression in error bound (5.36) is optimized for the setting where*

$\mathrm{Pen}_{\mathrm{gr}}(f^*) \asymp \sigma k$. *However, as we show in the proof, the bound can be improved when* σk *and* $\mathrm{Pen}_{\mathrm{gr}}(f^*)$ *have different orders of magnitude.*

Remark 5.6. *The prediction error rate provided in (5.37) is analogous to the rate established in [158] for the Group Lasso-based approach*[7]*, however, the latter rate replaces* $\psi(2k_*, \xi)^{-2}$ *with* $\phi(k_*, \xi)^{-2}$*. By Proposition 5.5, the former rate is at least as good as the latter, with a potential improvement due to the additional* ℓ_0 *group sparsity requirement in the definition of* ψ*. If for some fixed* $\xi > 1$ *quantity* $\psi(2k_*, \xi)^{-1}$ *is bounded by a universal constant, then inequality (5.37) yields the following prediction error rate:*

$$\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 \lesssim \sigma^2 k_* \Big[r_n^2 + \frac{\log(q)}{n} \Big].$$

This rate matches the one established in [158] for the Group Lasso-based approach under an analogous (but somewhat stronger) assumption on $\phi(k_*, \xi)^{-1}$*.*

Remark 5.7. *Bound* (5.36) *yields the following error rate without imposing assumptions on the design:*

$$\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 \lesssim \sigma^2 k_* \Big[k_*^{2\alpha} r_n^2 + \frac{\log(eq/k_*)}{n} \Big].$$

If $k_* \lesssim 1$ *or* $k_*^{2\alpha} r_n^2 \lesssim \log(eq/k_*)/n$*, then the above expression can be upper-bounded by*

$$\sigma^2 k_* \big[r_n^2 + \frac{\log(eq/k_*)}{n} \big].$$

Thus, $\widehat{f}$ *achieves the corresponding minimax lower bound on the prediction error [148, 157, 158].*

Remark 5.8. *When* $q = k_*$*, the prediction error rate given by bound (5.36) is* $k_*^{1+1/(2m+1)} r_n^2$*, which improves over the corresponding* $k_*^{1+3/(2m+1)} r_n^2$ *rate*[8] *derived in [109]. In particular, when* $m = 2$*, the former rate is* $k_*^{6/5} n^{-4/5}$*, while the latter is* $k_*^{8/5} n^{-4/5}$*. The improvement in*

[7] To the best of our knowledge, the bounds in [158] are overall the strongest in the literature for the Group Lasso-based approach, due to the relative weakness of the imposed conditions: see the discussion in Remark 12 of [158].

[8] Theorem 1 in [109] treats the number of predictors ($q = k_*$) as fixed and omits it from the expression for the error rate. However, an examination of the proof of their Theorem 1 and the entropy bound in their Lemma A.1, which explicitly accounts for the number of predictors, reveals the effect of the dimension k_*.

the rate is a consequence of the more refined entropy bounds derived in our proofs.

Remark 5.9. *In the special case of $m = 2$ and $k_* \lesssim 1$, bound (5.36) yields the prediction error rate of $n^{-4/5} + \log(q)/n$, which matches the optimal univariate rate of $n^{-4/5}$ when $\log(q) \lesssim n^{1/5}$.*

Remark 5.10. *If for some fixed $\xi > 1$ quantity $\psi(2k_*, \xi)^{-1}$ is bounded by a universal constant, then a direct consequence of Theorem 5.4 is the following estimation error rate:*

$$\sum_{j=1}^{q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n \lesssim \sigma k_* \Big[r_n + \sqrt{\frac{\log(q)}{n}} \Big].$$

5.6 Experiments

We present experiments that shed light on the practical performance of our proposals compared to the state of the art. In Section 5.6.1, we investigate the statistical properties of our algorithms for the Group ℓ_0 problem. In Section 5.6.2, we present computation times of our MIP algorithm. Section 5.6.3 investigates nonparametric sparse additive models.

5.6.1 Grouped variable selection

We consider both synthetic and real datasets in our experiments, as discussed below.

Synthetic data generation. The underlying model is $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}$, where $\boldsymbol{\beta}^* \in \mathbb{R}^p$ has q groups, all with the same size. Once we generate $\mathbf{X}$ (see below), every column is standardized to have unit ℓ_2-norm. The errors $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2), i = 1, \ldots, n$, are independent of $\mathbf{X}$, and σ^2 is chosen to achieve a desired signal-to-noise ratio (SNR)[9]. We note that the SNR values in our experiments are sufficiently high to make the true model support recovery possible.

Two different types of $\mathbf{X}$ are considered: (a) `example=1`: We first generate group representatives $\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q \sim \text{MVN}_q(0, \boldsymbol{\Sigma})$, where, $\boldsymbol{\Sigma}_{q\times q} = ((\sigma_{ij}))$, with $\sigma_{ij} = \rho^{|i-j|}$. Given a $\boldsymbol{\gamma}_g$, the covariates $\mathbf{x}_j, j \in \mathcal{G}_g$ are generated by adding independent Gaussian noise to a scalar

[9]For a generative model of the form $y_i = \mu_i + \epsilon_i$, we define SNR $= \text{Var}(\mu)/\text{Var}(\epsilon)$.

multiple of $\boldsymbol{\gamma}_g$, to achieve pairwise correlation of 0.9 within the group. (b) `example=2`: Here we take $\mathbf{X} \sim \mathrm{MVN}_p(0, \boldsymbol{\Sigma})$, where $\sigma_{ij} = \rho$, for all $i \neq j$, with $\sigma_{jj} = 1$ for all j.

To generate the true population regression coefficients, the k_* nonzero groups are taken to be equally spaced in $\{1, \ldots, q\}$. All the nonzero entries of $\boldsymbol{\beta}^*$ are drawn independently from a standard Gaussian distribution.

Competing algorithms and tuning. In the experiments of this section, we focus on the Group ℓ_0 problem defined in (5.1), and study the performance of our algorithms. We compare against the following state-of-the-art grouped variable selection methods: Group Lasso (based on $\ell_{2,1}$ regularization), Group MCP, and Group SCAD – these estimators are computed by using the R package `grpreg` [42]. For synthetic data, we construct a separate validation set with a fixed design. We tune the parameters of the different problems to minimize the prediction error on the validation set. Specifically, for each of Group ℓ_0 and Group Lasso, we tune the regularization parameter over a (one-dimensional) grid with 100 values. For MCP and SCAD, we tune the first parameter λ over a grid with 100 values, and leave the second parameter γ to its default value in `grpreg`.

Performance measures. Given an estimator $\hat{\boldsymbol{\beta}}$, we consider the following performance measures:

- **True Positives (TP)**: The number of nonzero groups that are in both $\hat{\boldsymbol{\beta}}$ and $\boldsymbol{\beta}^*$.
- **False Positives (FP)**: The number of nonzero groups in $\hat{\boldsymbol{\beta}}$ but not in $\boldsymbol{\beta}^*$
- **Recovery F1 Score**: The harmonic mean of precision and recall, i.e., F1 Score $= 2PR/(P+R)$, where $P = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$ is precision and $R = \mathrm{TP}/k_*$ is recall. We note that an F1 Score of 1 implies perfect support recovery.
- **Test MSE**: This is defined as $\frac{1}{n}\|\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2$.

Statistical performance for varying number of observations

In this experiment, we study the effect of varying the number of observations n on the performance of Group ℓ_0 and other state-of-the-art group regularizers (Group Lasso, MCP, and SCAD). We obtain approximate estimators to the Group ℓ_0 problem using Algorithms 5.1 and 5.2 (with $m = 1$). We generate 10 datasets having exponentially decaying correlation (i.e., under `example=1`) with a correlation parameter $\rho = 0.9$, $p = 5000$, a group size of 4, number of nonzero groups $k_* = 25$, and SNR $= 10$. This setting is relatively difficult for recovery as each group is highly correlated with a few others. We report the average performance measures over the 10 datasets in Figure 5-1.

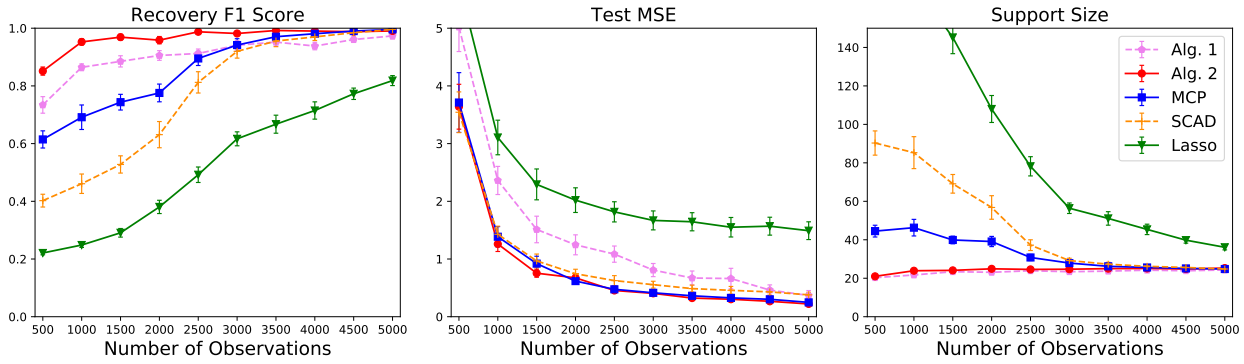


Figure 5-1: Performance measures for varying number of observations on a synthetic dataset with highly correlated features. Alg. 1 and Alg. 2 are our proposed algorithms. Here, "Lasso" is a shorthand for Group Lasso, we use the same convention for SCAD, MCP.

Figure 5-1 shows that Algorithm 5.2 notably outperforms the other methods in terms of variable selection; it perfectly recovers the support for $n \approx 2000$. Group MCP and SCAD require roughly 4500 observations to recover the true support, whereas Group Lasso does not recover the support even when $n = p$. Moreover, Algorithms 5.1 and 5.2 attain the smallest support sizes for any n, whereas the other methods require much larger supports, especially for small n. Algorithm 5.2 has the lowest test MSE for all n. The MSE of MCP matches that of Algorithm 5.2 in most of the cases, while the other methods lag behind. We also note that there is a gap between the MSE of Algorithms 5.1 and 5.2. This difference is likely due to Algorithm 5.2 doing a better job in optimization.

Statistical performance on high-dimensional instances

We compare the performance of the different methods under two high-dimensional settings. In both settings, we generate data with constant correlation (i.e., under `example=2`) and SNR = 10. Below is a description of the settings:

- Setting 1: $\rho = 0.9, n = 1000, p = 100,000, k = 10$, and a group size of 10.
- Setting 2: $\rho = 0.3, n = 1000, p = 100,000, k = 20$, and a group size of 4.

For each setting, we generate 10 random training and validation datasets, on which we train and tune the algorithms. To ensure a fair comparison in terms of running time, we solve the Group ℓ_0 problem approximately using Algorithm 5.2 (with $m = 1$), which typically has the same order of running time (seconds in this case) as the other group selection methods considered here. We report the averaged results for Settings 1 and 2 in Table 5.1.

Table 5.1: Performance measures for Setting 1 (top panel) and Setting 2 (bottom panel). Means are reported along with their standard errors.

	Algorithm	$\|\hat{\boldsymbol{\beta}}\|_0$	TP	FP	MSE	$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_\infty$
Setting 1	Group ℓ_0	98.0 (2.5)	9.7 (0.2)	0.1 (0.1)	7.8 (1.7)	0.8 (0.1)
	Group Lasso	2108 (222.6)	10.0 (0.0)	200.8 (22.3)	19.8 (3.4)	1.4 (0.12)
	Group MCP	294 (44.3)	10.0 (0.0)	19.4 (4.4)	11.7 (3.2)	0.95 (0.17)
	Group SCAD	637 (98.6)	10.0 (0.0)	53.7 (9.9)	18.4 (5.4)	1.2 (0.22)
Setting 2	Group ℓ_0	79.2 (1.3)	19.6 (0.2)	0.2 (0.1)	0.97 (0.08)	0.35 (0.03)
	Group Lasso	1139.2 (63.3)	19.9 (0.1)	264.9 (15.7)	4.42 (0.29)	0.67 (0.03)
	Group MCP	146.0 (15.6)	19.8 (0.1)	16.7 (3.9)	1.07 (0.07)	0.38 (0.03)
	Group SCAD	300.0 (36.3)	20.0 (0.0)	55.0 (9.1)	1.26 (0.10)	0.45 (0.05)

Under both settings, Group ℓ_0 selects significantly smaller support sizes and false positives than other methods, and is more consistent across the replications (as evidenced by the small standard error). For example, in Table 5.1 (top), Group ℓ_0 has a support size which is roughly 20 times smaller than the one for the Lasso and 3 times smaller than one for MCP. For few of the instances, one true positive is missed in Group ℓ_0, but the difference with the other methods is marginal. In terms of MSE and the estimation error (i.e., $\|\boldsymbol{\beta} - \boldsymbol{\beta}^*\|_\infty$), Group ℓ_0 appears to outperform the other methods, with the differences being most pronounced

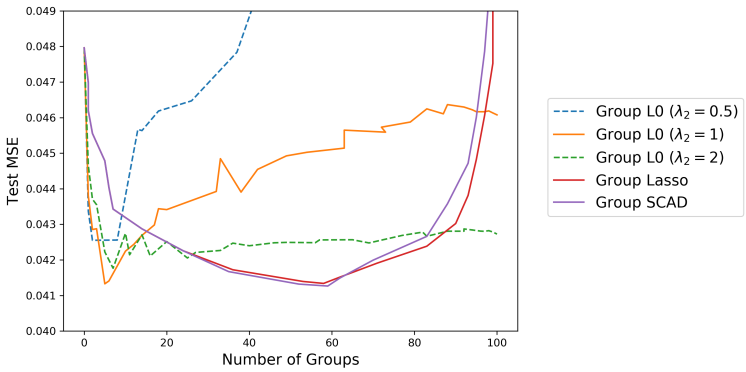


Figure 5-2: Test MSE on the Amazon Reviews dataset ($n = 3500$, $p = 3368$, and $q = 100$). For Group ℓ_0, we consider additional ridge regularization and vary the corresponding regularization parameter $\lambda_2 \in \{0.5, 1, 2\}$.

in the high correlation setting of Table 5.1 (top). This aligns with the results in Figure 5-1, where we saw that Group ℓ_0 leads to important improvements when features are highly correlated and n is small.

Real data

We study the performance of the different methods on the Amazon Reviews dataset [86]. After preprocessing, the dataset consists of 3482 predictors divided into 100 groups. We use 3500 and 2368 observations for training and testing, respectively. Additional details on the dataset and preprocessing are discussed in Appendix 5.B. On this dataset, we fit regularization paths for Group ℓ_0, Lasso, and SCAD[10]. For Group ℓ_0, we use an additional ridge regularization term[11] and consider $\lambda_2 \in \{0.5, 1, 2\}$. In Figure 5-2, we plot the test MSE at different sparsity levels. The results indicate that the lowest MSE is roughly the same for Group ℓ_0 ($\lambda_2 = 1$), Lasso, and SCAD; with Group ℓ_0 having a clear advantage in terms of the support size. Specifically, Group ℓ_0 with $\lambda_2 = 1$ attains the lowest MSE at 5 groups whereas Group Lasso and SCAD require around 60 groups to achieve a similar MSE performance.

In Appendix 5.B, we report results on another real dataset; and our conclusions are qualitatively similar to the example in Figure 5-2.

[10]We also tried group MCP, but the solver faced numerical problems—hence, their results are not reported.
[11]This is found to be useful here due to high feature correlations within a group.

5.6.2 MIP-based Global Optimality Certificates: Timing Comparisons

Here, we compare the running time of our BnB solver with Gurobi for obtaining globally optimal solutions. We generate synthetic data under `example=2`, and we study the effect of the number of predictors p on the running time. Specifically, we vary $p \in \{10^3, 10^4, 10^5, 10^6, 5 \times 10^6\}$ and fix the other data generation parameters as follows: group size of 10, $n = 10^3$, $\rho = 0.1$, $k_* = 5$, SNR $= 10$, and set all nonzero coefficients in $\boldsymbol{\beta}^*$ to 1. We solve the MIP in (5.26) to optimality, for two cases: **(i)** with ridge regularization ($\lambda_2 > 0$) and **(ii)** without ridge regularization ($\lambda_2 = 0$). In both cases, we fix $\lambda_1 = 0$. For case (i), we choose (λ_0, λ_2) so that the solution obtained has k_* nonzero groups and minimizes the ℓ_2 estimation error. More formally, for a fixed choice of (λ_0, λ_2), let $\boldsymbol{\theta}(\lambda_0, \lambda_2)$ denote a solution of (5.26). Then, we choose the parameters of case (ii) as follows:

$$(\lambda_0^*, \lambda_2^*) \in \underset{(\lambda_0, \lambda_2)}{\arg\min} \|\boldsymbol{\theta}(\lambda_0, \lambda_2) - \boldsymbol{\beta}^*\|_2 \;\text{ s.t. }\; G(\boldsymbol{\theta}(\lambda_0, \lambda_2)) = k_*.$$

We estimate $(\lambda_0^*, \lambda_2^*)$ by running Algorithm 5.2 on a two-dimensional grid with $\lambda_0 \in \{10^3, 2 \times 10^3, \ldots, 10^4\}$ and $\lambda_2 \in \{10^{-5}, 10^{-4}, \ldots, 10^5\}$. For case (ii), we choose λ_0 so that the corresponding solution has k_* nonzero groups. Let S^* be the support of the true solution $\boldsymbol{\beta}^*$, and let $\hat{\boldsymbol{\beta}}$ be the solution obtained by solving $\min_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta})$ s.t. $\boldsymbol{\beta}_{(S^*)^c} = 0$. Then, in both cases, we set $\mathcal{M}_{\text{U}}$ to $\max_{g \in [q]} \|\hat{\boldsymbol{\beta}}_g\|_2$. For the two solvers, we set the optimality gap[12] to 1% and use a warm start obtained from Algorithm 5.2. The running times were measured on a cluster with CentOS 7. Each job (i.e., a single run of a solver over one dataset) was allocated 4 cores of an Intel Xeon Gold 6130 CPU @ 2.10GHz processor and up to 120 GB of RAM. For each job, we set a time limit of 24 hours.

In Table 5.2, we report the running time (in seconds) for cases (i) and (ii). In both cases, the results indicate that our BnB can solve instances with $p = 5 \times 10^6$ in the order of minutes to hours, whereas Gurobi cannot solve the problem beyond $p = 10^3$ within the 24-hour time limit. Specifically, for $p \geq 10^4$, Gurobi's optimality gap is 100%. The reason behind this

[12]Given an upper bound UB and a lower bound LB, the optimality gap is defined as (UB-LB)/UB.

Table 5.2: Running time in seconds for solving Problem (5.26) to optimality. A dash (-) indicates that Gurobi cannot solve the problem in 24 hours and has an optimality gap of 100% upon termination.

p	Case (i): $\lambda_2 = \lambda_2^*$		Case (ii): $\lambda_2 = 0$	
	Ours	Gurobi	Ours	Gurobi
10^3	96	24223	373	8737
10^4	199	-	466	-
10^5	231	-	1136	-
10^6	386	-	1628	-
5×10^6	1922	-	11627	-

large gap is that Gurobi cannot solve the root relaxation in the 24-hour time limit, so the best lower bound upon termination is 0. The running times for our BnB solver in case (i) are lower than case (ii), and this can be attributed the perspective reformulation which exploits the presence of the ridge regularizer to speed up computation. It is also worth mentioning that our implementation of BnB is a prototype that does not exploit parallelism (commercial solvers like Gurobi exploit parallelism). Parallelizing our BnB implementation is expected to make it faster, especially on difficult instances where the search tree is large. In Appendix 5.B, we report the running times of our BnB and Gurobi for different choices of $\mathcal{M}_{\text{U}}$.

5.6.3 Nonparametric Additive Models

We study an expanded version of the popular Boston Housing dataset[13] as an application of our MIP framework to ℓ_0-sparse additive modeling. The dataset consists of 13 covariates. To get a better idea about the performance in the presence of irrelevant covariates, we augmented the data with 50 irrelevant covariates. Specifically, we selected 5 covariates uniformly at random. For each selected covariate, we randomly permuted the entries of the covariate vector and augmented the data with the permuted vector—we repeated this step 10 times. This led to 63 covariates in total. We randomly sampled 406 observations for training and 50 observations for validation, and we standardized the response and the covariates. We predict house price using the 63 covariates.

[13]The dataset was downloaded from `https://archive.ics.uci.edu/ml/datasets/Housing`.

We compare the performance of sparse additive models based on Group ℓ_0 and Group Lasso. In both approaches, we used B-splines of degree 3 for the basis functions, with 10 knots equi-spaced in the covariates. For the Group ℓ_0-based approach, we used formulation (5.6) and tuned λ over a grid of 100 values between 10^{-5} and 10^{-2} (equi-spaced on a logarithmic scale). We obtained the Group Lasso-based approach by relaxing all the binary variables in the MIP formulation of (5.6) to the interval $[0, 1]$, and we tuned λ over a grid of 100 values ranging from 10^{-4} to 1 (equi-spaced on a logarithmic scale). In Figure 5-3, we plot the test MSE versus the number of nonzeros, for each of the two models. The results indicate that the Group ℓ_0-based approach achieves the minimum test MSE at 7 nonzeros, whereas the Group Lasso-based method achieves its minimum MSE at around 60 nonzeros (without matching the performance of Group L0).

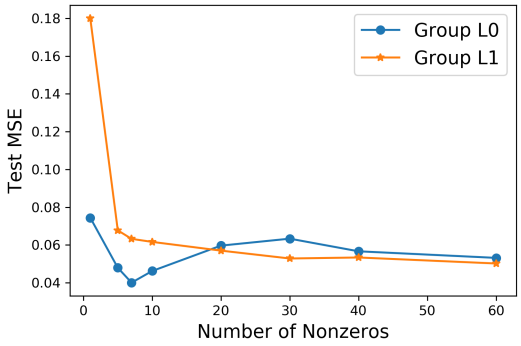


Figure 5-3: Test MSE versus the number of nonzeros on the Boston Housing dataset (with additional noisy covariates).

5.A Appendix: Proofs and Technical Details

Convex Relaxation of Problem (5.7)

Consider Problem (5.7) and suppose that the solution to this problem is bounded. Moreover, we assume that the ℓ_2-norms of every group $\mathbf{f}_j$ satisfies: $\|\mathbf{f}_j\|_2 \leq \mathcal{M}_{\text{U}}$. Then it follows that the problem is equivalent to:

$$\min \quad \|\mathbf{y} - \sum_{j=1}^{q} \mathbf{f}_j\|_2^2 + \lambda_0 \sum_{j\in[q]} z_j + \lambda \sum_{j=1}^{q} \|\mathbf{f}_j\|_{C_j} \quad \text{s.t.} \quad \|\mathbf{f}_j\|_2 \leq \mathcal{M}_{\text{U}} z_j, z_j \in \{0,1\}, j \in [q]. \quad (5.38)$$

Relaxing the z_j's in the above to $[0,1]$, leads to the following formulation:

$$\min \quad \|\mathbf{y} - \sum_{j=1}^{q} \mathbf{f}_j\|_2^2 + \lambda \sum_{j=1}^{q} \|\mathbf{f}_j\|_{C_j} + \lambda_1 \sum_{j=1}^{q} \|\mathbf{f}_j\|_2 \quad \text{s.t.} \quad \|\mathbf{f}_j\|_2 \leq \mathcal{M}_{\text{U}} \tag{5.39}$$

where $\lambda_1 := \frac{\lambda_0}{\mathcal{M}_{\text{U}}}$. Next, we (i) drop the constraints in the above, and (ii) rewrite the resulting problem as follows:

$$\Gamma_1 := \min \quad \|\mathbf{y} - \sum_{j=1}^{q} \mathbf{f}_j\|_2^2 + \lambda(\sum_{j=1}^{q} \|\mathbf{f}_j\|_{C_j} + \frac{\lambda_1}{\lambda} \sum_{j=1}^{q} \|\mathbf{f}_j\|_2). \tag{5.40}$$

Note that (5.40) is a relaxation of (5.38) (and consequently of (5.7)). Now, using the fact that

$$(\|\mathbf{f}_j\|_{C_j} + \frac{\lambda_1}{\lambda}\|\mathbf{f}_j\|_2)/\sqrt{2} \leq \sqrt{\|\mathbf{f}_j\|_{C_j}^2 + \left(\frac{\lambda_1}{\lambda}\right)^2 \|\mathbf{f}_j\|_2^2},$$

it follows that the following

$$\Gamma_2 := \min \quad \|\mathbf{y} - \sum_{j=1}^{q} \mathbf{f}_j\|_2^2 + \sqrt{2}\lambda \sum_{j=1}^{q} \left(\sqrt{\|\mathbf{f}_j\|_{C_j}^2 + \left(\frac{\lambda_1}{\lambda}\right)^2 \|\mathbf{f}_j\|_2^2} \right), \tag{5.41}$$

is an upper bound to Problem (5.40) (with the tuning parameters kept fixed). Note that Problem (5.41) is indeed the penalty considered in [120], with the choice of $\text{Pen}(f_j) = \sqrt{\|\mathbf{f}_j\|_{C_j}^2 + \lambda' \|\mathbf{f}_j\|_2^2}$, where λ' is appropriately chosen to match (5.41).

We note that the penalty chosen in formulation (5.40) is similar to the penalty considered in [148], wherein the authors consider an RKHS framework with penalization:

$$\lambda \sum_{j=1}^{q} \sqrt{\boldsymbol{\beta}_j' \mathbf{K}^j \boldsymbol{\beta}_j} + \lambda' \sum_{j=1}^{q} \|\mathbf{K}^j \boldsymbol{\beta}_j\|_2,$$

where $\mathbf{K}^j$ indicates the kernel basis matrix for the jth coordinate.

Proof of Theorem 5.1

The following lemma shows that there is a sufficient decrease in the objective after every group update in Algorithm 5.1. The result of this lemma will be used in the proof of Theorem 5.1.

Lemma 5.1. *(Sufficient Decrease) The sequence of iterates $\{\boldsymbol{\theta}^l\}$ in Algorithm 5.1 satisfies the following for every l and $g = 1 + (l \mod q)$:*

$$h(\boldsymbol{\theta}^l) - h(\boldsymbol{\theta}^{l+1}) \geq \frac{\hat{L}_g - L_g}{2} \|\boldsymbol{\theta}_g^l - \boldsymbol{\theta}_g^{l+1}\|_2^2. \tag{5.42}$$

Proof of Lemma 5.1. Fix some $l \geq 0$ and let $g = 1 + (l \mod q)$. Applying (5.11) to $(\boldsymbol{\theta}^{l+1}, \boldsymbol{\theta}^l)$ and adding $\Omega(\boldsymbol{\theta}^{l+1})$ to both sides, we get:

$$h(\boldsymbol{\theta}^{l+1}) \leq \ell(\boldsymbol{\theta}^l) + \langle \nabla_{\boldsymbol{\theta}_g} \ell(\boldsymbol{\theta}^l), \boldsymbol{\theta}_g^{l+1} - \boldsymbol{\theta}_g^l \rangle + \frac{L_g}{2} \|\boldsymbol{\theta}_g^{l+1} - \boldsymbol{\theta}_g^l\|_2^2 + \Omega(\boldsymbol{\theta}^{l+1}). \tag{5.43}$$

By rewriting the term $\frac{L_g}{2} \|\boldsymbol{\theta}_g^{l+1} - \boldsymbol{\theta}_g^l\|_2^2$ in the above as $\frac{L_g - \hat{L}_g}{2} \|\boldsymbol{\theta}_g^{l+1} - \boldsymbol{\theta}_g^l\|_2^2 + \frac{\hat{L}_g}{2} \|\boldsymbol{\theta}_g^{l+1} - \boldsymbol{\theta}_g^l\|_2^2$ and regrouping terms, we get:

$$h(\boldsymbol{\theta}^{l+1}) \leq \tilde{g}(\boldsymbol{\theta}^{l+1}; \boldsymbol{\theta}^l) + \frac{L_g - \hat{L}_g}{2} \|\boldsymbol{\theta}_g^{l+1} - \boldsymbol{\theta}_g^l\|_2^2. \tag{5.44}$$

But $\tilde{g}(\boldsymbol{\theta}^{l+1}; \boldsymbol{\theta}^l) \leq \tilde{g}(\boldsymbol{\theta}^l; \boldsymbol{\theta}^l)$ (by the definition of $\boldsymbol{\theta}^{l+1}$ in (5.13)). Moreover, $\tilde{g}(\boldsymbol{\theta}^l; \boldsymbol{\theta}^l) = h(\boldsymbol{\theta}^l)$, which implies $\tilde{g}(\boldsymbol{\theta}^{l+1}; \boldsymbol{\theta}^l) \leq h(\boldsymbol{\theta}^l)$. Using the latter bound in (5.44), we arrive to the result of the lemma.

Proof of the theorem. In the rest of this proof, we utilize the following definition: $E(\boldsymbol{\theta}_S) := \ell(\boldsymbol{\theta}_S) + \lambda_1 \sum_{g \in S} \|\boldsymbol{\theta}_g\|_2$.

- **Part 1.** We will show that the event $\text{Supp}(\boldsymbol{\theta}^l) \neq \text{Supp}(\boldsymbol{\theta}^{l+1})$ cannot happen infinitely often. Suppose that $\text{Supp}(\boldsymbol{\theta}^l) \neq \text{Supp}(\boldsymbol{\theta}^{l+1})$ holds for some l. Then, either one of the following cases must hold for $g = 1 + (l \mod q)$: (I) $\boldsymbol{\theta}_g^l = 0 \neq \boldsymbol{\theta}_g^{l+1}$ or (II) $\boldsymbol{\theta}_g^l \neq 0 = \boldsymbol{\theta}_g^{l+1}$. Next, we will consider Case (I). Since $\boldsymbol{\theta}_g^{l+1} \neq 0$, then from the

definition of the thresholding operator in (5.14), we have $\|\boldsymbol{\theta}^{l+1}\|_2 > \sqrt{\frac{2\lambda_0}{\hat{L}_g}}$. Plugging the latter inequality into Lemma 5.1, we get:

$$h(\boldsymbol{\theta}^l) - h(\boldsymbol{\theta}^{l+1}) \geq \frac{\hat{L}_g - L_g}{\hat{L}_g}\lambda_0. \tag{5.45}$$

The same result in (5.45) applies for Case (II) as well. Thus, whenever the support changes, the objective improves by a positive constant (defined in the r.h.s of (5.45)), which combined with the fact that $h(\boldsymbol{\theta}) \geq 0$, implies that the support cannot change infinitely often.

- **Part 2.** First, we will show that the function $E(\boldsymbol{\theta}_S)$ is strongly convex. This trivially holds under Assumption 1(a). Next, we will assume that only Assumption 1(b) is satisfied. In this case, we have $h(\boldsymbol{\theta}^0) \leq h(\hat{\boldsymbol{\theta}})$ (where $\hat{\boldsymbol{\theta}}$ is defined in Assumption 1(b)). Since Algorithm 5.1 is a descent algorithm, we have $h(\boldsymbol{\theta}^l) \leq h(\hat{\boldsymbol{\theta}})$ for all $l \geq 0$. Thus, $E(\boldsymbol{\theta}^l) + \lambda_0 G(\boldsymbol{\theta}^l) \leq E(\hat{\boldsymbol{\theta}}) + \lambda_0 G(\hat{\boldsymbol{\theta}})$, which combined with the fact that $E(\boldsymbol{\theta}^l) \geq E(\hat{\boldsymbol{\theta}})$, implies that $G(\boldsymbol{\theta}^l) \leq G(\hat{\boldsymbol{\theta}})$ for all l. Thus, by the definition of k in the assumption, we have $\|\boldsymbol{\theta}^l\|_0 \leq k$ for all l. But since every k columns in $\mathbf{W}$ are linearly independent, we conclude that $E(\boldsymbol{\theta}_S)$ is strongly convex.

  After the support stabilizes (by Part 1), Algorithm 5.1 becomes equivalent to minimizing the strongly convex function $E(\boldsymbol{\theta}_S)$ using cyclic CD. By standard results on CD (e.g., see [22]), this is guaranteed to converge to a stationary solution $\boldsymbol{\theta}^*$ of $E(\boldsymbol{\theta}_S)$. This establishes (5.15).

  Finally, we will show that (5.16) and (5.17) hold. By the definition of the thresholding operator in (5.14), we have

$$\|\boldsymbol{\theta}_g^l\|_2 > \sqrt{\frac{2\lambda_0}{\hat{L}_g}}, \quad \forall g \in S. \tag{5.46}$$

Taking the limit as $l \to \infty$, we arrive to (5.16). Similarly, we have

$$\|\nabla_{\boldsymbol{\theta}_g} \ell(\boldsymbol{\theta}^l)\|_2 \leq \sqrt{2\lambda_0 \hat{L}_g} + \lambda_1, \quad \forall g \in S^c. \tag{5.47}$$

Taking the limit $l \to \infty$ leads to (5.17).

- **Part 3.** After support stabilization, Algorithm 5.1 is equivalent to performing cyclic CD to minimize the function $E(\boldsymbol{\theta}_S)$. Moreover, every iterate of the algorithm after support stabilization, i.e., $\boldsymbol{\theta}_S^l$ for $l \geq K$, belongs to the set $D := \{\boldsymbol{\theta}_S \mid \|\boldsymbol{\theta}_S\|_2 \geq \sqrt{\frac{2\lambda_0}{\hat{L}_g}}\}$ (this follows from (5.14)). Note that $\nabla_{\boldsymbol{\theta}_S} E(\boldsymbol{\theta}_S)$ is group-wise Lipschitz continuous over D, i.e., the following holds for every $g \in [q]$:

$$\|\nabla_{\boldsymbol{\theta}_S} E(\boldsymbol{\theta}_S^1) - \nabla_{\boldsymbol{\theta}_S} E(\boldsymbol{\theta}_S^2)\|_2 \leq \tilde{L}_g \|\boldsymbol{\theta}_S^1 - \boldsymbol{\theta}_S^2\|_2, \quad \forall \boldsymbol{\theta}_S^1, \boldsymbol{\theta}_S^2 \in D \text{ s.t. } \boldsymbol{\theta}_i^1 = \boldsymbol{\theta}_i^2 \; \forall i \neq g$$

where $\tilde{L}_g = \hat{L}_g + 2\lambda_1$. Similarly, $\nabla_{\boldsymbol{\theta}_S} E(\boldsymbol{\theta}_S)$ has a (global) Lipschitz constant of $L_S + 2|S|\lambda_1$, over D.

Lemma 3.3 of [18] bounds the objective values of cyclic CD after one full cycle. Their result holds for continuously differentiable functions whose gradient is Lipschitz over $\mathbb{R}^n$. Our function's gradient is Lipschitz over D, but we note that [18]'s result can be easily extended to D, leading to the following bound:

$$E(\boldsymbol{\theta}_S^{lq}) - E(\boldsymbol{\theta}_S^{(l+1)q}) \geq \frac{1}{2\eta} \|\nabla_{\boldsymbol{\theta}_S} E(\boldsymbol{\theta}_S^{lq})\|_2^2, \quad \forall\, l \geq K, \tag{5.48}$$

where η is defined in the statement of the theorem. In part 2, we have shown that $E(\boldsymbol{\theta}_S)$ is strongly convex. Thus, the following holds:

$$E(\boldsymbol{\alpha}_S) \geq E(\boldsymbol{\theta}_S) + \langle \nabla E(\boldsymbol{\theta}_S), \boldsymbol{\alpha}_S - \boldsymbol{\theta}_S \rangle + \frac{\sigma_S}{2} \|\boldsymbol{\alpha}_S - \boldsymbol{\theta}_S\|_2^2, \quad \forall \boldsymbol{\alpha}_S, \boldsymbol{\theta}_S. \tag{5.49}$$

Minimizing both sides in (5.49) w.r.t. $\boldsymbol{\alpha}_S$ and rearranging terms, we get

$$E(\boldsymbol{\theta}_S) - E(\boldsymbol{\theta}_S^*) \leq \frac{1}{2\sigma_S} \|\nabla_{\boldsymbol{\theta}_S} E(\boldsymbol{\theta}_S)\|_2^2, \quad \forall \boldsymbol{\theta}_S. \tag{5.50}$$

Inequalities (5.48) and (5.50) lead to:

$$(E(\boldsymbol{\theta}_S^{lq}) - E(\boldsymbol{\theta}_S^*)) - (E(\boldsymbol{\theta}_S^{(l+1)q}) - E(\boldsymbol{\theta}_S^*)) \geq \frac{1}{2\eta}\|\nabla_{\boldsymbol{\theta}_S} E(\boldsymbol{\theta}_S^{lq})\|_2^2 \tag{5.51}$$

$$\geq \frac{\sigma_S}{\eta}(E(\boldsymbol{\theta}_S^{lq}) - E(\boldsymbol{\theta}_S^*)). \tag{5.52}$$

Rearranging the terms in the above yields:

$$E(\boldsymbol{\theta}^{(l+1)q}) - E(\boldsymbol{\theta}^*) \leq \left(1 - \frac{\sigma_S}{\eta}\right)\Big(E(\boldsymbol{\theta}^{lq}) - E(\boldsymbol{\theta}^*)\Big). \tag{5.53}$$

Finally, we note that the function E in the above can be replaced by h (because of support stabilization), which establishes part 3.

Proof of Theorem 5.2

By Theorem 5.1, the support of the iterates in Algorithm 5.1 stabilizes, say on a support S, and converges to a solution of $\min_{\boldsymbol{\theta},\text{Supp}(\theta)=S} h(\boldsymbol{\theta})$. The latter observation along with the fact that Step 2 of Algorithm 5.2 ensures strict descent, imply that the sequence of solutions $\boldsymbol{\theta}^t$ in Algorithm 5.2 must have distinct supports. Therefore, the algorithm terminates in a finite number of iterations. Note that $\boldsymbol{\theta}^\dagger$ is the output of Algorithm 5.1 so it must satisfy the characterization given in part 2 of Theorem 5.1. Moreover, the search in Step 2 must fail at $\boldsymbol{\theta}^\dagger$, and thus (5.21) holds.

Proof of Proposition 5.2

Let $F_1(\boldsymbol{\theta}, \boldsymbol{z})$ and $F_2(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{s})$ be the objective functions in (5.25) and (5.26), respectively. Note that by definition, $v_2 = F_2(\boldsymbol{\theta}^*, \boldsymbol{z}^*, \boldsymbol{s}^*)$. Since $(\boldsymbol{\theta}^*, \boldsymbol{z}^*)$ is feasible for the problem corresponding to v_1, we have:

$$v_2 - v_1 \geq F_2(\boldsymbol{\theta}^*, \boldsymbol{z}^*, \boldsymbol{s}^*) - F_1(\boldsymbol{\theta}^*, \boldsymbol{z}^*) \tag{5.54}$$

Since $(\boldsymbol{\theta}^*, \boldsymbol{z}^*, \boldsymbol{s}^*)$ is optimal for the problem of v_2, it must satisfy $s_g^* = 0$ if $z_g^* = 0$ and $s_g^* = \frac{\|\boldsymbol{\theta}_g^*\|_2^2}{z_g^*}$ otherwise (because this is the smallest value of s_g, which satisfies (5.26c)). Plug-

ging s_g^* into the term $F_2(\boldsymbol{\theta}^*, \boldsymbol{z}^*, \boldsymbol{s}^*)$ in (5.54) and simplifying, leads to the result of the proposition.

Proof of Proposition 5.3

The root relaxation of (5.26) can be written as:

$$\min_{\boldsymbol{\theta}} \quad \left\{ \tilde{\ell}(\boldsymbol{\theta}) + \lambda_1 \sum_{g=1}^{q} \|\boldsymbol{\theta}_g\|_2 + \sum_{g=1}^{q} \min_{z_g, s_g} (\lambda_0 z_g + \lambda_2 s_g) \right\} \tag{5.55}$$

$$\text{s.t.} \quad \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}} z_g, \ g \in [q] \tag{5.56}$$

$$s_g z_g \geq \|\boldsymbol{\theta}_g\|_2^2, \quad g \in [q] \tag{5.57}$$

$$z_g \in [0,1], s_g \geq 0, \quad g \in [q] \tag{5.58}$$

Define

$$\omega(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) = \min_{z_g, s_g} (\lambda_0 z_g + \lambda_2 s_g) \quad \text{s.t.} \quad (5.56), (5.57), (5.58). \tag{5.59}$$

Note that the above optimization problem appears inside the second summation of (5.55). Next, we will derive a closed form expression for (5.59). Let $(\boldsymbol{\theta}_g, z_g, s_g)$ be some feasible solution. Then, the solution $(\boldsymbol{\theta}_g, \hat{z}_g, s_g)$, where $\hat{z}_g = \max\{\frac{\|\boldsymbol{\theta}_g\|_2^2}{s_g}, \frac{\|\boldsymbol{\theta}_g\|_2}{\mathcal{M}_{\text{U}}}\}$, has an objective value which is less than or equal to that of $(\boldsymbol{\theta}_g, z_g, s_g)$ (since $\hat{z}_g$ is the smallest possible choice of z_g which satisfies all the constraints)—if $\boldsymbol{\theta}_g = \mathbf{0}$ and $s_g = 0$, we assume that $\frac{\|\boldsymbol{\theta}_g\|_2^2}{s_g} = 0$, which leads to $\hat{z}_g = 0$. Thus, replacing constraints (5.56) and (5.57) with the constraint $z = \max\{\frac{\|\boldsymbol{\theta}_g\|_2^2}{s_g}, \frac{\|\boldsymbol{\theta}_g\|_2}{\mathcal{M}_{\text{U}}}\}$ does not change the optimal objective of the problem. This replacement leads to the following equivalent problem:

$$\omega(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) = \min_{z_g, s_g} (\lambda_0 z_g + \lambda_2 s_g) \quad \text{s.t.} \quad z_g = \max\left\{ \frac{\|\boldsymbol{\theta}_g\|_2^2}{s_g}, \frac{\|\boldsymbol{\theta}_g\|_2}{\mathcal{M}_{\text{U}}} \right\}, z_g \in [0,1], s_g \geq 0. \tag{5.60}$$

In the above, we can eliminate z_g by plugging its expression into the the objective and the constraint $z_g \in [0,1]$, which leads to the following equivalent formulation:

$$\omega(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) = \min_{s_g} \max \left\{ \underbrace{\frac{\lambda_0 \|\boldsymbol{\theta}_g\|_2^2}{s_g} + \lambda_2 s_g}_{\text{Term 1}}, \underbrace{\frac{\lambda_0 \|\boldsymbol{\theta}_g\|_2}{\mathcal{M}_{\text{U}}} + \lambda_2 s_g}_{\text{Term 2}} \right\} \;\; s.t. \;\; s_g \geq \|\boldsymbol{\theta}_g\|_2^2, \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}. \tag{5.61}$$

Suppose that Term 1 in (5.61) attains the maximum. This holds iff Term 1 $\geq$ Term 2, which simplifies to: $s_g \leq \mathcal{M}_{\text{U}} \|\boldsymbol{\theta}_g\|_2$. Term 1 is convex in s_g, so the solution of (5.61) (obtained via solving the first order optimality condition, assuming $s_g \leq \mathcal{M}_{\text{U}} \|\boldsymbol{\theta}_g\|_2$) is given $s_g^* = \sqrt{\lambda_0/\lambda_2} \|\boldsymbol{\theta}_g\|_2$ if $\|\boldsymbol{\theta}_g\|_2 \leq \sqrt{\lambda_0/\lambda_2} \leq M$, and $s_g^* = \|\boldsymbol{\theta}_g\|_2^2$ if $\sqrt{\lambda_0/\lambda_2} \leq \|\boldsymbol{\theta}_g\|_2 \leq M$. Plugging s_g^* into (5.61), leads to $\omega(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) = 2\lambda_0 \mathcal{H}(\sqrt{\lambda_2/\lambda_0} \|\boldsymbol{\theta}_g\|_2)$, for $\sqrt{\lambda_0/\lambda_2} \leq \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}$.

Now suppose Term 2 attains the maximum in (5.61). There are two lower bounds on s_g in this case: $s_g \geq \mathcal{M}_{\text{U}} \|\boldsymbol{\theta}_g\|_2$ (from Term 1 $\leq$ Term 2) and $s_g \geq \|\boldsymbol{\theta}_g\|_2^2$ (from the feasible set in (5.61)). Since $\|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}$, we have $\mathcal{M}_{\text{U}} \|\boldsymbol{\theta}_g\|_2 \geq \|\boldsymbol{\theta}_g\|_2^2$, which implies that $s_g \geq \mathcal{M}_{\text{U}} \|\boldsymbol{\theta}_g\|_2$ is the only lower bound needed. Thus, we can simplify (5.61) to:

$$\omega(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) = \min_{s_g} \frac{\lambda_0 \|\boldsymbol{\theta}_g\|_2}{\mathcal{M}_{\text{U}}} + \lambda_2 s_g \;\; s.t. \;\; s_g \geq \mathcal{M}_{\text{U}} \|\boldsymbol{\theta}_g\|_2, \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\text{U}}.$$

The optimal solution of the above is given by $s_g^* = \mathcal{M}_{\text{U}} \|\boldsymbol{\theta}_g\|_2$, and this holds for $\sqrt{\lambda_0/\lambda_2} \geq \mathcal{M}_{\text{U}}$. Plugging s_g^* into (5.61) leads to $\omega(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}}) = (\lambda_0/\mathcal{M}_{\text{U}} + \lambda_2 \mathcal{M}_{\text{U}}) \|\boldsymbol{\theta}_g\|_2$, for $\sqrt{\lambda_0/\lambda_2} \geq \mathcal{M}_{\text{U}}$. Finally, we replace the inner minimization in (5.55) by the closed form expression of $\omega(\boldsymbol{\theta}_g; \boldsymbol{\lambda}, \mathcal{M}_{\text{U}})$, which leads to the result of the proposition.

Proof of Proposition 5.4

Because $\kappa_{k,c} \geq \kappa_{k,1}$ for $c \geq 1$, it is sufficient to derive the stated inequality for $c = 1$.

Consider an arbitrary $\boldsymbol{\beta}$ satisfying $\boldsymbol{\beta} \neq \mathbf{0}$ and $G(\boldsymbol{\beta}) \leq 2k$. We let $J_0 \subseteq [q]$ index the k largest values in the set $\{\|\boldsymbol{\beta}_g\|_{2,1}\}_{g \in [q]}$, noting that $|J_0| = k$ and $\|\boldsymbol{\beta}_{J_0^c}\|_{2,1} \leq \|\boldsymbol{\beta}_{J_0}\|_{2,1}$. The stated

inequality follows from an observation that

$$\frac{\sqrt{2k}\|\mathbf{X}\boldsymbol{\beta}\|_2}{\sqrt{n}\|\boldsymbol{\beta}\|_{2,1}} \geq \frac{\sqrt{2k}\|\mathbf{X}\boldsymbol{\beta}\|_2}{2\sqrt{n}\|\boldsymbol{\beta}_{J_0}\|_{2,1}} \geq \frac{\kappa_{k,1}}{\sqrt{2}}.$$

□

Proof of Theorem 5.3

Optimality of $\widehat{\boldsymbol{\beta}}$ and feasibility of $\boldsymbol{\beta}^*$ imply $\|\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}\|_2^2 \leq \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2$, which leads to

$$\|\mathbf{X}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)\|_2^2 \leq 2\boldsymbol{\epsilon}^\top \mathbf{X}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*). \tag{5.62}$$

We will derive a bound for the right hand side of inequality (5.62).

First, consider a fixed subset $J \subseteq [q]$ such that $|J| = 2k$. We define $I_J = \cup_{g \in J} \mathcal{G}_g$ and $s = Tk$, noting that $|I_J| = 2s$. We choose an orthonormal basis $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_{2s}]$, such that the corresponding linear space contains the one spanned by features $\{\mathbf{x}_j\}_{j \in I_J}$. Then, $\|\boldsymbol{\Phi}^\top \boldsymbol{\epsilon}\|_2^2/\sigma^2$ has chi-square distribution with $2s$ degrees of freedom, and

$$\boldsymbol{\epsilon}^\top \mathbf{X}\boldsymbol{\theta} \leq \|\boldsymbol{\Phi}^\top \boldsymbol{\epsilon}\|_2 \|\mathbf{X}\boldsymbol{\theta}\|_2,$$

for all $\boldsymbol{\theta} \in \mathbb{R}^p$ with $\text{supp}(\boldsymbol{\theta}) \subseteq I_J$. Applying a chi-square tail bound (for example, the one in Section 8.3.2 of [44]), we derive that $|\boldsymbol{\Phi}^\top \boldsymbol{\epsilon}|^2 \lesssim \sigma^2 s(1+a)$ with probability at least $1 - \exp(-2sa)$. Consequently, with probability at least $1 - \exp(-2sa)$, inequality

$$\boldsymbol{\epsilon}^T \mathbf{X}\boldsymbol{\theta} \lesssim \left[\sigma^2 s(1+a)\right]^{1/2} \|\mathbf{X}\boldsymbol{\theta}\|_2 \tag{5.63}$$

holds uniformly for all $\boldsymbol{\theta} \in \mathbb{R}^p$ with $\text{supp}(\boldsymbol{\theta}) \subseteq I_J$.

We now extend this bound to *all* subsets $J \subseteq [q]$ that have size $2k$. Note that the number of such subsets is bounded by $(qe/2k)^{2k}$. Applying the union bound, we deduce that inequality (5.63) holds uniformly over both such J and $\boldsymbol{\theta}$ with probability at least $1 - \exp(-2sa + 2k\log(qe/2k))$. We note that $G(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*) \leq 2k$ and take $a = T^{-1}\log(eq/2k) + [2s]^{-1}\log(1/\delta_0)$.

It follows that

$$\boldsymbol{\epsilon}^T\mathbf{X}(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}^*) \lesssim \Big[\sigma^2 k[T+\log(eq/k)]+\sigma^2\log(1/\delta_0)\Big]^{1/2}\|\mathbf{X}(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}^*)\|_2,$$

with probability at least $1-\delta_0$. We complete the proof by combining the above bound with inequality (5.62). □

Proof of Corollary 5.2

We let c_0 be the universal constant from the error bound in Theorem 5.3 and define

$$W = \|\mathbf{X}\widehat{\boldsymbol{\beta}}-\mathbf{X}\boldsymbol{\beta}^*\|_2^2 - c_0\sigma^2 k\big[T+\log(q/k)\big].$$

By Theorem 5.3 we have $W \le c_0\sigma^2\log(1/\delta_0)$ with probability at least $1-\delta_0$. Hence,

$$\mathbb{P}\big(W > w\big) \le e^{-w/[c_0\sigma^2]},$$

for every non-negative w. Consequently,

$$\mathbb{E}W \le \int_0^\infty \mathbb{P}\big(W>w\big)dw \le \int_0^\infty e^{-w/[c_0\sigma^2]}dw \le c_0\sigma^2.$$

Thus, by the definition of W, we have

$$\mathbb{E}\|\mathbf{X}\widehat{\boldsymbol{\beta}}-\mathbf{X}\boldsymbol{\beta}^*\|_2^2 \le c_0\sigma^2 k\big[T+\log(q/k)\big]+c_0\sigma^2.$$

□

Proof of Proposition 5.5

Consider an arbitrary $f \in A_{2k,\xi}$. Let J_0 be the index set corresponding to the k components f_j with the largest $\|\cdot\|_n$ norm. Write r_n for $n^{-m/(2m+1)}$. Note that

$$r_n\mathrm{Pen}_g(f) \le (\xi/2-1/2)\sum\nolimits_{j=1}^q \|\mathbf{f}_j\|_n \le (\xi-1)\sum_{j\in J_0}\|\mathbf{f}_j\|_n,$$

and hence

$$\sum_{j \notin J_0} \|\mathbf{f}_j\|_n + r_n \mathrm{Pen}_{\mathrm{gr}}(f) \leq \xi \sum_{j \in J_0} \|\mathbf{f}_j\|_n.$$

Consequently, $f \in B(J_0, \xi)$. To complete the proof, we note that

$$\frac{\sqrt{2k}\|\mathbf{f}\|_n}{\sum_{j=1}^q \|\mathbf{f}_j\|_n} \geq \frac{\sqrt{2k}\|\mathbf{f}\|_n}{2\sum_{j \in J_0} \|\mathbf{f}_j\|_n} \geq \phi(k, \xi)/\sqrt{2}.$$

□

Proof of Theorem 5.4

By analogy with the $\|\cdot\|_n$ notation, we define $(\boldsymbol{\epsilon}, \mathbf{v})_n = (1/n)\sum_{i=1}^n \epsilon_i v_i$, for each $\mathbf{v} \in \mathbb{R}^n$. The global optimality of $\widehat{f}$, together with the feasibility of f^*, implies the following inequality:

$$\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 + \lambda_n \mathrm{Pen}_{\mathrm{gr}}(\widehat{f}) \leq 2(\boldsymbol{\epsilon}, \widehat{\mathbf{f}} - \mathbf{f}^*)_n + \lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*). \tag{5.64}$$

To control the term $(\boldsymbol{\epsilon}, \widehat{\mathbf{f}} - \mathbf{f}^*)_n$ we need the following result, which is proved in Section 5.A.

Lemma 5.2. *Let $\mathcal{F}_s = \{f : \ f \in \mathcal{C}_{\mathrm{gr}}, \ G(f) \leq s\}$. Then, with probability at least $1 - \epsilon$, inequality*

$$\begin{aligned}(\boldsymbol{\epsilon}/\sigma, \mathbf{f})_n \ \lesssim \ & \Big[s^{1/2+\gamma/(2m)} r_n + \sqrt{\frac{s\log(eq/s)}{n}} + \sqrt{\frac{\log(1/\epsilon)}{n}}\Big]\|\mathbf{f}\|_n \\ & + \Big[s^{1/2-\gamma(2m-1)/(2m)} r_n^2 + s^{-\gamma} r_n \sqrt{\frac{s\log(eq/s)}{n}} + s^{-\gamma} r_n \sqrt{\frac{\log(1/\epsilon)}{n}}\Big]\mathrm{Pen}_{\mathrm{gr}}(f)\end{aligned}$$

holds uniformly over $f \in \mathcal{F}_s$.

We now prove inequalities (5.36) and (5.37) in the statement of Theorem 5.4.

Proof of inequality (5.36). Note that $G(\widehat{f} - f^*) \leq 2k$. Applying Lemma (5.2) with

$f = \widehat{f} - f^*$, $s = 2k$ and $\epsilon = (k/q)^k$, we conclude that, with probability at least $1-(k/q)^k$,

$$\begin{aligned}(\boldsymbol{\epsilon}/\sigma, \widehat{\mathbf{f}} - \mathbf{f}^*)_n &\leq \tilde{c}_1 k^{1/2}\Big[k^{\gamma/(2m)} r_n + \sqrt{\frac{\log(eq/k)}{n}}\Big]\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n \\ &\quad + (c_1/4)\Big[k^{1/2-\gamma(2m-1)/(2m)} r_n^2 + k^{1/2-\gamma} r_n \sqrt{\frac{\log(eq/k)}{n}}\Big]\mathrm{Pen}_{\mathrm{gr}}(\widehat{f} - f^*) \quad (5.65)\end{aligned}$$

for some universal constants $\tilde{c}_1$ and c_1.

For the remainder of the proof we restrict our attention to the random event on which (5.65) holds. We will establish a general prediction error bound, from which inequality (5.36) will follow by setting $\gamma = m/(2m+1)$. We let

$$\begin{aligned}\tau_n &:= 2\tilde{c}_1 \sigma k^{1/2}\Big[k^{\gamma/(2m)} r_n + \sqrt{\frac{\log(eq/k)}{n}}\Big] \quad \text{and} \\ \lambda_n &\geq c_1 \sigma\Big[k^{1/2-\gamma(2m-1)/(2m)} r_n^2 + k^{1/2-\gamma} r_n \sqrt{\frac{\log(eq/k)}{n}}\Big],\end{aligned}$$

noting that when $\gamma = m/(2m+1)$, the last inequality matches the corresponding lower-bound on λ_n in the statement of Theorem 5.4. Multiplying inequality (5.64) by two and then applying (5.65) with $f = \widehat{f} - f^*$, we derive

$$\begin{aligned}2\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 + \lambda_n \mathrm{Pen}_{\mathrm{gr}}(\widehat{f}) &\leq 2\tau_n \|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n + 3\lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*) \\ &\leq \|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 + \tau_n^2 + 3\lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*).\end{aligned}$$

Consequently,

$$\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 \lesssim \sigma^2 k\Big[k^{\gamma/m} r_n + \frac{\log(eq/k)}{n}\Big] + \lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*).$$

Inequality (5.36) then follows from the above bound by letting $\gamma = m/(2m+1)$. We note that this choice of γ optimizes the prediction error rate in the setting where $\mathrm{Pen}_{\mathrm{gr}}(f^*) \asymp \sigma k$, however, the rate can be improved when $\mathrm{Pen}_{\mathrm{gr}}(f^*)$ and σk have different orders of magnitude.

Proof of inequality (5.37). Applying Lemma (5.2) with $s = 1$ and $\epsilon = 1/q$, we deduce

that with probability at least $1 - 1/q$, inequality

$$(\boldsymbol{\epsilon}/\sigma, \mathbf{f}_j)_n \lesssim \Big[r_n + \sqrt{\frac{\log(q)}{n}}\Big]\Big[\|\mathbf{f}_j\|_n + r_n \mathrm{Pen}(f_j)\Big]$$

holds uniformly over $f \in \mathcal{C}_{\mathrm{gr}}$ and $j \in [q]$. The above bound implies that there exists a universal constant c_0, such that

$$(\boldsymbol{\epsilon}/\sigma, \mathbf{f})_n = \sum_{j=1}^{q} (\boldsymbol{\epsilon}/\sigma, \mathbf{f}_j)_n \leq c_0 \Big[r_n + \sqrt{\frac{\log(q)}{n}}\Big]\Big[\sum_{j=1}^{q} \|\mathbf{f}_j\|_n + r_n \mathrm{Pen}_{\mathrm{gr}}(f)\Big].$$

Letting $f = \widehat{f} - f^*$, we conclude that

$$(\boldsymbol{\epsilon}/\sigma, \widehat{\mathbf{f}} - \mathbf{f}^*)_n \leq c_0 \Big[r_n + \sqrt{\frac{\log(q)}{n}}\Big]\Big[\sum_{j=1}^{q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n + r_n \mathrm{Pen}_{\mathrm{gr}}(\widehat{f} - f^*)\Big] \tag{5.66}$$

with probability at least $1 - 1/q$.

For the remainder of the proof we restrict our attention to the random event on which (5.66) holds. We define $\mu_n = 4c_0\sigma\big[r_n + \sqrt{\log(q)/n}\big]$ and let $\lambda_n \geq 4\mu_n r_n \xi/(\xi - 1)$. Applying inequality (5.66), we rewrite inequality (5.64) as follows:

$$2\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 + \lambda_n \mathrm{Pen}_{\mathrm{gr}}(\widehat{f} - f^*) \leq \mu_n \sum_{j=1}^{q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n + 3\lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*). \tag{5.67}$$

We now consider two possible cases.

Case i): $\mu_n \sum_{j=1}^{q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n \geq 3\lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*)$. It follows that

$$2\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 + \lambda_n \mathrm{Pen}_{\mathrm{gr}}(\widehat{f} - f^*) \leq 2\mu_n \sum_{j=1}^{q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n, \tag{5.68}$$

and, consequently, $2r_n \mathrm{Pen}_{\mathrm{gr}}(\widehat{f} - f^*) \leq 4(\mu_n r_n/\lambda_n) \sum_{j=1}^{q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n \leq (\xi - 1) \sum_{j=1}^{q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n$. Taking into account inequality $G(\widehat{f} - f^*) \leq 2k$ and Definition 5.2, we then derive

$$\sum_{j \leq q} \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n \leq [2k]^{1/2} [\psi(2k, \xi)]^{-1} \|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n. \tag{5.69}$$

Combining this bound with inequality (5.68), we colclude

$$\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 \leq \mu_n [2k]^{1/2} [\psi(2k, \xi)]^{-1} \|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n,$$

which implies the stated prediction error bound.

Case ii): $\mu_n \sum_{j=1}^q \|\widehat{\mathbf{f}}_j - \mathbf{f}_j^*\|_n < 3\lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*)$. Going back to inequality (5.67), we derive

$$2\|\widehat{\mathbf{f}} - \mathbf{f}^*\|_n^2 + \lambda_n \mathrm{Pen}_{\mathrm{gr}}(\widehat{f} - f^*) \leq 6\lambda_n \mathrm{Pen}_{\mathrm{gr}}(f^*),$$

which implies the stated prediction error bound. □

Proof of Lemma 5.2

Given $J \subseteq [q]$, we define a functional class $\mathcal{F}(J) = \{f : \ f(\mathbf{x}) = \sum_{j \in J} f_j(x_j), \ f_j \in \mathcal{C}\}$. We will need the following result, which is proved in Section 5.A.

Lemma 5.3. *Let $J \subseteq [q]$. Then, with probability at least $1 - e^{-t}$, inequality*

$$(\boldsymbol{\epsilon}/\sigma, \mathbf{f})_n \lesssim \left[|J|^{1/2+\gamma/(2m)} r_n + \sqrt{t/n} \right] \|\mathbf{f}\|_n + \left[|J|^{1/2-\gamma(2m-1)/(2m)} r_n^2 + |J|^{-\gamma} r_n \sqrt{t/n} \right] Pen_{\mathrm{gr}}(f)$$

holds uniformly over $f \in \mathcal{F}(J)$.

Let M_s denote the number of distinct subsets of $[q]$ that have size s. We note that $\log(M_s) \leq s \log(eq/s)$ and, thus, $M_s e^{-t} \leq e^{s \log(eq/s) - t}$. Applying Lemma 5.3 together with the union bound, we derive that, with probability at least $1 - e^{s \log(eq/s) - t}$, inequality

$$(\boldsymbol{\epsilon}/\sigma, \mathbf{f})_n \lesssim \left[s^{1/2+\gamma/(2m)} r_n + \sqrt{t/n} \right] \|\mathbf{f}\|_n + \left[s^{1/2-\gamma(2m-1)/(2m)} r_n^2 + s^{-\gamma} r_n \sqrt{t/n} \right] \mathrm{Pen}_{\mathrm{gr}}(f)$$

holds uniformly over $f \in \mathcal{F}_s$. We complete the proof by noting that for $t = s \log(eq/s) +$

$\log(1/\epsilon)$ the above inequality becomes

$$\begin{aligned}(\boldsymbol{\epsilon}/\sigma, \mathbf{f})_n \;\lesssim\; & \Big[s^{1/2+\gamma/(2m)} r_n + \sqrt{\frac{s\log(eq/s)}{n}} + \sqrt{\frac{\log(1/\epsilon)}{n}}\Big]\|\mathbf{f}\|_n \\ & + \Big[s^{1/2-\gamma(2m-1)/(2m)} r_n^2 + s^{-\gamma} r_n \sqrt{\frac{s\log(eq/s)}{n}} + s^{-\gamma} r_n \sqrt{\frac{\log(1/\epsilon)}{n}}\Big]\mathrm{Pen}_{\mathrm{gr}}(f),\end{aligned}$$

and the corresponding lower-bound on the probability simplifies to $1-\epsilon$. □

Proof of Lemma 5.3

Given a positive constant δ and a metric space $\mathcal{H}$ endowed with the norm $\|\cdot\|$, we use the standard notation and write $H(\delta, \mathcal{H}, \|\cdot\|)$ for the δ-entropy of $\mathcal{H}$ with respect to $\|\cdot\|$. More specifically, $H(\delta, \mathcal{H}, \|\cdot\|)$ is the natural logarithm of the smallest number of balls with radius δ needed to cover $\mathcal{H}$.

With a slight abuse of notation, we extend the domain of $\|\cdot\|_n$ from vectors in $\mathbb{R}^n$ to real-valued functions on $[0,1]^q$ by letting $\|\cdot\|_n$ be the empirical L_2-norm. Thus, given a function h, we let $\|h\|_n = [\sum_{i=1}^n h(\mathbf{x}_i)^2/n]^{1/2}$. This extension is consistent in the sense that $\|f\|_n = \|\mathbf{f}\|_n$ and $\|f_j\|_n = \|\mathbf{f}_j\|_n$ for $f \in \mathcal{C}_{\mathrm{gr}}$, $j \in [q]$.

We let $\mathcal{H}(J) = \{h :\ h \in \mathcal{F}(J),\ \|h\|_n/(r_n|J|^{-\gamma}) + \mathrm{Pen}_{\mathrm{gr}}(h) \le 1\}$, noting that $\|h\|_n \le r_n|J|^{-\gamma}$ and $\mathrm{Pen}_{\mathrm{gr}}(h) \le 1$ for every $h \in \mathcal{H}(J)$. By Corollary 8.3 in [167] (cf. Lemma 12 in the supplementary material for [158]),

$$\sup_{h\in\mathcal{H}(J)} (\boldsymbol{\epsilon}/\sigma, \mathbf{h})_n \lesssim n^{-1/2} \int_0^{r_n|J|^{-\gamma}} \sqrt{H(u, \mathcal{H}(J), \|\cdot\|_n)}\,du + r_n|J|^{-\gamma}\sqrt{t/n} \tag{5.70}$$

with probability at least $1-e^{-t}$. To bound the entropy, we will use the following result, proved in Section 5.A.

Lemma 5.4. *$H(u, \mathcal{H}(J), \|\cdot\|_n) \lesssim |J|(1/u)^{1/m}$ for $u \in (0,1)$.*

Noting that $r_n = n^{-m/(2m+1)}$ and, thus, $n^{-1/2} = r_n^{(2m+1)/(2m)}$, we derive

$$\begin{aligned} n^{-1/2} \int_0^{r_n|J|^{-\gamma}} \sqrt{H(u, \mathcal{H}(J), \|\cdot\|_n)} du &\lesssim n^{-1/2} \int_0^{r_n|J|^{-\gamma}} |J|^{1/2} u^{-1/(2m)} du \\ &\lesssim |J|^{1/2} n^{-1/2} \Big[r_n |J|^{-\gamma} \Big]^{(2m-1)/(2m)} \\ &= r_n^{(2m+1)/(2m)+(2m-1)/(2m)} |J|^{1/2-\gamma(2m-1)/(2m)} \\ &= r_n^2 |J|^{1/2-\gamma(2m-1)/(2m)}. \end{aligned}$$

Applying bound (5.70), we conclude that

$$\sup_{h \in \mathcal{H}(J)} (\boldsymbol{\epsilon}/\sigma, \mathbf{h})_n \lesssim r_n^2 |J|^{1/2-\gamma(2m-1)/(2m)} + r_n |J|^{-\gamma} \sqrt{t/n}$$

with probability at least $1 - e^{-t}$. The statement of the lemma is then a consequence of the fact that for every $f \in \mathcal{F}(J)$, function $f / \big[\|f\|_n / (r_n |J|^{-\gamma}) + \mathrm{Pen}_{\mathrm{gr}}(f) \big]$ falls in the class $\mathcal{H}(J)$.

□

Proof of Lemma 5.4

We will establish the stated entropy bound for a somewhat larger functional space $\mathcal{H}'_J = \{h : \ h \in \mathcal{F}(J), \ \|h\|_n + \mathrm{Pen}_{\mathrm{gr}}(h) \leq 1\}$. We treat m as fixed, so that universal constants in inequalities below are allowed to depend on m.

Consider an arbitrary $g \in \mathcal{C}$. By the Sobolev embedding theorem [for example, 135, Theorem 3.13], we can write g as a sum of a polynomial of degree $m-1$ and a function $\tilde{g}$ that satisfies $\|\tilde{g}\|_{L_2} \lesssim \mathrm{Pen}(g)$, where we note that $\mathrm{Pen}(g) = \mathrm{Pen}(\tilde{g})$. Applying Lemma 10.9 in [167], which builds on the interpolation inequality of [3], we derive $\|\tilde{g}\|_\infty \lesssim \mathrm{Pen}(\tilde{g})$. Thus,

$\mathcal{H}'_J \subseteq \{p + \tilde{h} :\ p \in \mathcal{P}_J,\ \tilde{h} \in \tilde{\mathcal{H}}_J\}$, where

$$\begin{aligned}
\mathcal{P}_J &= \{p :\ p(\mathbf{x}) = \alpha_0 + \sum_{j \in J} \sum_{l=1}^{m-1} \alpha_{jl} x_j^l,\ \alpha_0 \in \mathbb{R},\ \alpha_{jl} \in \mathbb{R}\ \forall j, k,\ \|p\|_n \leq 2\} \\
\tilde{\mathcal{H}}_J &= \{\tilde{h} :\ \tilde{h} \in \mathcal{F}(J),\ \mathrm{Pen}_{\mathrm{gr}}(\tilde{h}) \leq 1,\ \|\tilde{h}_j\|_\infty \lesssim \mathrm{Pen}(\tilde{h}_j)\ \forall j \in J\}.
\end{aligned}$$

We are able to impose the bound $\|p\|_n \leq 2$ in the definition of $\mathcal{P}_J$, because if $h = p + \tilde{h}$ for $h \in \mathcal{H}'_J$ and $\tilde{h} \in \tilde{\mathcal{H}}_J$, then $\|p + \tilde{h}\|_n \leq 1$ and $\|\tilde{h}\|_n \leq \mathrm{Pen}_{\mathrm{gr}}(\tilde{h}) \leq 1$. Consequently,

$$H(u, \mathcal{H}(J), \|\cdot\|_n) \leq H(u, \mathcal{H}'_J, \|\cdot\|_n) \leq H(u/2, \mathcal{P}_J, \|\cdot\|_n) + H(u/2, \tilde{\mathcal{H}}_J, \|\cdot\|_\infty), \qquad (5.71)$$

where we used the fact that the unit ball with respect to the $\|\cdot\|_\infty$-norm is contained within the corresponding ball with respect to the $\|\cdot\|_n$-norm. We note that $\mathcal{P}_J$ is a ball of radis 2, with respect to the $\|\cdot\|_n$-norm, in a linear functional space of dimension $|J|(m-1)+1$. Hence, $H(u/2, \mathcal{P}_J, \|\cdot\|_n) \lesssim |J| + |J| \log(1/u)$ by, for example, Corollary 2.6 in [167]. Thus, the result of Lemma 5.4 follows from 5.71 if we also establish that $H(\delta, \tilde{\mathcal{H}}_J, \|\cdot\|_\infty) \lesssim |J|(1/\delta)^{1/m}$ for $\delta \in (0, 1)$.

It is only left to derive the stated bound on $H(\delta, \tilde{\mathcal{H}}_J, \|\cdot\|_\infty)$. Note that we can represent functional class $\tilde{\mathcal{H}}_J$ as follows:

$$\tilde{\mathcal{H}}_J = \left\{ \tilde{h} :\ \tilde{h}(\mathbf{x}) = \sum_{j \in J} \lambda_j g_j(x_j),\ \sum_{j \in J} |\lambda_j| \leq 1\ g_j \in \mathcal{C},\ \mathrm{Pen}(g_j) \leq 1,\ \|g_j\|_\infty \leq 1\ \forall j \in J \right\}.$$

Given functions $\tilde{h}(\mathbf{x}) = \sum_{j \in J} \lambda_j g_j(x_j)$ and $\tilde{h}'(\mathbf{x}) = \sum_{j \in J} \lambda'_j g'_j(x_j)$ in $\tilde{\mathcal{H}}_J$, we have

$$\begin{aligned}
\|\tilde{h} - \tilde{h}'\|_\infty &\leq \|\sum_{j \in J} \lambda_j g_j - \sum_{j \in J} \lambda_j g'_j\|_\infty + \|\sum_{j \in J} \lambda_j g'_j - \sum_{j \in J} \lambda'_j g'_j\|_\infty \\
&\leq \max_{j \in J} \|g_j - g'_j\|_\infty \sum_{j \in J} |\lambda_j| + \max_{j \in J} \|g'_j\|_\infty \sum_{j \in J} |\lambda_j - \lambda'_j| \\
&\leq \max_{j \in J} \|g_j - g'_j\|_\infty + \sum_{j \in J} |\lambda_j - \lambda'_j|.
\end{aligned}$$

Consequently, if we let $\mathcal{G} = \{g :\ g \in \mathcal{C},\ \mathrm{Pen}(g) \leq 1,\ \|g\|_\infty \leq 1\}$, let $\|\cdot\|_1$ denote the ℓ_1-norm

and let B_1^d denote a unit ℓ_1-ball in $\mathbb{R}^d$, then

$$H(\delta, \tilde{\mathcal{H}}_J, \|\cdot\|_\infty) \leq |J| H(\delta/2, \mathcal{G}, \|\cdot\|_\infty) + H(\delta/2, B_1^{|J|}, \|\cdot\|_1).$$

By the results in [31], $H(\delta/2, \mathcal{G}_j, \|\cdot\|_\infty) \lesssim (1/\delta)^{1/m}$. By the standard bounds on the covering numbers of a norm ball, $H(\delta/2, B_1^{|J|}, \|\cdot\|_1) \lesssim |J| + |J| \log(1/\delta)$. Thus, $H(\delta, \tilde{\mathcal{H}}_J, \|\cdot\|_\infty) \lesssim |J|(1/\delta)^{1/m}$ for $\delta \in (0, 1)$. □

5.B Appendix: Additional Experimental Results and Details

Performance on the Birthweight Dataset

We study the Birthweight dataset, taken from the `R` package `grpreg`. Here, we predict birth weight using 7 grouped covariates. The dataset has 189 observations, which we randomly split into 75% for training and 25% for testing. On this dataset, we fit regularization paths for Group ℓ_0 , Lasso, and SCAD. For Group ℓ_0, we use an additional ℓ_2 regularization and consider $\lambda_2 \in \{1, 2, 4\}$. In Figure 5-4, we plot the test MSE versus the sparsity level for the different methods. The results show that the Group ℓ_0-based methods outperform Group Lasso and SCAD when the group size is 2 or more.

Additional Timing Comparisons

Here we consider the same setup as in the experiment of Section 5.6.2, and we report the running times for additional values of $\mathcal{M}_{\text{U}}$ to demonstrate the sensitivity of the runtime to $\mathcal{M}_{\text{U}}$. Let M^* be the value of $\mathcal{M}_{\text{U}}$ used in Section 5.6.2—note that this is the smallest value of $\mathcal{M}_{\text{U}}$. We express our choices of $\mathcal{M}_{\text{U}}$ in terms of M^*. We report the results for cases (i) and (ii) in Tables 5.3 and 5.4, respectively.

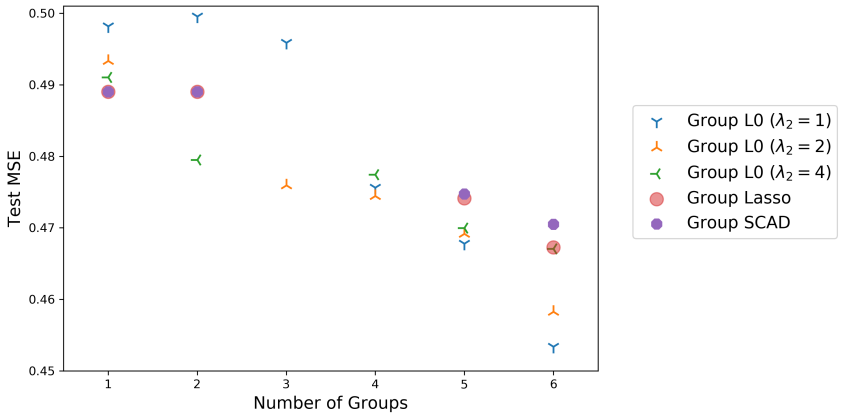


Figure 5-4: Test MSE on the Birthweight dataset. For Group ℓ_0, we consider additional ridge regularization and vary the corresponding regularization parameter $\lambda_2 \in \{1, 2, 4\}$. Group sizes 3 and 4 could not be attained using Group Lasso and SCAD.

Table 5.3: Running time in seconds for solving case (i), i.e., the MIP in (5.26) with $\lambda_2 = \lambda_2^*$, to optimality. A dash (-) indicates that Gurobi cannot solve the problem in 24 hours and has an optimality gap of 100% upon termination.

p	$\mathcal{M}_{\text{U}} = M^*$		$\mathcal{M}_{\text{U}} = 1.5M^*$		$\mathcal{M}_{\text{U}} = \infty$	
	Ours	Gurobi	Ours	Gurobi	Ours	Gurobi
10^3	96	24223	186	12320	192	2399
10^4	199	-	245	-	333	-
10^5	231	-	404	-	421	-
10^6	386	-	1014	-	1250	-
5×10^6	1922	-	3686	-	4036	-

Table 5.4: Running time in seconds for solving case (ii), i.e., the MIP in (5.26) with $\lambda_2 = 0$, to optimality. A star or dash (-) indicates that the solver cannot solve the problem in 24 hours. For star, the optimality gap (in percent) is shown in parenthesis, whereas the gap is 100% for dash.

p	$\mathcal{M}_{\text{U}} = M^*$		$\mathcal{M}_{\text{U}} = 1.5M^*$		$\mathcal{M}_{\text{U}} = 2M^*$	
	Ours	Gurobi	Ours	Gurobi	Ours	Gurobi
10^3	373	8737	913	10675	1010	13901
10^4	466	-	2813	-	*(3.9)	-
10^5	1136	-	*(4.7)	-	*(20.7)	-
10^6	1628	-	*(5.1)	-	*(21.6)	-

Description of the Amazon Reviews Dataset

This dataset is a subset of the Amazon Grocery and Gourmet Food dataset [93]. To obtain $\mathbf{X}$ and $\mathbf{y}$, we follow the same steps described in [86], and we restrict $\mathbf{X}$ to the top 5500 words in the corpus. Here $\mathbf{X}$ is a TF/IDF representation of the text reviews and $\mathbf{y}$ is a continuous

variable which measures review helpfulness. To obtain the groups, we run Latent Dirichlet Allocation (LDA) [33] on the corpus using `scikit-learn` [141], where we set the number of groups to 100. We then use the LDA solution to construct a collection of probability vectors $\{\boldsymbol{\pi}^{(i)}\}_{i=1}^{100}$, each corresponding to a topic. Here $\pi_j^{(i)}$ refers to the probability of encountering word j in topic i. We assign word j to the group with index $\arg\max_i \{\pi_j^{(i)}\}_{i=1}^{100}$ (i.e., to the group that allocates j the highest probability). For example, the top 5 words in group 1 are "coffee roast cup keurig cups" so the topic is on coffee. Group 2 has "bpa worse cans dented claim", which refers to problems with the packaging of the product. To obtain the training set, we sub-sample uniformly at random from the corpus and remove any covariates with zero variance (after sub-sampling), which reduces the number of covariates from 5500 to 3482. Note that the 100 groups have different sizes, ranging between 9 and 85.

Chapter 6

Learning Hierarchical Interactions at Scale: Convex and Discrete Optimization Algorithms

A part of this chapter appeared in AISTATS [87].

6.1 Introduction

In many machine learning applications, augmenting main effects with pairwise interactions can lead to better statistical models. Given a response vector $y \in \mathbb{R}^n$ and data matrix $X = [X_1, X_2, \ldots, X_p] \in \mathbb{R}^{n \times p}$, we consider a linear model of the form:

$$y = \beta_0 + \sum\nolimits_i X_i \beta_i + \sum\nolimits_{i<j} \theta_{ij}(X_i * X_j) + \epsilon, \tag{6.1}$$

where $*$ denotes element-wise multiplication and ϵ is a noise vector. Above, β_0 is the intercept, $\sum_i X_i \beta_i$ corresponds to the main effects and $\sum_{i<j} \theta_{ij}(X_i * X_j)$ denotes the interaction effects. The goal here is to learn the coefficients β, θ. For small n, this quickly leads to an ill-posed problem as the total number of coefficients is $1 + p + \binom{p}{2}$, which can far exceed n. Thus, imposing sparsity can be beneficial from both the statistical and computational

viewpoints. While vanilla sparsity-inducing regularization methods (e.g., using the ℓ_0 or ℓ_1 norm) can help, structured sparsity can be much more effective in this setting. Particularly, we consider enforcing sparsity under the *strong hierarchy* (SH) constraint [119, 30], which states that an interaction term should be non-zero only if its corresponding main effect terms are both non-zero. SH can be expressed via the following combinatorial statement:

$$\textbf{Strong Hierarchy (SH):} \;\; \theta_{ij} \neq 0 \Longrightarrow \beta_i \neq 0 \;\wedge\; \beta_j \neq 0$$

SH is a natural property that is widely used in high-dimensional statistics: it leads to more interpretable models with good predictive performance [56, 119, 30]. Moreover, SH promotes *practical sparsity*, i.e., it reduces the number of main features that need to be measured when making new predictions—this can significantly save on future data collection costs [30].

An impressive line of work for learning sparse interactions under SH is based on a regularization framework (see [54, 146, 30, 108, 181, 155] and the references therein). These methods minimize the empirical risk with sparsity-inducing regularizers to enforce SH. While these lead to estimators with good statistical properties, computation remains a major challenge. Indeed, the complex nature of the regularizers used to enforce SH prevents most current optimization algorithms from scaling beyond $p \sim 10^3$. However, in many real-world applications p can be in the order of tens of thousands, which is limiting the adoption of SH methods in practice. To address this shortcoming, we propose a convex regularization framework and develop a novel scalable algorithm, by carefully exploiting the problem-specific structural sparsity. Our algorithm can solve the SH learning problem with $p = 50,000$ ($\sim 10^9$ interactions) and achieves significant speed-ups (over 4900x) compared to state-of-the-art convex optimization algorithms. In addition, motivated by the statistical benefits of ℓ_0 regularization we saw in Chapters 2 and 3, we consider a discrete formulation of the SH learning problem. We develop a scalable nonlinear branch-and-bound (BnB) framework for the discrete formulation. Specifically, our BnB uses the proposed convex optimization algorithm to efficiently solve the node subproblems. Our experiments indicate that the proposed BnB can solve the discrete formulation for problems with up to 50 million interactions (assuming highly sparse solutions are desired)—over 1000x larger than what can be handled by state-of-the-art MIP

solvers such as Gurobi and MOSEK.

6.1.1 Problem Formulation

Various convex regularizers (a.k.a. penalties) for enforcing SH exist in the literature. In many cases, the choices seem somewhat ad hoc and involve auxiliary variables that can increase the memory and computational footprints. Here we transparently derive our regularizer via a convex relaxation of a mixed integer program (MIP) [27] that enforces SH. In what follows, we denote the interaction column $X_i * X_j$ by $\tilde{X}_{ij}$. Performing variable selection under SH for model (6.1) can be naturally expressed using ℓ_0 regularization:

$$\begin{aligned} \min_{\beta,\theta} \quad & f(\beta,\theta) + \alpha_1\|\beta\|_0 + \alpha_2\|\theta\|_0 \\ \text{s.t.} \quad & \theta_{ij} \neq 0 \implies \beta_i \neq 0 \text{ and } \beta_j \neq 0 \end{aligned} \tag{6.2}$$

where $f(\beta,\theta) = \frac{1}{2}\|y - X\beta - \sum_{i<j}\tilde{X}_{ij}\theta_{ij}\|_2^2$ and $\|u\|_0$ denotes the number of non-zeros in the vector u. The parameters α_1 and α_2 control the number of non-zeros. Note that we ignore the intercept term to simplify the presentation.

Problem (6.2) can be expressed using the following MIP[1], in which we introduce auxiliary binary variables to model the SH constraint and the ℓ_0 (pseudo) norms:

$$\begin{aligned} \min_{\beta,\theta,z} \quad & f(\beta,\theta) + \alpha_1 \sum\nolimits_i z_i + \alpha_2 \sum\nolimits_{i<j} z_{ij} \\ \text{s.t.} \quad & |\beta_i| \leq M z_i, \ \forall\ i \\ & |\theta_{ij}| \leq M z_{ij}, \quad z_{ij} \leq z_i, \quad z_{ij} \leq z_j, \ \forall\ i<j \\ & z_i \in \{0,1\} \ \forall\ i, \quad z_{ij} \in \{0,1\} \ \forall\ i<j \end{aligned} \tag{6.3}$$

where the optimization variables are β, θ, and z. In the above, M is a large constant chosen such that some optimal solution β^*, θ^* to (6.2) satisfies $\|\beta^*, \theta^*\|_\infty \leq M$. Here $z_i = 0$ implies $\beta_i = 0$ and similarly $z_{ij} = 0$ implies $\theta_{ij} = 0$. Problem (6.3) is known to be NP-Hard [128] and can be very difficult to scale using existing MIP solvers. Our goal in this chapter is two-fold.

[1]There can be pathological cases where the MIP does not satisfy SH. However, when y is drawn from a continuous distribution, SH is satisfied w.p. 1. We discuss this in more detail in Section 1 of the appendix.

First, we aim to develop a scalable algorithm for solving a convex relaxation of Problem (6.3). Second, we aim to develop an exact algorithm for solving MIP (6.3) to optimality—we achieve this by using our scalable convex optimization solver within a nonlinear BnB framework.

For easier presentation, we introduce some notation. For every $i \in \{1, 2, \ldots, p\}$, we define G_i as the set of indices of all interactions corresponding to β_i, i.e.,

$$G_i \stackrel{\text{def}}{=} \{(1,i), (2,i), \ldots, (i-1,i), (i,i+1), \ldots, (i,p)\}.$$

We use the notation θ_{G_i} to refer to the vector of θ_{ij}'s whose indices are in G_i.

In Lemma 6.1, we derive a convex relaxation of Problem (6.3). Specifically, we relax the binary variables in Problem (6.3) to $[0, 1]$, and then simplify the problem to remove the dependence on the z_i's and z_{ij}'s, i.e., we move from the extended (β, θ, z) space back to the original (β, θ) space. The resulting relaxation involves box constraints on the coefficients β, θ—we eliminate these constraints to simplify the problem.

Lemma 6.1. *(Convex Relaxation) The following is a convex relaxation of Problem* (6.3)*:*

$$\min_{\beta,\theta} \quad f(\beta, \theta) + \Omega(\beta, \theta), \tag{6.4}$$

where

$$\Omega(\beta, \theta) \stackrel{\text{def}}{=} \lambda_1 \sum_{i=1}^{p} \max\{|\beta_i|, \|\theta_{G_i}\|_\infty\} + \lambda_2 \|\theta\|_1,$$

and $\lambda_1 = \alpha_1/M, \lambda_2 = \alpha_2/M$*.*

Relaxation (6.4) will be the main focus of our convex optimization algorithm. We note that the relaxation in (6.4) belongs to the original (β, θ) space. From an algorithmic perspective, this space is easier to work with compared to an extended space (e.g., the one involving z). [155] propose a general family of problems for enforcing SH, which includes our relaxation above as a special case (however, they do not discuss the connections to the original Problem (6.2)). The solutions of (6.4) satisfy SH with probability one under model (6.1) when $\epsilon \sim$

$N(0, \sigma^2 I)$ (see [155]). For a general response y, there can be pathological cases where SH is not satisfied by (6.4), however, these cases rarely appear in practice and are usually not considered by current regularization-based approaches for SH [146, 30, 108, 155].

Contributions and Organization

The main contribution of our work is developing a scalable algorithm for solving (6.4). Our proposal is based on proximal gradient descent (PGD). However, PGD is limited by two major computational bottlenecks, due to the scale of the problem. First, the proximal problem does not admit a closed-form solution, so solving it requires running an iterative optimization algorithm over $\mathcal{O}(p^2)$ variables. Second, the repeated gradient computations can be very expensive as each computation requires $\mathcal{O}(np^2)$ operations.

To mitigate these bottlenecks, we **(i)** introduce new *proximal screening rules* that can efficiently identify many of the zero variables and groups in the proximal problem, **(ii)** demonstrate how our proposed screening rules can decompose the proximal problem so that it can be solved in parallel, and **(iii)** develop a specialized active-set algorithm along with a novel *gradient screening* method for avoiding costly gradient evaluations. We also demonstrate how our scalable convex optimization solver can be used within a nonlinear BnB framework to solve the MIP in (6.3) to optimality. We open source the implementation of the convex optimization algorithm through our toolkit hierScale[2]. We demonstrate how our convex algorithm scales to high-dimensional problems having dense matrices with $p = 50,000$ ($\sim 10^9$ interactions), achieving over 4900x speed-ups compared to the state of the art. We also show how our nonlinear BnB, based on the convex algorithm, can solve instances of MIP (6.3) with ~ 50 million interactions (when highly sparse solutions are desired), which is over 1000x larger than what can be handled using Gurobi and MOSEK.

Organization: In Section 6.2, we introduce proximal screening and decomposition rules that allow for solving the proximal problem efficiently in PGD. In Section 6.3, we develop an active set algorithm (based on PGD) for solving the convex relaxation in (6.4). Then, in Section 6.4, we present our custom BnB framework for solving MIP (6.3). Finally, in Section

[2] `https://pypi.org/project/hierScale`

6.5, we present the results of our computational experiments.

Related Work

Many methods for enforcing SH exist in the literature. The methods can be broadly categorized into multi-step methods (e.g., [179, 79]), Bayesian and approximate methods (e.g., [53, 161]), and optimization-based methods. We discuss relevant work in the latter category as they are directly related to our work. [54] re-parameterize the interactions problem so that $\theta_{ij} = \gamma_{ij}\beta_i\beta_j$ (where γ_{ij} is an optimization variable) and enforce sparsity by adding an ℓ_1 norm regularization on β and θ—this approach, however, leads to a challenging non-convex optimization problem. [146] enforce SH by using ℓ_2 regularization on the predictions made by every group. [30] propose the Hierarchical Lasso, which shares a similar objective with our problem, except that $\|\theta_{G_i}\|_1$ is used instead of $\|\theta_{G_i}\|_\infty$. They use ADMM [39] for computation. [108] propose an overlapped group Lasso formulation and solve it using a variant of the FISTA algorithm [17] along with strong screening rules [163]. Their toolkit glinternet seems to be the fastest toolkit for learning sparse interactions we are aware of. [155] consider a formulation similar to (6.4), but with the ℓ_∞ norm replaced with ℓ_2 norm, and develop prediction error bounds and a splitting-based algorithm—the largest problem they consider has $p = 1000$.

[114] consider learning problems regularized with sum of ℓ_∞ norms over groups (with potential overlaps), which includes our problem as a special case. They show that the proximal operator can be efficiently solved using network flow algorithms and propose algorithms based on PGD and ADMM. Our approaches differ: here we exploit the specific structure of our objective function (particularly the presence of both the ℓ_∞ and ℓ_1 norms) to derive the proximal screening rules and decompose the proximal problem—such screening/decomposition rules are not discussed in [114]. We also note that [99] develop a scalable PGD-based algorithm for problems regularized with sums of ℓ_∞ or ℓ_2 norms, under a tree structure constraint. However, our model does not satisfy this constraint and thus their algorithms are not applicable. Many other works also consider algorithms for structured sparsity and discuss interesting connections to submodularity (e.g., see [11, 12, 10] and the references therein).

Unfortunately, prior work cannot easily scale beyond p in the hundreds to few thousands. A main reason is that the standard algorithms (e.g., PGD and ADMM) are limited by costly gradient evaluations and by solving expensive sub-problems (e.g., solving the proximal problem in PGD requires an iterative optimization method). Our proposal addresses these key computational bottlenecks. For related work on solving MIPs using BnB, we refer the reader to Chapter 3.

Notation: We use the notation $[p]$ to refer to the set $\{1, 2, \ldots, p\}$. We denote the complement of a set A by A^c. For a set $A \subseteq [p]$, β_A refers to the sub-vector of β restricted to coordinates in A. We use $\nabla_{\beta_A, \theta_B} f(\beta, \theta)$ to refer to the components of $\nabla f(\beta, \theta)$ corresponding to the vectors β_A and θ_B. For any scalar a, we define $[a]_+ = \max\{a, 0\}$. A function $u \mapsto g(u)$ is said to be Lipschitz with parameter L if $\|g(u) - g(v)\|_2 \leq L\|u - v\|_2$ for all u, v. Proofs of all lemmas and theorems are in the appendix.

6.2 Proximal Screening and Decomposition

To solve the non-smooth convex problem (6.4), we use PGD [17, 132], which is an effective and popular choice for handling structured sparse learning problems (e.g., see [12, 114, 51] and references therein). However, there are two major bottlenecks: (i) solving the proximal problem which does not admit a closed-form solution and (ii) the repeated gradient computations each requiring $\mathcal{O}(np^2)$ operations. In this section, we address bottleneck (i) through new proximal screening and decomposition rules, and we handle (ii) in Section 6.3 using active-set updates and gradient screening.

We first present the basic PGD algorithm below.

Algorithm 6.1: Proximal Gradient Descent

- Input: β^0, θ^0 and L: the Lipschitz parameter of the gradient map $(\beta, \theta) \mapsto \nabla f(\beta, \theta)$.
- **For** $k \geq 0$ repeat the following till convergence:

$$\tilde{\beta} \leftarrow \beta^k - \nabla_\beta f(\beta^k, \theta^k)/L, \quad \tilde{\theta} \leftarrow \theta^k - \nabla_\theta f(\beta^k, \theta^k)/L$$
$$\begin{bmatrix} \beta^{k+1} \\ \theta^{k+1} \end{bmatrix} \leftarrow \underset{\beta,\theta}{\arg\min} \frac{L}{2} \left\| \begin{bmatrix} \beta - \tilde{\beta} \\ \theta - \tilde{\theta} \end{bmatrix} \right\|_2^2 + \Omega(\beta, \theta) \tag{6.5}$$

The sequence $\{\beta^k\}$ generated by Algorithm 6.1 is guaranteed to converge to an optimal solution [15]. The objective values converge at a rate of $\mathcal{O}(1/k)$, and this can be improved to $\mathcal{O}(1/k^2)$ by using accelerated PGD [132, 15]. However, there is no closed-form solution for the proximal problem in (6.5)—this is due to the overlapping variables in the ℓ_∞ norms. Thus, iterative optimization algorithms are needed to solve (6.5). For example, [114] presents a dual reformulation and an efficient network flow algorithm for solving a class of proximal problems which includes (6.5). In the appendix, we present an alternative dual which uses less variables (as we exploit the specific structure of our problem) and present a dual block coordinate ascent (BCA) algorithm for solving it. However, iterative algorithms (e.g., [114]'s or our proposed BCA) require significant time to solve (6.5) when p is large, which can lead to a serious bottleneck.

In what follows, we present algorithm-agnostic schemes to speed up solving the proximal problem. In Section 6.2.1, we introduce screening rules to eliminate variables from the proximal problem (prior to optimization). In Section 6.2.2, we demonstrate how these rules can decompose the proximal problem, which allows for solving it in parallel.

6.2.1 Proximal Screening

Typically, we expect the solutions of (6.5) to be sparse, as $\Omega(\beta, \theta)$ incorporates sparsity-inducing norms. To exploit this sparsity, we propose new *proximal screening* rules, which can efficiently identify many of the zero groups and variables in (6.5). We present the rules

in Theorem 6.1.

Theorem 6.1. *(Proximal Screening) Let β^*, θ^* be the optimal solution of Problem (6.5). Then, the following group-level rule holds for every $i \in [p]$:*

$$\sum_{t \in G_i} \Big[|\tilde{\theta}_t| - \frac{\lambda_2}{L} \Big]_+ \leq \frac{\lambda_1}{L} - |\tilde{\beta}_i| \implies \beta_i^*, \theta_{G_i}^* = 0, \tag{6.6}$$

and the following feature-level rule holds for $i < j$:

$$|\tilde{\theta}_{ij}| \leq \frac{\lambda_2}{L} \implies \theta_{ij}^* = 0. \tag{6.7}$$

The rules in Theorem 6.1 can be used to optimize (6.5) over a smaller set of variables (i.e., only over the variables that do not pass the screening checks). These rules are easy to check. Particularly, the rule in (6.6) requires $\mathcal{O}(p)$ operations to screen a group of p variables. The feature-level rule allows us to set θ_{ij}'s with $|\tilde{\theta}_{ij}| \leq \lambda_2/L$ to zero. The group-level rule is less restrictive: in group i, the θ_{ij}'s with $|\tilde{\theta}_{ij}| > \lambda_2/L$ can be still set to zero if $|\tilde{\beta}_i|$ is sufficiently small (i.e., if the contribution of main effect i is weak).

Related Work on Screening Rules: Our proposed rules are safe in the sense that only variables that are zero in the optimal solution of (6.5) can be discarded. However, our rules are different from the *safe screening rules* used in the literature (e.g., [71, 174, 38, 106, 173, 129, 126, 130]) that are designed to identify zero variables in the full problem. In particular, our rules can potentially eliminate more variables in the proximal problem compared to the safe rules, since a variable can be safe to eliminate from a proximal problem but not from the full problem. This allows for exploiting more parallelism when solving the proximal problem (see Theorem 6.2). Moreover, the state-of-the-art safe rules (e.g., the sequential rules of [174], and the dynamic and Gap rules of [38] and [130]), update (improve) the rules as the algorithm progresses, by leveraging previous solutions. However, every such rule update entails a full gradient computation, which can be very costly in our problem (see [130] for a survey and a discussion on this computational issue). On the other hand, our rules do not require gradient computations. In our experiments, we compare against glinternet [108] which uses strong

screening rules (an approximate and aggressive variant of safe rules proposed in [163]).

6.2.2 Proximal Decomposition

An important consequence of Theorem 6.1 is that it usually decomposes Problem (6.5) into many independent smaller optimization problems. This allows solving (6.5) in parallel. Before presenting the decomposition formally, we give a simple motivating example.

Example 6.1. *Suppose $p = 2$ with one interaction effect; and rule* (6.7) *has identified $\theta^*_{12} = 0$. We can now eliminate θ_{12} and solve the proximal problem* (6.5) *with $\Omega(\beta, \theta) = \lambda_1|\beta_1| + \lambda_1|\beta_2|$. This decomposes the problem into two independent optimization tasks involving β_1 and β_2—the solutions can be easily obtained via soft-thresholding.*

Next, we formalize the idea of decomposition. Let us define $\mathcal{V}$ as the set of indices of the groups that do not pass the screening test in (6.6), i.e.,

$$\mathcal{V} = \Big\{ i \in [p] \;\; \Big| \sum_{t \in G_i} \Big[|\tilde{\theta}_t| - \frac{\lambda_2}{L} \Big]_+ > \frac{\lambda_1}{L} - |\tilde{\beta}_i| \Big\}. \tag{6.8}$$

We also define $\mathcal{E}$ as the set of interaction indices that do not pass the test in (6.7) and whose corresponding main indices are in $\mathcal{V}$, i.e.,

$$\mathcal{E} = \Big\{ (i,j) \in \mathcal{V}^2 \;\; | \; |\tilde{\theta}_{ij}| > \lambda_2 / L \Big\}. \tag{6.9}$$

Definition 6.1. *(Connected Components) Define the simple undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V}$ is defined in* (6.8) *and the edge set $\mathcal{E}$ is defined in* (6.9)*. Let the κ connected components of $\mathcal{G}$ be denoted by $\{\mathcal{G}_l\}_1^\kappa$, where $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l),\ l \in [\kappa]$.*

Theorem 6.2 states that Problem (6.5) can be solved by decomposing it into smaller independent optimization problems, each corresponding to a connected component of the graph $\mathcal{G}$.

Theorem 6.2. *Let β^*, θ^* be the optimal solution of the proximal problem in* (6.5)*. Then,*

$\beta^*_{\mathcal{V}^c} = 0, \quad \theta^*_{\mathcal{E}^c} = 0$, *and for every* $l \in [\kappa]$*:*

$$\begin{bmatrix} \beta^*_{\mathcal{V}_l} \\ \theta^*_{\mathcal{E}_l} \end{bmatrix} = \underset{\beta_{\mathcal{V}_l}, \theta_{\mathcal{E}_l}}{\arg\min} \frac{L}{2} \left\| \begin{bmatrix} \beta_{\mathcal{V}_l} - \tilde{\beta}_{\mathcal{V}_l} \\ \theta_{\mathcal{E}_l} - \tilde{\theta}_{\mathcal{E}_l} \end{bmatrix} \right\|_2^2 + \Omega(\beta_{\mathcal{V}_l}, \theta_{\mathcal{E}_l}).$$

Theorem 6.2 allows for solving (6.5) in parallel. The extent of parallelism depends on the number of the connected components and their sizes. In practice, we solve the problem for a regularization path with warm starts[3], along with active-set updates (discussed in Section 6.3). These can significantly reduce the sizes of the connected components. Indeed, our experiments on real high-dimensional datasets (with $p \sim 5000$), indicate that the maximum number of vertices and edges in each connected component is in the order of hundreds to few thousands (for all solutions in the path)—see the appendix for details. The majority of the connected components are typically isolated (i.e., composed of a single vertex), so their corresponding optimization problems can be solved by a simple soft thresholding operation. However, we note that, in the absence of warm starts, the number of edges can grow into hundreds of thousands for the same datasets.

6.3 Active Sets and Gradient Screening

In Section 6.2, we addressed the bottleneck of solving the proximal problem. In this section, we focus on another major bottleneck: the repeated full gradient computations in PGD. Particularly, we develop an active-set algorithm that exploits the screening rules and decomposition we introduced in Section 6.2 to reduce the number of full gradient computations. Moreover, we introduce a new gradient screening method which aids in reducing the cost of every gradient computation.

[3]A regularization path is the sequence of solutions obtained by solving (6.4) for a sequence of λ_1's and λ_2's. The solution of the current λ_1, λ_2 is used as a warm start (initial solution) when solving for the next λ_1, λ_2 in the sequence.

6.3.1 Active Set Updates

Every evaluation of $\nabla f(\beta, \theta)$ in Algorithm 6.1 requires $\mathcal{O}(np^2)$ operations, which can take several minutes on a modern machine when $p \sim 50,000$. To minimize the number of these full gradient evaluations, we propose an active-set algorithm: we run Algorithm 6.1 over a small subset of variables, namely, the active set. After obtaining an optimal solution restricted to the active set, we augment the set with the variables that violate the optimality conditions (if any) and resolve the problem on the new set. Such an approach is effectively used for speeding up sparse learning algorithms in other contexts [121, 69, 125].

Our active set is defined by the sets $\mathcal{A}$ and $\mathcal{T}$ containing indices of the main effects and interaction effects (respectively) to be included in the model. Given $\mathcal{A}$ and $\mathcal{T}$ we consider

$$\begin{aligned} \hat{\beta}, \hat{\theta} \in \arg\min_{\beta,\theta} \quad & f(\beta, \theta) + \Omega(\beta, \theta) \\ \text{s.t.} \quad & \beta_{\mathcal{A}^c} = 0, \quad \theta_{\mathcal{T}^c} = 0, \end{aligned} \tag{6.10}$$

which is solved with Algorithm 6.1 *restricted* to the active set. To check whether $\hat{\beta}, \hat{\theta}$ is optimal for Problem (6.4), we run a single iteration of Algorithm 6.1 over all the variables (including those outside the active set) – we refer to this as the *master iteration*. If the master iteration does not change the support (i.e., the non-zeros), then the solution $\hat{\beta}, \hat{\theta}$ is optimal. Otherwise, we augment $\mathcal{A}$ and $\mathcal{T}$ with the variables that became non-zero (after the iteration) and solve (6.10) again. To speed up the costly master iteration, we perform screening and decompose the master iteration as described in Theorem 6.2 so that it can be solved in parallel. We present the algorithm more formally below:

Algorithm 6.2: Parallel Active-set Algorithm

- Input: Initial estimates of $\mathcal{A}$ and $\mathcal{T}$
- **Repeat** until convergence:
  1. Solve the problem restricted to the active set, i.e., (6.10) to get a solution $\hat{\beta}, \hat{\theta}$.
  2. Compute $\nabla_\beta f(\hat{\beta}, \hat{\theta})$ and $\nabla_\theta f(\hat{\beta}, \hat{\theta})$. Set $\tilde{\beta} \leftarrow \hat{\beta} - \frac{1}{L}\nabla_\beta f(\hat{\beta}, \hat{\theta})$, $\tilde{\theta} \leftarrow \hat{\theta} - \frac{1}{L}\nabla_\theta f(\hat{\beta}, \hat{\theta})$.
  3. Construct the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in Definition 1 and find its connected components.
  4. Solve (6.5) in parallel using Theorem 6.2. If the support stays the same, **terminate**, o.w., augment $\mathcal{A}$ and $\mathcal{T}$ with variables that just became non-zero.

Steps 2-4 in Algorithm 6.2 are the equivalent of performing one iteration of Algorithm 6.1 over all variables, while using screening and decomposition. We note that screening and decomposition are also very useful at the active-set level (i.e., for step 1) as we need to repeatedly solve the proximal problem till convergence. Typically, the active-set sizes are relatively small, which can help further mitigate the cost of solving the proximal problem.

6.3.2 Gradient Screening

Algorithm 6.2 effectively reduces the total number of gradient computations. However, even a single gradient computation can take minutes for $p \sim 50,000$. Here we propose a novel gradient screening method to reduce the cost of every gradient computation in Algorithm 6.2. Specifically, every time a gradient is needed, we identify parts of the gradient that are not essential to optimization—this allows for computing a smaller (and cheaper) gradient. This is a major improvement over current active-set approaches, which require a full gradient computation to check optimality.

We note that the full gradient in step 2 of Algorithm 6.2 is only used to construct the graph $\mathcal{G}$ in step 3. The next lemma, states that only a part of the gradient is needed to construct $\mathcal{G}$.

Lemma 6.2. *The graph $\mathcal{G}$ in step 2 of Algorithm 6.2 can be constructed from the following gradients: $\nabla_\beta f(\hat{\beta}, \hat{\theta})$, $\nabla_{\theta_\mathcal{T}} f(\hat{\beta}, \hat{\theta})$, and $\nabla_{\theta_\mathcal{S}} f(\hat{\beta}, \hat{\theta})$, where $\mathcal{S}$ is the critical set defined by*

$$\mathcal{S} = \{(i,j) \in \mathcal{T}^c \mid |\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})| > \lambda_2\}. \tag{6.11}$$

Note that $\nabla_\beta f(\hat{\beta}, \hat{\theta})$ is relatively easy to compute, and $\nabla_{\theta_\mathcal{T}} f(\hat{\beta}, \hat{\theta})$ is available as a byproduct of step 1. Thus, if the size of $\mathcal{S}$ in (6.11) is small, then Lemma 6.2 suggests a significant reduction in the computation time of step 2. Specifically, if $\mathcal{S}$ is given, computing the gradients in Lemma 2 has a cost $\mathcal{O}(n(p + |\mathcal{S}|))$, which can be much smaller than the cost of full gradient computation $\mathcal{O}(np^2)$. As discussed below, we can obtain $\mathcal{S}$ without explicitly computing $\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})$ for all $(i,j) \in \mathcal{T}^c$—an operation that costs $\mathcal{O}(np^2)$.

Our key idea is to obtain a set $\hat{\mathcal{S}}$ that is guaranteed to contain $\mathcal{S}$ (i.e., $\mathcal{S} \subset \hat{\mathcal{S}}$), by using currently available information on gradients. Note that $|\hat{\mathcal{S}}|$ can be larger than $|\mathcal{S}|$, but we will require $|\hat{\mathcal{S}}|$ to be significantly smaller than p^2. The next lemma, presents a way to construct $\hat{\mathcal{S}}$ by using the gradient of a solution β^w, θ^w that was obtained prior to $\hat{\beta}, \hat{\theta}$ (e.g., from a warm start or a previous iteration of Algorithm 6.2).

Lemma 6.3. *Let $\beta^w \in \mathbb{R}^p$ and $\theta^w \in \mathbb{R}^{\binom{p}{2}}$ be arbitrary vectors, and let $\mathcal{S}$ be the critical set defined in* (6.11)*. Define $\gamma = (X\beta^w + \tilde{X}\theta^w) - (X\hat{\beta} + \tilde{X}\hat{\theta})$ and $C = \max_{i,j} \|\tilde{X}_{ij}\|_2$. Then, the following holds:*

$$\mathcal{S} \subseteq \hat{\mathcal{S}} \stackrel{\text{def}}{=} \{(i,j) \in \mathcal{T}^c \mid |\nabla_{\theta_{ij}} f(\beta^w, \theta^w)| > \lambda_2 - C\|\gamma\|_2\}.$$

The set $\hat{\mathcal{S}}$ in Lemma 6.3 is constructed based on the gradient at a previous estimate (β^w, θ^w); and not at the current point $(\hat{\beta}, \hat{\theta})$. When the predictions made by the estimators $(\hat{\beta}, \hat{\theta})$ and (β^w, θ^w) are close (i.e., small $\|\gamma\|_2$), $\hat{\mathcal{S}}$ is a good estimate of $\mathcal{S}$.

Lemma 6.3 lays the foundation of our *gradient screening* procedure. During the course of Algorithm 6.2, we always maintain a solution β^w, θ^w for which we store $|\nabla_\theta f(\beta^w, \theta^w)|$. We replace step 2 in Algorithm 6.2 with the following gradient screening module:

Gradient Screening

1. Compute $\hat{\mathcal{S}}$ (defined in Lemma 6.3) and $\nabla_{\theta_{\hat{\mathcal{S}}}} f(\hat{\beta}, \hat{\theta})$ to obtain $\nabla_{\theta_{\mathcal{S}}} f(\hat{\beta}, \hat{\theta})$.

2. Compute $\nabla_{\beta} f(\hat{\beta}, \hat{\theta})$ and obtain $\nabla_{\theta_{\mathcal{T}}} f(\hat{\beta}, \hat{\theta})$. Set $\tilde{\beta} \leftarrow \hat{\beta} - \frac{1}{L}\nabla_{\beta} f(\hat{\beta}, \hat{\theta})$ and $\tilde{\theta}_{ij} \leftarrow \hat{\theta}_{ij} - \frac{1}{L}\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})$ for every $(i, j) \in \mathcal{T} \cup \mathcal{S}$.

3. If $|\hat{\mathcal{S}}| > p$, set $(\beta^w, \theta^w) \leftarrow (\hat{\beta}, \hat{\theta})$ and compute/store $|\nabla_{\theta} f(\beta^w, \theta^w)|$.

The set $\hat{\mathcal{S}}$ can be identified in $\mathcal{O}(\log p)$ by using a variant of binary search on the sorted entries of $|\nabla_{\theta} f(\beta^w, \theta^w)|$ (the latter can be sorted once at a cost of $\mathcal{O}(p^2 \log p)$ and stored). Moreover, computing $\nabla_{\theta_{\hat{\mathcal{S}}}} f(\hat{\beta}, \hat{\theta})$ and obtaining $\nabla_{\theta_{\mathcal{S}}} f(\hat{\beta}, \hat{\theta})$ is $\mathcal{O}(n(p + |\hat{\mathcal{S}}|))$. Thus, the complexity of gradient screening can be much smaller than $\mathcal{O}(np^2)$. For gradient screening to yield speed-ups, we need $|\hat{\mathcal{S}}| \ll p^2$ (otherwise, the overall complexity will be similar to that with no gradient screening). Thus, in Step 3 in the above, we update β^w, θ^w when $|\hat{\mathcal{S}}|$ becomes large (recall that improving the estimate β^w, θ^w can reduce the size of $|\hat{\mathcal{S}}|$). In particular, Step 3 is performed when $|\hat{\mathcal{S}}| > p$; we choose the threshold p to ensure that $|\hat{\mathcal{S}}|$ stays in the order of p in subsequent iterations[4]. The initial β^w, θ^w can be obtained from a previous solution in the regularization path. In practice, predictions across consecutive solutions in the path are usually close, making $\hat{\mathcal{S}}$ close to $\mathcal{S}$—this explains the multi-fold speed-ups we observe in our experiments.

6.4 A Custom Nonlinear BnB for Solving the MIP

In this section, we focus on solving the MIP in (6.3) to optimality. As demonstrated in Chapter 3 and also in the experiments of this chapter, state-of-the-art MIP solvers face scalability issues when handling ℓ_0-regularized learning problems. One of the main reasons for the lack of scalability in MIP solvers is that they rely on general-purpose relaxation solvers, which cannot easily exploit sparsity. To overcome this limitation, we design a custom nonlinear BnB framework that exploits the sparsity in the problem to improve the running

[4]Other choices are possible, but it is important that the chosen threshold keeps $|\hat{\mathcal{S}}|$ from growing in the order of p^2.

time. For an overview and discussion on nonlinear BnB, we refer the reader to Chapter 3. First, we discuss how to obtain and solve the root relaxation in our BnB. We obtain the root relaxation by relaxing all the binary variables (z_is and z_{ij}s) in MIP (6.3) to the interval $[0, 1]$. Based on (6.14) (in the proof of Lemma 6.1), the root relaxation can be written equivalently as:

$$\min_{\beta,\theta} \quad f(\beta, \theta) + \Omega(\beta, \theta) \quad \text{s.t. } \|\beta, \theta\|_\infty \leq M \tag{6.12}$$

Note that (6.12) has the same objective function as Problem (6.4), but with the additional box constraint: $\|\beta, \theta\|_\infty \leq M$. Algorithm 6.2 can thus be applied to (6.12) with a slight modification: Problem (6.10) in Algorithm 6.2 should be solved under the box constraint, i.e., Problem (6.10) should be replaced with

$$\begin{aligned} \hat{\beta}, \hat{\theta} \in \arg\min_{\beta,\theta} \quad & f(\beta, \theta) + \Omega(\beta, \theta) \\ \text{s.t.} \quad & \beta_{\mathcal{A}^c} = 0, \quad \theta_{\mathcal{T}^c} = 0, \ \|\beta, \theta\|_\infty \leq M. \end{aligned} \tag{6.13}$$

Therefore, we use Algorithm 6.2 (with the modification discussed above) to solve the root relaxation. We branch based on the binary variables (the z_is and z_{ij}s) in MIP (6.3). At each node of the BnB search tree, we obtain a nonlinear relaxation by relaxing all the free binary variables to $[0, 1]$. The node subproblems are thus similar to (6.12) with minor modifications to account for the z_is and z_{ij}s that are fixed to zero or one (due to branching). Thus, we use Algorithm 6.2 (after accounting for the fixed variables) to solve the node subproblems. Similar to Chapter 3, we share the active sets and warm starts between every parent node and its children, and we use approximate strong branching (where optimization is restricted to the active set). The experiments in Section 6.5.2 indicate that our BnB algorithm can scale to much larger instances compared to Gurobi and MOSEK and can attain better statistical performance (especially in terms of variable selection) compared to existing convex relaxations.

6.5 Experiments

We study the empirical performance of our algorithms and compare with popular toolkits for learning interactions under SH: hierNet [30] and glinternet [108]; in addition to Boosting with trees using XGBoost (which is often used to learn interactions). We also use the optimization toolbox SPAMS [114] to solve (6.4) and compare its running time with hierScale.

Our Toolkit hierScale: We open-sourced our toolkit hierScale (`https://pypi.org/project/hierScale`). The toolkit is written in Python with critical code sections compiled into machine code using Numba [104]. hierScale has a low memory footprint (it generates interaction columns on the fly) and supports multi-core computation.

Synthetic Data Generation: We generate $X_{n\times p}$ to be an i.i.d. standard Gaussian ensemble; and form $y = X\beta^0 + \tilde{X}\theta^0 + \epsilon$, where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ is independent of X. For β^0 and θ^0, we consider three settings: (I) Hierarchical Truth: β^0 and θ^0 satisfy SH, (II) Anti-Hierarchical Truth: $\theta^0_{ij} \neq 0 \implies \beta^0_i = 0, \beta^0_j = 0$, and (III) Main-Only Truth: $\theta^0 = 0$. In (I)–(III), all non-zero coefficients are set to 1. For (I) and (II), we set $\|\theta^0\|_0 = \|\beta^0\|_0$. We take σ^2 such that the signal-to-noise-ratio (SNR) $= \text{Var}(X\beta^0 + \tilde{X}\theta^0)/\sigma^2 = 10$.

6.5.1 Convex Optimization Algorithms

Timings: We compare the running time of hierScale (which implements our convex optimization algorithm) with hierNet, glinternet, and SPAMS, on both synthetic and real datasets. For SPAMS, we use the same objective function of hierScale, and fit a regularization path (with warm starts) using FISTA. We note that SPAMS is not designed to solve interactions problems, so we had to generate the interaction columns apriori, which limits us to problems where the interactions can fit in memory. We generate the synthetic data under hierarchical truth with $n = 1000$ and $\|\beta^0\|_0 = \|\theta^0\|_0 = 5$ and consider p up to $50,000$ ($\sim 10^9$ interactions). Among real data, we consider the Amazon Reviews dataset [86] and use two variants of it: Amazon-1 ($p = 10160$, $n = 1000$) and Amazon-2 ($p = 5000$, $n = 1000$). We also consider the dataset from CoEPrA 2006 (Regression Problem 1) ($p = 5786$, $n = 89$)[5],

[5]Available at `http://www.coepra.org`

and the Riboflavin dataset ($p = 4088, n = 71$) [46]. For all toolkits, we set the tolerance level to 10^{-6}, $\lambda_{\min} = 0.05\lambda_{\max}$ and generate a path with 100 solutions. For hierScale and SPAMS, we set $\lambda_2 = 2\lambda_1$. Computations are carried out on a machine with a 12-core Intel Xeon E5 @ 2.7 GHz and 64GB of RAM.

The results are in Table 6.1. hierScale achieves over a 4900x speed-up compared to hierNet, 700x speed-up compared to SPAMS, and 690x speed-up compared to glinternet (e.g., on the Amazon dataset glinternet cannot terminate in a week). We note that [155] recently proposed an algorithm for a problem similar to ours, but do not provide a public toolkit—the largest problem reported in their paper is for $p = 10^3$: In the best case, their method takes ~ 51 seconds per solution, whereas hierScale takes 0.2 seconds.

Table 6.1: Average time (s) for obtaining a solution in the regularization path. The symbols * and ** indicate that the toolkit does not terminate in 3 days and 1 week, respectively. The dash (-) indicates a crash due to memory issues. The dot indicates that the data matrix could not fit in memory.

Toolkit	Synthetic datasets						Real datasets			
	p=500	1000	2000	5000	15000	50000	Ribo. (p=4088)	Coepra (p=5786)	Amazon-1 (p=5000)	Amazon-2 (p=10160)
hierScale	0.1	0.2	0.5	3.2	26.6	730	2.2	3.3	1.7	8.7
glinternet	0.3	1	3.8	24.4	226.9	*	2.7	7.8	*	**
hierNet	490	-	-	-	-	-	-	-	-	-
SPAMS	17.5	67.9	351.9	.	.	.	1042.3	-	.	.

Variable Selection: We compare the False Discovery Rate (FDR) at different sparsity levels. We generate synthetic data with $n = 100, p = 200$, and $\|\beta^0\|_0 = 10$ and report the FDR averaged over 100 datasets, in Figure 6-1. Under hierarchical truth, all the methods perform roughly similarly. However, under anti-hierarchy or main-only truth, our method can perform significantly better.

Prediction Tasks: We now compare the prediction performance of hierScale with the competing methods on both synthetic and real datasets.

Synthetic data: We use the same dataset as for variable selection (above). We tune on a separate validation set and report the prediction MSE on the testing data (training, val-

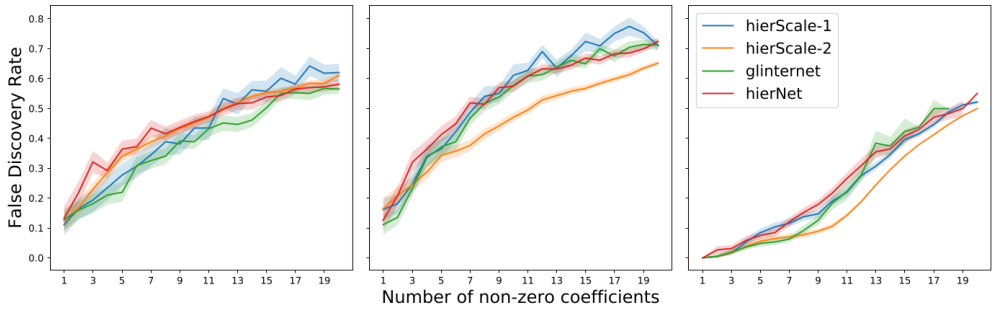


Figure 6-1: False Discovery Rate (FDR) for different methods, under 3 settings (I) Hierarchical Truth, (II) Anti-Hierarchical Truth and (III) Main-Only Truth (left to right). hierScale-1 and hierScale-2 refer to our method with $\lambda_2 = \lambda_1$ and $\lambda_2 = 2\lambda_1$, respectively. The shaded regions correspond to the standard error.

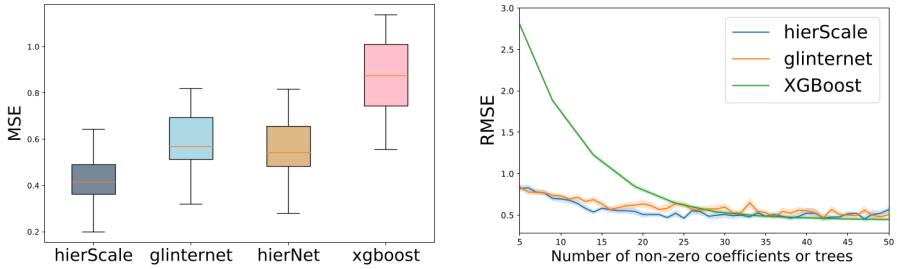


Figure 6-2: Left: Test MSE on the synthetic data (Main-Only truth). **Right**: Test RMSE on the Riboflavin dataset ($n = 50$, $p = 4088$). XGBoost is limited to depth 2. The shaded regions represent the standard error.

idation, and test sets are of the same size). For hierNet and glinternet, we tune over 50 parameter values. For hierScale we use 50 λ_1-values and $\lambda_2 \in \{\lambda_1, 2\lambda_1\}$. For XGBoost we use 50 tree-sizes (between 20 and 1000) and learning rates $\in \{0.01, 0.1\}$. Results across 20 runs (for Main Only Truth) are in Figure 6-2: Our method shows significant improvements in prediction accuracy. The results for hierarchical and anti-hierarchical truth are in the appendix, with results across methods being comparable.

Real data: We consider the Riboflavin dataset ($p = 4088$, $n = 71$) [46] for predicting Vitamin B_2 production from gene expression levels. We train on 50 randomly chosen samples and compute the test RMSE on the remaining 21 samples. To reduce the variability, we repeat this training/testing procedure 30 times and report the average RMSE. We plot the RMSE versus the sparsity level in Figure 6-2. For hierScale we vary $\lambda_2 \in \{\lambda_1, 10\lambda_1, 100\lambda_1\}$, and for XGBoost we vary the learning rate $\in \{0.01, 0.1, 1\}$. Figure 6-2 reports the best RMSE (across the different λ_2's for hierScale the learning rates for XGBoost). We see that the SH methods (hierScale and glinternet) can be more effective than boosting at low/moderate sparsity levels (note $n = 71$ is small), which is desirable for interpretable learning. On this dataset, hierScale shows marginal improvement over glinternet. We note that [108]'s experiments also indicate that SH methods can be more effective than boosted trees in terms of FDR.

6.5.2 Exact MIP Algorithms

In this section, we compare the running time and statistical performance of our BnB algorithm to the state of the art. We implemented our BnB in Python, with critical code sections compiled into machine code using Numba [104].

Running Time: We compare the running time of three solvers for solving MIP (6.3): our BnB, Gurobi, and MOSEK. The datasets were generated under Hierarchical Truth, with $n = 1000, p \in \{100, 500, 5000, 10000\}$, and $\|\beta^0\|_0 = \|\theta^0\|_0 = 5$. We chose α_1 and α_2 in (6.3) so that the optimal solution has 10 non-zeros, and we set[6] $M = 1.2$. We carried out

[6]Our goal is to compare the performance of our proposed BnB solver versus commercial MIQP solvers for a pre-specified user-defined choice of M.

this experiment on a machine with a 12-core Intel Xeon E5 @ 2.7 GHz and 64GB of RAM, running OSX 10.13.6, Gurobi 8.1, and MOSEK 9.1.9 (accessed from CVXPY 1.0.25). In Tables 6.2 and 6.3, we report the running time for solving MIP (6.3) to optimality and for obtaining the best upper bound, respectively. The results indicate that, on these particular datasets, our BnB is more than three orders of magnitude faster than Gurobi and MOSEK, solving problems with 50 million interactions in minutes.

Table 6.2: Time (s) for solving MIP (6.3)

Solver	Appx. # of interactions			
	5000	10^5	10^6	50×10^6
Ours	15	27	285	1216
Gurobi	2576	-	-	*
MOSEK	436	>36000†	-	*

Table 6.3: Time (s) for Best Upper Bd.

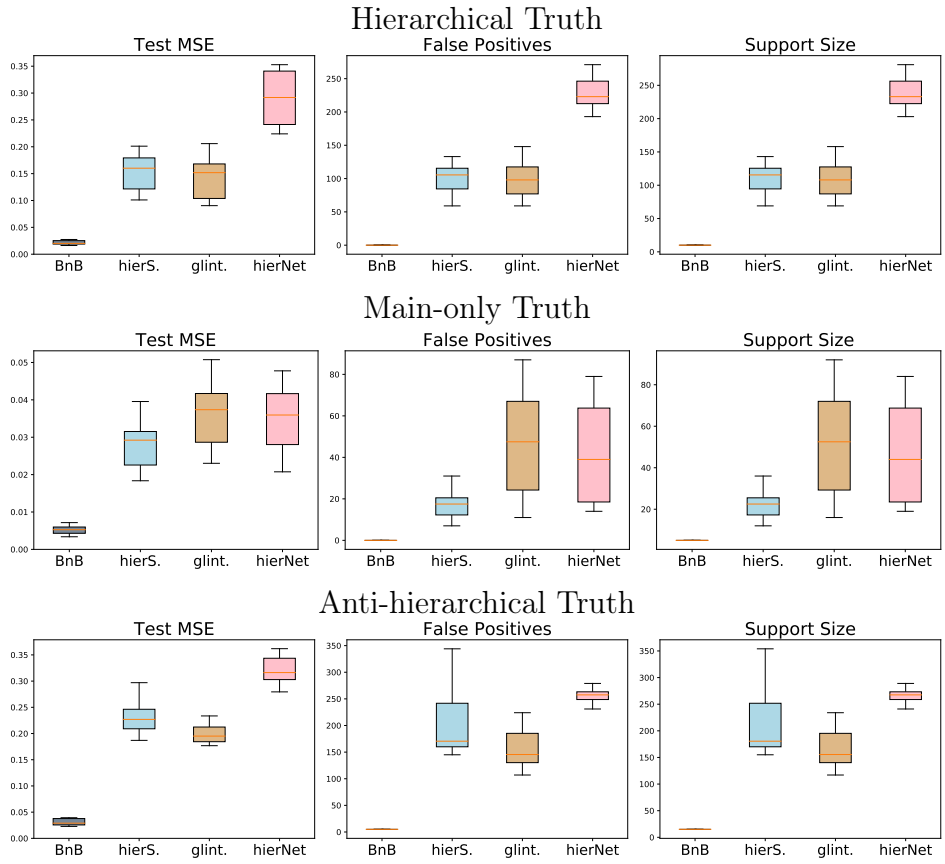
Solver	Appx. # of interactions			
	5000	10^5	10^6	50×10^6
Ours	0.2	0.7	50	205
Gurobi	2530	-	-	*
MOSEK	36.4	∼32400	-	*

"-": Cannot terminate in 24 hours. "*": Cannot fit into memory. "†" Reached a gap of 24% after 10 hours, but was then terminated by the OS due to excessive mem. usage.

Statistical Performance: In this experiment, we compare the statistical performance of MIP (6.3) (solved using our proposed BnB) versus the following state-of-the-art convex relaxations for SH: hierScale (our proposed convex optimization algorithm), glinternet [108], and hierNet [30]. We generate 10 random synthetic datasets for each of Settings (I)-(III), where we set $n = 500, p = 200, \|\beta^0\|_0 = \|\theta^0\|_0 = 5$. We tune the parameters of the algorithms on separate validation sets having the same number of samples as the training sets. For hierScale, glinternet, and hierNet we solve for regularization paths with 50 solutions, whereas for BnB we only solve for 5 solutions (since larger regularization paths can generate many identical solutions). In Figure 6-3, we report box plots of the test MSE, number of false positives, and support size (i.e., total number of non-zeros), for Settings (I)-(III). Overall, the results indicate that our BnB achieves a significantly lower test MSE and false positives compared to the convex relaxations, under all settings. For example, under Setting (I), i.e., Hierarchical Truth, BnB has zero false positives, whereas the best convex relaxation has over 100 on average. However, it is important to note that BnB can solve the corresponding MIP in practical times only when highly sparse solutions are desired (e.g., 10 nonzeros in this experiment). For problems where denser solutions are required, convex optimization based

algorithms, such as our proposal hierScale, may be preferred in practice.

Figure 6-3: Box plots of different statistical metrics for our proposed BnB, which solves MIP (6.3), and three convex-optimization based algorithms: hierScale, glinternet, and hierNet.



6.6 Conclusion and Future Work

We studied the problem of learning sparse interactions under the strong hierarchy constraint. We introduced a transparent convex relaxation for the problem and developed a scalable algorithm based on proximal gradient descent. Our algorithm employs new screening rules for decomposing the proximal problem along with a specialized active-set method and gradient screening for mitigating costly gradient computations. In addition, we developed a nonlinear BnB framework for solving a discrete formulation of the SH problem. The BnB uses the proposed convex optimization algorithm to efficiently solve the node subproblems. Experiments on real and synthetic datasets show that both our convex and discrete algorithms achieve significant speed-ups over the state of the art and more robust variable selection.

There are several promising directions for future work. For instance, our proximal and gradient screening methods can be extended to general group-sparsity problems with overlaps between the group norms, which can potentially speed up the current solvers. Moreover, it would be interesting to establish bounds on the sizes of the connected components and the number of variables eliminated by gradient screening.

6.A Appendix: Proofs and Technical Details

Does the MIP enforce Strong Hierarchy?

We note that there can be a pathological case where the MIP does not satisfy SH. We will briefly discuss why this case corresponds to a zero probability event, when y is drawn from a continuous distribution (this happens for example, if $\epsilon \sim N(0, \sigma^2)$ with $\sigma > 0$). First, we assume that α_1 and α_2 are chosen large enough so that the number of selected variables (as counted by $\sum z_i + \sum_{i<j} z_{ij}$) is less than or equal to the number of samples n. We will also assume that the X_i's and $\tilde{X}_{ij}$'s corresponding to the nonzero z_i's and z_{ij}'s have full rank.

A pathological case can happen when the optimal solution of the MIP satisfies: $z^*_{ij} = 1$, $z^*_i = 1$, $\beta^*_i = 0$, and $\theta^*_{ij} \neq 0$ for some i and j (for example). However, in the latter case, β_i is a free variable, i.e., $|\beta_i| \leq M$ (since $z^*_i = 1$ and we assume M to be sufficiently large). Thus, $\beta^*_i = 0$ is equivalent to saying that a least squares solution on the support defined by the nonzero z^*_i's and z^*_{ij}'s, leads to a coordinate β^*_i which is exactly zero. We know that this is a zero probability event when y is drawn from a continuous distribution (assuming the number of variables is less than or equal to the number of samples and the corresponding columns have full rank).

Proof of Lemma 6.1

Suppose the z_i's and z_{ij}'s are relaxed to $[0, 1]$ and fix some feasible solution β, θ. Let us (partially) minimize the objective function with respect to z, while keeping β, θ fixed, to

obtain a solution z^*. Then, z^* must satisfy $z_i^* = \max\{\frac{|\beta_i|}{M}, \max_{k,j\in G_i} z_{kj}\}$ for every i (since this choice is the smallest feasible z_i). Moreover, $z_{ij}^* = \frac{|\theta_{ij}|}{M}$ for every $i < j$ (by the same reasoning). Substituting the optimal values z_i^* and z_{ij}^* leads to

$$\min_{\beta,\theta} \quad f(\beta,\theta) + \Omega(\beta,\theta) \quad \text{s.t. } \|\beta,\theta\|_\infty \leq M \tag{6.14}$$

where $\Omega(\beta,\theta) \stackrel{\text{def}}{=} \lambda_1 \sum_{i=1}^p \max\{|\beta_i|, \|\theta_{G_i}\|_\infty\} + \lambda_2\|\theta\|_1$ and $\lambda_1 = \alpha_1/M, \lambda_2 = \alpha_2/M$. Finally, we note that the box constraint in the above formulation can be removed, and the resulting formulation is still a valid relaxation.

Dual Reformulation of the Proximal Problem

In this section, we present a dual reformulation of the proximal problem, which will facilitate solving the problem. We note that [99] and [114] dualize a proximal problem which involves sum of ℓ_∞ norms (their proximal problem thus includes ours as a special case). However, the dual we will present here uses $\Theta(p^2)$ less variables as it exploits the presence of the ℓ_1 norm in the objective. First, we introduce some necessary notation. We define the soft-thresholding operator as follows:

$$\mathcal{S}_\gamma(\tilde{v}) = \begin{cases} 0 & \text{if } |\tilde{v}| \leq \gamma \\ (|\tilde{v}| - \gamma)\,\text{sign}(\tilde{v}) & \text{o.w.} \end{cases}$$

We associate every β^i with a dual variable $u^i \in \mathbb{R}$, and every θ_{ij} with two dual variables: $w_j^i \in \mathbb{R}$ and $w_i^j \in \mathbb{R}$. Moreover, we use the notation $w^i \in \mathbb{R}^{p-1}$ to refer to the vector composed of w_j^i for all j such that $j \neq i$.

Theorem 6.3. *(Dual formulation) A dual of the proximal problem is:*

$$\max_{u,w} \quad q(u,w) \quad s.t. \qquad \|(u^i, w^i)\|_1 \leq 1 \quad \forall\ i \in [p] \tag{6.15}$$

where $q(u, w)$ is a continuously differentiable function with a Lipschitz continuous gradient, and

$$\nabla_{u^i} q(u, w) = \lambda_1 \Big(\tilde{\beta}_i - \frac{\lambda_1}{L} u^i \Big)$$

$$\nabla_{w_j^i} q(u, w) = \nabla_{w_i^j} q(u, w) = \lambda_1 \mathcal{S}_{\frac{\lambda_2}{L}} \big(\tilde{\theta}_{ij} - \frac{\lambda_1}{L} (w_j^i + w_i^j) \big).$$

If u^*, w^* is a solution to (6.15), then the solution to the proximal problem is:

$$\beta_i^* = \frac{\nabla_{u^i} q(u^*, w^*)}{\lambda_1} \quad \textit{and} \quad \theta_{ij}^* = \frac{\nabla_{w_j^i} q(u^*, w^*)}{\lambda_1}. \tag{6.16}$$

Proof. Since the ℓ_1 norm is the dual of the l_∞ norm, we have:

$$\max\{|\beta_i|, \|\theta_{G_i}\|_\infty\} = \max_{u^i \in \mathbb{R},\ w^i \in \mathbb{R}^{p-1}} u^i \beta_i + \langle w^i, \theta_{G_i} \rangle \quad \text{s.t.} \|(u^i, w^i)\|_1 \leq 1 \tag{6.17}$$

Plugging the above into the proximal problem and switching the order of the min and max (which is justified by strong duality), we arrive to the dual of the proximal problem:

$$\begin{aligned} \max_{u,w} \min_{\beta,\theta} \frac{L}{2} \left\| \begin{bmatrix} \beta - \tilde{\beta} \\ \theta - \tilde{\theta} \end{bmatrix} \right\|_2^2 + \lambda_1 \sum_i (u^i \beta_i + \langle w^i, \theta_{G_i} \rangle) + \lambda_2 \|\theta\|_1 \\ \text{s.t. } \|(u^i, w^i)\|_1 \leq 1, \forall i \in [p] \end{aligned} \tag{6.18}$$

Note that each dual variable u^i is a scalar which corresponds to the primal variable β^i. Similarly, the dual vector $w^i \in \mathbb{R}^{p-1}$ corresponds to θ_{G_i}. The term $\sum_i (u^i \beta_i + \langle w^i, \theta_{G_i} \rangle)$ in (6.18) can be written as $\sum_i u^i \beta_i + \sum_{i<j} \theta_{ij} (w_j^i + w_i^j)$, where w_j^i and w_i^j are the components of the vectors w^i and w^j, respectively, corresponding to θ_{ij}. Using this notation, we can rewrite Problem (6.18) as follows:

$$\begin{aligned} \max_{u,w} \min_{\beta,\theta} \sum_i h(\beta_i, u^i; \tilde{\beta}_i) + \sum_{i<j} g(\theta_{ij}, w_j^i, w_i^j; \tilde{\theta}_{ij}) \\ \text{s.t. } \|(u^i, w^i)\|_1 \leq 1, \forall i \in [p] \end{aligned} \tag{6.19}$$

where

$$h(a,b;\tilde{a}) \stackrel{\text{def}}{=} \frac{L}{2}(a-\tilde{a})^2 + \lambda_1 ab \quad \text{and} \quad g(a,b,c;\tilde{a}) \stackrel{\text{def}}{=} \frac{L}{2}(a-\tilde{a})^2 + \lambda_1 a(b+c) + \lambda_2|a|$$

The optimal solution of the inner minimization in (6.19) is (uniquely) given by:

$$\begin{aligned} \beta_i^* &\stackrel{\text{def}}{=} \arg\min_{\beta_i} h(\beta_i, u^i; \tilde{\beta}_i) = \tilde{\beta}_i - \frac{\lambda_1}{L} u^i \\ \theta_{ij}^* &\stackrel{\text{def}}{=} \arg\min_{\theta_{ij}} g(\theta_{ij}, w_j^i, w_i^j; \tilde{\theta}_{ij}) = \mathcal{S}_{\frac{\lambda_2}{L}}\left(\tilde{\theta}_{ij} - \frac{\lambda_1}{L}(w_j^i + w_i^j)\right) \end{aligned} \tag{6.20}$$

Therefore, the dual problem can equivalently written as:

$$\max_{u,w} \underbrace{\sum_i h(\beta_i^*, u^i; \tilde{\beta}_i) + \sum_{i<j} g(\theta_{ij}^*, w_j^i, w_i^j; \tilde{\theta}_{ij})}_{q(u,w)} \quad \text{s.t. } \|(u^i, w^i)\|_1 \leq 1 \quad \forall\ i \tag{6.21}$$

Finally, since the solution β^*, θ^* is defined in (6.20) is unique, Danskin's theorem implies that the dual objective function $q(u,w)$ is continuously differentiable and that

$$\nabla_{u^i} q(u,w) = \lambda_1 \beta_i^* \quad \text{and} \quad \nabla_{w_j^i} q(u,w) = \nabla_{w_i^j} q(u,w) = \lambda_1 \theta_{ij}^*. \tag{6.22}$$

□

In problem (6.15), the separability of the feasible set across the (u^i, w^i)'s and the smoothness of $q(u,w)$ make the problem well-suited for the application of block coordinate ascent (BCA) [22, 165], which optimizes with respect to a single block at a time. When updating a particular block in BCA, we perform inexact maximization by taking a step in the direction of the gradient of the block and projecting the resultant vector onto the feasible set, i.e., the ℓ_1 ball. We present the algorithm more formally below.

Algorithm 6.3: BCA for Solving (6.15)

- Initialize with u, w and take step size $\alpha_i, i \in [p]$.
- For $i \in [p]$ perform updates (till convergence):

$$\begin{bmatrix} u^i \\ w^i \end{bmatrix} \leftarrow \mathcal{P}_{\|.\|_1 \leq 1}\left(\begin{bmatrix} u^i \\ w^i \end{bmatrix} + \alpha_i \nabla_{u^i, w^i} q(u, w) \right),$$

where for a vector a, $\mathcal{P}_{\|.\|_1 \leq 1}(a)$ denotes projection of a onto the unit ℓ_1-ball.

The Lipschitz parameter of $\nabla_{u^i, w^i} q(u, w)$ is given by $L_i = p\frac{\lambda_1^2}{L}$ (this follows by observing that each component of $\nabla_{u^i, w^i} q(u, w)$ is a piece-wise linear function with a maximal slope of $\frac{\lambda_1^2}{L}$). Thus, by standard results on block coordinate descent (e.g., [18, 22]), Algorithm 6.3 with step size $\alpha_i = \frac{1}{L_i}$ converges at a rate of $\mathcal{O}(\frac{1}{t})$ (where t is the iteration counter). We note that BCA has been applied to the dual of structured sparsity problems (e.g., [99, 181])—however, the duals considered in the latter works are different.

Proof of Theorem 6.1

We prove the theorem using the dual reformulation presented in Theorem (6.3) and the block coordinate ascent (BCA) algorithm presented in Section 6.A. Suppose $\sum_{t \in G_i} \max\{|\tilde{\theta}_t| - \frac{\lambda_2}{L}, 0\} \leq \frac{\lambda_1}{L} - |\tilde{\beta}_i|$ is satisfied for some i. Let u, w be some feasible solution to Problem (6.15) (e.g., solution of all zeros). Now update u, w as follows:

$$u^i = \frac{L}{\lambda_1}\tilde{\beta}_i, \tag{6.23}$$

and for every $t \in G_i$, let j be the index in t different from i and set:

$$w_j^i = \max\left\{ \frac{L}{\lambda_1}|\tilde{\theta}_t| - \frac{\lambda_2}{\lambda_1}, 0 \right\} \operatorname{sign}(\tilde{\theta}_t) \quad \text{and} \quad w_i^j = 0 \tag{6.24}$$

It is easy to check that u, w is still feasible for Problem (6.15) after this update and that $\nabla_{u^i, w^i} q(u, w) = 0$ and $\nabla_{w_i^j} q(u, w) = 0$ for every j. Thus, BCA will never change u^i, w^i, or

w_i^j (for any j) in subsequent iterations. Since BCA is guaranteed to converge to an optimal solution, we conclude that the values in (6.23) and (6.24) (which correspond to $\beta_i^*, \theta_{G_i}^* = 0$) are optimal.

For the case when $|\tilde{\theta}_{ij}| \leq \frac{\lambda_2}{L}$, if we set $w_j^i = w_i^j = 0$, then $\nabla_{w_j^i} q(u, w) = \nabla_{w_i^j} q(u, w) = 0$, so BCA will never change w_j^i or w_i^j, which leads to $\theta_{ij}^* = 0$.

Proof of Theorem 6.2

By Theorem 6.1, we have $\beta_{\mathcal{V}^c}^* = 0, \quad \theta_{\mathcal{E}^c}^* = 0$. Plugging the latter into the proximal problem leads to:

$$\begin{bmatrix} \beta_{\mathcal{V}}^* \\ \theta_{\mathcal{E}}^* \end{bmatrix} = \underset{\beta_{\mathcal{V}}, \theta_{\mathcal{E}}}{\arg\min} \frac{L}{2} \left\| \begin{bmatrix} \beta_{\mathcal{V}} - \tilde{\beta}_{\mathcal{V}} \\ \theta_{\mathcal{E}} - \tilde{\theta}_{\mathcal{E}} \end{bmatrix} \right\|_2^2 + \Omega(\beta_{\mathcal{V}}, \theta_{\mathcal{E}}) \tag{6.25}$$

where $\Omega(\beta_{\mathcal{V}}, \theta_{\mathcal{E}}) = \lambda_1 \sum_{i \in \mathcal{V}} \max\{|\beta_i|, \|\theta_{\tilde{G}_i}\|_\infty\} + \lambda_2 \|\theta_{\mathcal{E}}\|_1$ and $\tilde{G}_i = G_i \setminus \mathcal{E}^c$. But by the definition of the connected components, the following holds:

$$\Omega(\beta_{\mathcal{V}}, \theta_{\mathcal{E}}) = \sum_{l \in [k]} \Big[\lambda_1 \sum_{i \in \mathcal{V}_l} \max\{|\beta_i|, \|\theta_{\tilde{G}_i}\|_\infty\} + \lambda_2 \|\theta_{\mathcal{E}_l}\|_1 \Big] = \sum_{l \in [k]} \Omega(\beta_{\mathcal{V}_l}, \theta_{\mathcal{E}_l}).$$

The above implies that the proximal problem is separable across the blocks $\beta_{\mathcal{V}_l}, \theta_{\mathcal{E}_l}$, which leads to the result of the theorem.

Proof of Lemma 6.2

First, note that the full gradient $\nabla_{\beta,\theta} f(\hat{\beta}, \hat{\theta})$ is sufficient for constructing $\mathcal{G}$ (see steps 2 and 3 of Algorithm 6.2). The (i, j)'s in $\mathcal{T}^c$ whose $|\tilde{\theta}_{ij}| \leq \lambda_2 / L$ are not needed to construct $\mathcal{G}$ (this follows from the definitions of $\mathcal{V}$ and $\mathcal{E}$). For every $(i, j) \in \mathcal{T}^c$, we have $\hat{\theta}_{ij} = 0$, so the condition $|\tilde{\theta}_{ij}| \leq \lambda_2 / L$ is equivalent to $|\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})| \leq \lambda_2$ (see the definition of $\tilde{\theta}_{ij}$ in step 2 of Algorithm 6.2). Thus, the (i, j)'s in $\mathcal{T}^c$ with $|\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})| \leq \lambda_2$ are not needed to construct $\mathcal{G}$. The latter indices are exactly those in $\mathcal{S}^c$. Thus, the remaining parts of the gradient are: $\nabla_\beta f(\hat{\beta}, \hat{\theta})$, $\nabla_{\theta_\mathcal{T}} f(\hat{\beta}, \hat{\theta})$, and $\nabla_{\theta_\mathcal{S}} f(\hat{\beta}, \hat{\theta})$—these are sufficient to construct $\mathcal{G}$.

Proof of Lemma 6.3

Let $(i,j) \in \mathcal{S}$. By the triangle inequality:

$$|\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})| \leq |\nabla_{\theta_{ij}} f(\beta^w, \theta^w)| + |\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta}) - \nabla_{\theta_{ij}} f(\beta^w, \theta^w)| \tag{6.26}$$

Writing down the gradients explicitly and using Cauchy-Schwarz, we get:

$$\begin{aligned} |\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta}) - \nabla_{\theta_{ij}} f(\beta^w, \theta^w)| &= |\tilde{X}_{ij}^T (y - X\hat{\beta} - \tilde{X}\hat{\theta}) - \tilde{X}_{ij}^T (y - X\beta^w - \tilde{X}\theta^w)| \\ &\leq \|\tilde{X}_{ij}\|_2 \|(X\hat{\beta} + \tilde{X}\hat{\theta}) - (X\beta^w + \tilde{X}\theta^w)\|_2 \\ &\leq C\|\gamma\|_2 \end{aligned}$$

Plugging the upper bound above into (6.26), we get $|\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})| \leq |\nabla_{\theta_{ij}} f(\beta^w, \theta^w)| + C\|\gamma\|_2$. Therefore, if $(i,j) \in \mathcal{S}$, i.e., $|\nabla_{\theta_{ij}} f(\hat{\beta}, \hat{\theta})| > \lambda_2$ then $|\nabla_{\theta_{ij}} f(\beta^w, \theta^w)| + C\|\gamma\|_2 > \lambda_2$, implying that $(i,j) \in \hat{\mathcal{S}}$.

6.B Appendix: Additional Experimental Results

Sizes of Connected Components For the Riboflavin ($p = 4088$, $n = 71$) and Coepra ($p = 5786$, $n = 89$) datasets (discussed in this chapter), we fit a regularization path with 100 solutions using Algorithm 6.2. In Table 6.4, we report the maximum number of edges and vertices encountered across all the connected components and for all the 100 solutions in the path.

Table 6.4: Maximum size of the connected components across a regularization path of 100 solutions.

Dataset	$\lambda_2 = 2\lambda_1$		$\lambda_2 = \lambda_1$	
	# Edges	# Vertices	# Edges	# Vertices
Ribo	350	149	1855	693
Coepra	227	86	400	103

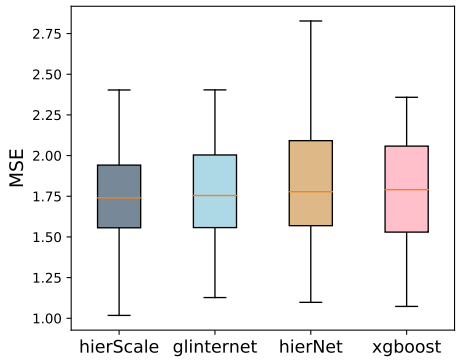


Figure 6-4: MSE on the test set for synthetic data (Anti-Hierarchical truth).

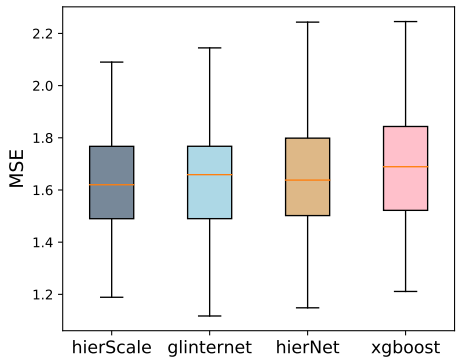


Figure 6-5: MSE on the test set for synthetic data (Hierarchical truth).

Chapter 7

Conclusion and Additional Work

Conclusion

This thesis presented scalable discrete optimization methods and new statistical insights for a fundamental class of sparse learning problems. In Chapters 2 and 3, we considered the problem of ℓ_0-regularized linear regression (with additional convex regularizers). Specifically, in Chapter 2, we developed fast, approximate algorithms, based on cyclic coordinate descent and local combinatorial optimization, and established convergence guarantees. In Chapter 3, we proposed a specialized nonlinear branch-and-bound (BnB) framework that can solve the problem to global optimality. By using a tailored first-order method to solve the node subproblems, our BnB can scale to problems with up to $\sim 10^7$ features, which is over 1000x larger than what commercial MIP solvers can handle. In Chapter 4, we developed both exact and approximate algorithms for ℓ_0-regularized classification. We also established new statistical bounds for a family of ℓ_0-regularized classification problems. In Chapter 5, we considered the problem of grouped variable selection (based on ℓ_0 regularization), which includes both the problems of sparse linear regression and nonparametric sparse additive modeling. We generalized the approximate and exact algorithms proposed in Chapters 1 and 2 to this setting, and established new statistical bounds. In Chapter 6, we considered the problem of sparse regression with pairwise interactions, under the strong hierarchy constraint.

For a convex formulation of the problem, we proposed a specialized first-order method that can scale to $\sim 10^9$ interactions. We also developed a specialized nonlinear BnB that solves a discrete formulation of the problem, and which can scale to problems with up to 50 million interactions.

In Chapters 2, 4, 5, and 6, we performed systematic experiments to gain insights into the performance of ℓ_0-based estimators compared to popular sparse learning methods (for example, methods based on the ℓ_1, MCP, and SCAD penalties or greedy procedures). For all the problems considered, we observed that if the noise strength is sufficiently low, then ℓ_0-based estimators can outperform the competing methods in terms of various statistical metrics (variable selection, prediction, and estimation). The improvements are most pronounced when some features are highly correlated or when the sample size is relatively small compared to the number of features. However, if the noise strength is high, pure ℓ_0 regularization can suffer from overfitting. The combination of ℓ_0 and ℓ_2 regularization appears to mitigate this issue. Indeed, the $\ell_0\ell_2$-regularized estimators we considered, outperformed the competing sparse learning methods over a wide range of noise levels and various statistical regimes.

There are multiple exciting directions for future work in sparse learning and, more broadly, in mixed integer programming. Given the scalability and the promising statistical performance of the discrete optimization methods we presented, it is natural to extend these methods to more general sparse learning problems. One important example is multitask learning, where the goal is to exploit the relationships between different learning tasks to improve statistical performance [49]. In multitask learning, structured sparsity is commonly used to control parameter sharing between tasks, so we think that generalizations of our approach in Chapters 5 and 6 can be useful for this problem.

More generally, we believe that first-order methods can have an important role in mixed integer programming, especially in problems where sparse solutions are desired. As demonstrated throughout this thesis, first-order methods can be effective for obtaining good quality warm starts for MIPs and also for solving the continuous subproblems in BnB. Thus, it would

be interesting to study the more general classes of MIP problems in which first-order methods can be applicable and useful.

Additional Work

Here we briefly discuss *conditional computation*, a structured sparsity problem that is recently gaining attention in (differentiable) decision trees and neural networks. This problem can be viewed as a generalization of the problems we considered in Chapters 5 and 6. Conditional computation refers to the ability of a learning model to activate parts of its architecture on a per-example basis, i.e., each input example (data point) will propagate through a small part of the model's architecture [21, 91]. One popular model that naturally supports conditional computation is a decision tree: each example is routed through a single root-to-leaf path. Thus, conditional computation can be thought of as a form of structured (hierarchical) sparsity, in which sparsity is imposed on a per-example basis. Since a relatively small number of parameters is activated by each example, conditional computation can lead to more efficient computation and can also enhance interpretability [21, 91].

While considered state-of-the-art learners in many applications, neural networks do not naturally support conditional computation. Indeed, neural networks perform a dense form of computation, in which every example is processed at all the units in the network. In [91][1], we equip neural networks with a new model that enables conditional computation, with the goal of enhancing computational efficiency and interpretability. Specifically, we propose the *tree ensemble layer* (TEL): a new layer for neural networks, composed of an ensemble of decision trees that support conditional computation. TEL can be used anywhere in a neural network, similar to a dense layer. Moreover, TEL is continuously differentiable, which is an important property that allows for end-to-end training using first-order methods. Prior to TEL, the state-of-the-art techniques for conditional computation in neural networks were either non-differentiable or used post-processing heuristics to achieve conditional computation at inference time (see [91] for a literature survey). Below we give an overview of TEL and our technical contributions in [91].

[1] This is a joint work with Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder.

Since standard decision trees are not continuous [68], integrating them into a neural network is challenging. To make TEL differentiable, we rely on soft trees—a differentiable variant of decision trees that is based on the hierarchical mixture of experts [100]. On a high-level, soft trees are similar to classical decision trees, but their internal nodes (i) use hyperplane splits instead of axis-aligned splits (i.e., they split on a linear combination of variables), and (ii) use a probabilistic routing mechanism that routes each example simultaneously to the left and right, with different proportions (as opposed to classical decision trees, which route each example in exactly one direction). Due to the probabilistic routing mechanism, soft trees do not support conditional computation (specifically, each example propagates to all the leaves). Next, we give a simple example of how a soft tree works.

Simple Example of a Soft Tree: Let us consider a soft (binary) tree with depth 1, i.e., with one internal node and two leaves. The root node stores a (learnable) vector $w \in \mathbb{R}^p$ that defines the hyperplane split. The left and right leaves store the (learnable) weights $o_1 \in \mathbb{R}$ and $o_2 \in \mathbb{R}$, respectively. Given an example $x \in \mathbb{R}^p$, the root node routes x to the left with probability $\sigma(w^T x)$ and to the right with probability $1 - \sigma(w^T x)$, where σ is the logistic function. The output of the tree is defined as an expectation over the two leaves, i.e., $\sigma(w^T x)o_1 + (1 - \sigma(w^T x))o_2$. Since the logistic function outputs values in $(0, 1)$, the example x will reach both leaves with positive probability, and thus conditional computation is not supported. For a more formal description of soft trees, see [91].

In [91], we propose an enhancement of the standard soft tree model that enables conditional computation, while still being differentiable. We achieve this by introducing the *smooth-step function*, a continuously differentiable approximation of the step function, which we use for routing. In Figure 7-1, we plot the smooth-step function versus the (rescaled) logistic function $f(t) = (1 + e^{-6t})^{-1}$. The smooth-step function is a cubic piece-wise polynomial, which is S-shaped, similar to the logistic function. However, unlike the logistic function, the smooth-step function can output an exact zero or one when the magnitude of the input is sufficiently large (specifically, larger than 0.5, as shown in Figure 7-1). In [91], we propose replacing the logistic function with the smooth-step function for performing example routing in soft trees. At any internal node, if the smooth-step function outputs a zero or one probability, then

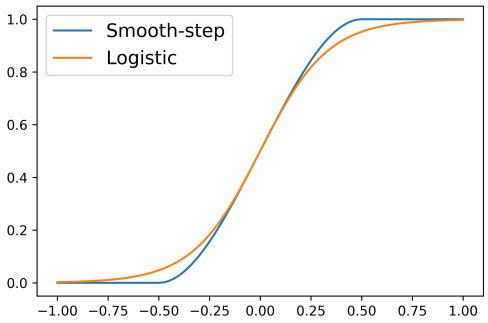


Figure 7-1: The smooth-step function versus a (rescaled) logistic function.

the corresponding example will be routed in exactly a single direction (similar to classical decision trees). We exploit the latter observation and demonstrate empirically that, with careful initialization, our soft trees can be trained using first-order methods while supporting conditional computation. Although our proposed model supports conditional computation, automatic differentiation frameworks (such as TensorFlow [1]) cannot exploit conditional computation to speed up training or inference. Thus, in [91], we propose specialized forward and backward propagation algorithms that exploit sparsity to speed up computation. We provide an open-source implementation of our proposed model and algorithms. Empirically, we demonstrate that TEL can lead to over 10x speed-ups compared to standard soft trees. Moreover, on a collection of 26 real datasets, we show that TEL is competitive with state-of-the-art models, such as gradient boosted decision trees [68] and dense layers, in terms of compactness and predictive performance.

For more details on TEL, we refer the reader to [91]. Overall, TEL offers an interesting bridge between continuous and discrete optimization. In particular, our proposed smooth-step function is continuous, but when trained using first-order methods, it leads to discrete (zero or one) values. We think that generalizations of this idea may be effective for learning discrete structures in complex models (such as neural networks) that require continuous optimization. For example, see our recent work [92], in which we leverage the smooth-step function to learn sparse mixtures of experts.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, and Michael Isard. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[3] S Agmon. *Lectures on Elliptic Boundary Value Problems*. Van Nostrand, Princeton, NJ, 1965.

[4] M Selim Aktürk, Alper Atamtürk, and Sinan Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, 37(3):187–191, 2009.

[5] Erling D Andersen, Cornelis Roos, and Tamas Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2):249–277, 2003.

[6] David Applegate, Robert Bixby, William Cook, and Vasek Chvátal. On the solution of traveling salesman problems. *Documenta Mathematica*, 1998.

[7] Alper Atamturk and Andres Gomez. Rank-one convexification for sparse regression. *arXiv preprint arXiv:1901.10334*, 2019.

[8] Alper Atamturk and Andres Gomez. Safe screening rules for l0-regression from perspective relaxations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 421–430. PMLR, 13–18 Jul 2020.

[9] F.R. Bach. Consistency of the group lasso and multiple learning kernel. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

[10] Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373, 2013.

[11] Francis R Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in neural information processing systems*, pages 105–112, 2009.

[12] Francis R Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems*, pages 118–126, 2010.

[13] Sohail Bahmani, Bhiksha Raj, and Petros T Boufounos. Greedy sparsity-constrained optimization. *Journal of Machine Learning Research*, 14(Mar):807–841, 2013.

[14] EML Beale, MG Kendall, and DW Mann. The discarding of variables in multivariate analysis. *Biometrika*, 54(3-4):357–366, 1967.

[15] Amir Beck. *First-Order Methods in Optimization*, volume 25. SIAM, 2017.

[16] Amir Beck and Yonina C. Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM Journal on Optimization*, 23(3):1480–1509, 2013.

[17] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[18] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

[19] Alexandre Belloni and Victor Chernozhukov. l1-penalized quantile regression in high-dimensional sparse models. *The Annals of Statistics*, 39(1):82–130, 2011.

[20] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 05 2013.

[21] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *CoRR*, abs/1511.06297, 2015.

[22] D.P. Bertsekas. *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific, 2016.

[23] Dimitris Bertsimas and Angela King. Logistic regression: From art to science. *Statistical Science*, 32(3):367–384, 2017.

[24] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *Annals of Statistics*, 44(2):813–852, 2016.

[25] Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. Sparse classification: a scalable discrete optimization perspective. *arXiv preprint arXiv:1710.01352*, 2017.

[26] Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. Sparse regression: Scalable algorithms and empirical performance. *arXiv preprint arXiv:1902.06547*, 2019.

[27] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[28] Dimitris Bertsimas, Bart Van Parys, et al. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics*, 48(1):300–323, 2020.

[29] Peter J Bickel, Ya'acov Ritov, and Alexandre B Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

[30] Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

[31] M. S. Birman and M. Z. Solomjak. Piecewise-polynomial approximations of functions of the classes w_p^α. *Math. USSR-Sbornik*, 2(3):295–317, 1967.

[32] Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica, Extra Volume: Optimization Stories*, pages 107–121, 2012.

[33] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[34] Christian Bliek1ú, Pierre Bonami, and Andrea Lodi. Solving mixed-integer quadratic programming problems with ibm-cplex: a progress report. In *Proceedings of the twenty-sixth RAMP symposium*, pages 16–17, 2014.

[35] Thomas Blumensath and Mike Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5-6):629–654, 2008.

[36] Thomas Blumensath and Mike Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[37] Pierre Bonami, Jon Lee, Sven Leyffer, and Andreas Wächter. More branch-and-bound experiments in convex nonlinear integer programming. *Preprint ANL/MCS-P1949-0911, Argonne National Laboratory, Mathematics and Computer Science Division*, 2011.

[38] Antoine Bonnefoy, Valentin Emiya, Liva Ralaivola, and Remi Gribonval. Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso. *IEEE Transactions on Signal Processing*, 63(19):5121–5132, 2015.

[39] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[40] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[41] Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The annals of applied statistics*, 5(1):232, 2011.

[42] Patrick Breheny and Jian Huang. Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and computing*, 25(2):173–187, 2015.

[43] Peter Bühlmann and Sara van-de-Geer. *Statistics for high-dimensional data*. Springer, 2011.

[44] Peter Bühlmann and Sara Van De Geer. *Statistics for High-dimensional Data: Methods, Theory and Applications*. Springer Science & Business Media, 2011.

[45] Florentina Bunea, Alexandre B Tsybakov, and Marten H Wegkamp. Aggregation for gaussian regression. *The Annals of Statistics*, 35(4):1674–1697, 2007.

[46] Peter Bühlmann, Markus Kalisch, and Lukas Meier. High-dimensional statistics with a view toward applications in biology. *Annual Review of Statistics and Its Application*, 1(1):255–278, 2014.

[47] Emmanuel Candes and Mark A Davenport. How well can we estimate a sparse vector? *Applied and Computational Harmonic Analysis*, 34(2):317–323, 2013.

[48] Emmanuel J Candes and Terence Tao. The Dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, pages 2313–2351, 2007.

[49] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[50] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[51] Xi Chen, Qihang Lin, Seyoung Kim, Jaime G Carbonell, and Eric P Xing. Smoothing proximal gradient method for general structured sparse regression. *The Annals of Applied Statistics*, 6(2):719–752, 2012.

[52] C. Chesneau and M. Hebiri. Some theoretical results on the grouped variables lasso. *Mathematical Methods of Statistics*, 17:317–326, 2008.

[53] Hugh Chipman. Bayesian variable selection with related predictors. *Canadian Journal of Statistics*, 24(1):17–36, 1996.

[54] Nam Hee Choi, William Li, and Ji Zhu. Variable selection with the strong heredity constraint and its oracle property. *Journal of the American Statistical Association*, 105(489):354–364, 2010.

[55] Jens Clausen and Michael Perregaard. On the best search strategy in parallel branch-and-bound: Best-first search versus lazy depth-first search. *Annals of Operations Research*, 90:1–17, 1999.

[56] David R Cox. Interaction. *International Statistical Review/Revue Internationale de Statistique*, pages 1–24, 1984.

[57] Alison Cozad, Nikolaos V. Sahinidis, and David C. Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227, 2014.

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

[59] Gamarnik David and Zadik Ilias. High dimensional regression with binary coefficients. estimating squared error and a phase transtition. In *Conference on Learning Theory*, pages 948–953, 2017.

[60] Antoine Dedieu, Hussein Hazimeh, and Rahul Mazumder. Learning sparse classifiers: Continuous and mixed integer optimization perspectives. *arXiv preprint arXiv:2001.06471*, 2020.

[61] H. Dong, K. Chen, and J. Linderoth. Regularization vs. Relaxation: A conic optimization perspective of statistical variable selection. *ArXiv e-prints*, October 2015.

[62] Norman R Draper and Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 1998.

[63] Marco A Duran and Ignacio E Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, 36(3):307–339, 1986.

[64] Chandra Erdman and Nancy Bates. The us census bureau mail return rate challenge: Crowdsourcing to develop a hard-to-count score. *Statistics*, page 08, 2014.

[65] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[66] Alyson K Fletcher, Sundeep Rangan, and Vivek K Goyal. Necessary and sufficient conditions for sparsity pattern recovery. *IEEE Transactions on Information Theory*, 55(12):5758–5772, 2009.

[67] Antonio Frangioni and Claudio Gentile. Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106(2):225–236, 2006.

[68] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[69] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[70] David Gamarnik and Ilias Zadik. High dimensional regression with binary coefficients. estimating squared error and a phase transtition. In *Conference on Learning Theory*, pages 948–953, 2017.

[71] Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv preprint arXiv:1009.4219*, 2010.

[72] Pinghua Gong, Changshui Zhang, Zhaosong Lu, Jianhua Huang, and Jieping Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *International Conference on Machine Learning*, pages 37–45, 2013.

[73] Eitan Greenshtein et al. Best subset selection, persistence in high-dimensional statistical learning and optimization under l1 constraint. *The Annals of Statistics*, 34(5):2367–2386, 2006.

[74] Oktay Günlük and Jeff Linderoth. Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Mathematical programming*, 124(1-2):183–205, 2010.

[75] Mert Gurbuzbalaban, Asuman Ozdaglar, Pablo A Parrilo, and Nuri Vanli. When cyclic coordinate descent outperforms randomized coordinate descent. In *Advances in Neural Information Processing Systems*, pages 6999–7007, 2017.

[76] Incorporate Gurobi Optimization. Gurobi optimizer reference manual. *URL http://www. gurobi. com*, 2020.

[77] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, pages 545–552, 2005.

[78] Sven Hammarling and Craig Lucas. Updating the qr factorization and the least squares problem. *Manchester Institute for Mathematical Sciences, University of Manchester*, 2008.

[79] Ning Hao and Hao Helen Zhang. Interaction screening for ultrahigh-dimensional data. *Journal of the American Statistical Association*, 109(507):1285–1301, 2014.

[80] David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81 – 102, 1978.

[81] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[82] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, 1990.

[83] T. Hastie, R. Tibshirani, and R. J. Tibshirani. Extended Comparisons of Best Subset Selection, Forward Stepwise Selection, and the Lasso. *ArXiv e-prints*, July 2017.

[84] Trevor Hastie, Robert Tibshirani, and Ryan Tibshirani. Best subset, forward stepwise or lasso? analysis and recommendations based on extensive comparisons. *Statist. Sci.*, 35(4):579–592, 11 2020.

[85] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[86] Hussein Hazimeh and Rahul Mazumder. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Operations Research*, 68(5):1517–1537, 2020.

[87] Hussein Hazimeh and Rahul Mazumder. Learning hierarchical interactions at scale: A convex optimization approach. In *International Conference on Artificial Intelligence and Statistics*, pages 1833–1843. PMLR, 2020.

[88] Hussein Hazimeh, Rahul Mazumder, and Tim Nonet. *L0Learn: Fast Algorithms for Best Subset Selection*, 2021. R package version 2.0.3.

[89] Hussein Hazimeh, Rahul Mazumder, and Peter Radchenko. Grouped variable selection with discrete optimization: Computational and statistical perspectives. *arXiv preprint arXiv:2104.07084*, 2021.

[90] Hussein Hazimeh, Rahul Mazumder, and Ali Saab. Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *arXiv preprint arXiv:2004.06152*, 2020.

[91] Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

[92] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed H Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *arXiv preprint arXiv:2106.03760*, 2021.

[93] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 507–517, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[94] Ronald R Hocking and RN Leslie. Selection of the best subset in regression analysis. *Technometrics*, 9(4):531–540, 1967.

[95] Mingyi Hong, Xiangfeng Wang, Meisam Razaviyayn, and Zhi-Quan Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, 163(1-2):85–114, 2017.

[96] J. Huang, B. Breheny, and S. Ma. A selective review of group selection in high-dimensional models. *Statistical Science*, 27:481–499, 2012.

[97] J. Huang, J.L. Horowitz, and F. Wei. Variable selection in nonparametric additive models. *The Annals of Statistics*, 38:2282–2313, 2010.

[98] J. Huang and T. Zhang. The benefit of group sparsity. *The Annals of Statistics*, 38:1978–2004, 2010.

[99] Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12(Jul):2297–2334, 2011.

[100] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[101] Michael Jünger, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A Wolsey. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-art*. Springer Science & Business Media, 2009.

[102] Vladimir Koltchinskii and Ming Yuan. Sparsity in multiple kernel learning. *The Annals of Statistics*, 38(6):3660–3695, 2010.

[103] Ja-Yong Koo, Yoonkyung Lee, Yuwon Kim, and Changyi Park. A Bahadur representation of the linear support vector machine. *Journal of Machine Learning Research*, 9(Jul):1343–1368, 2008.

[104] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[105] Jon Lee and Sven Leyffer. *Mixed integer nonlinear programming*, volume 154. Springer Science & Business Media, 2011.

[106] Seunghak Lee and Eric P Xing. Screening rules for overlapping group lasso. *arXiv preprint arXiv:1410.6880*, 2014.

[107] Huan Li and Zhouchen Lin. Accelerated proximal gradient methods for nonconvex programming. *Advances in Neural Information Processing Systems*, 28:379–387, 2015.

[108] Michael Lim and Trevor Hastie. Learning interactions via hierarchical group-lasso regularization. *Journal of Computational and Graphical Statistics*, 24(3):627–654, 2015.

[109] Y. Lin and H. H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics*, 34:2272–2297, 2006.

[110] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

[111] K. Lounici, M. Pontil, S. van de Geer, and A. Tsybakov. Oracle inequalities and optimal inference under group sparsity. *The Annals of Statistics*, 39(4):2164–2204, 2011.

[112] Zhaosong Lu. Iterative hard thresholding methods for l0 regularized convex cone programming. *Mathematical Programming*, 147(1):125–154, Oct 2014.

[113] Miles Lubin, Emre Yamangil, Russell Bent, and Juan Pablo Vielma. Extended formulations in mixed-integer convex programming. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 102–113. Springer, 2016.

[114] Julien Mairal, Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Convex and network flow optimization for structured sparsity. *Journal of Machine Learning Research*, 12(Sep):2681–2720, 2011.

[115] R. Mazumder, P. Radchenko, and A. Dedieu. Subset Selection with Shrinkage: Sparse Linear Modeling when the SNR is low. *ArXiv e-prints*, August 2017.

[116] Rahul Mazumder et al. Discussion of "best subset, forward stepwise or lasso? analysis and recommendations based on extensive comparisons". *Statistical Science*, 35(4):602–608, 2020.

[117] Rahul Mazumder, Jerome H. Friedman, and Trevor Hastie. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011. PMID: 25580042.

[118] Rahul Mazumder and Peter Radchenko. The Discrete Dantzig Selector: Estimating sparse linear models via mixed integer linear optimization. *IEEE Transactions on Information Theory*, 63 (5)(5):3053 – 3075, 2017.

[119] Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 2 edition, 1989.

[120] L Meier, S. van de Geer, and P. Bühlmann. High-dimensional additive modeling. *The Annals of Statistics*, 37:3779–3821, 2009.

[121] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[122] Alan Miller. *Subset selection in regression*. CRC Press Washington, 2002.

[123] Ryuhei Miyashiro and Yuichi Takano. Subset selection by mallows' cp: A mixed integer programming approach. *Expert Systems with Applications*, 42(1):325–331, 2015.

[124] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[125] Marine Le Morvan and Jean-Philippe Vert. Whinter: A working set algorithm for high-dimensional sparse second order interaction models. *arXiv preprint arXiv:1802.05980*, 2018.

[126] Kazuya Nakagawa, Shinya Suzumura, Masayuki Karasuyama, Koji Tsuda, and Ichiro Takeuchi. Safe pattern pruning: An efficient approach for predictive pattern mining. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 1785–1794. ACM, 2016.

[127] Y. Nardi and A. Rinaldo. On the asymptotic properties of the group lasso estimator for linear models. *Electronic Journal of Statistics*, 2:605–633, 2008.

[128] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[129] Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap safe screening rules for sparse-group lasso. In *Advances in Neural Information Processing Systems*, pages 388–396, 2016.

[130] Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap safe screening rules for sparsity enforcing penalties. *The Journal of Machine Learning Research*, 18(1):4671–4703, 2017.

[131] Sahand Negahban, Bin Yu, Martin J Wainwright, and Pradeep K Ravikumar. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*, pages 1348–1356, 2009.

[132] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[133] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[134] G. Obozinski, M. J. Wainwright, and M. I. Jordan. Support and union recovery in high-dimensional multivariate regression. *The Annals of Statistics*, 39:1–47, 2011.

[135] John Tinsley Oden and Junuthula Narasimha Reddy. *An introduction to the mathematical theory of finite elements*. Wiley, New York, 1976.

[136] Art B Owen. A robust hybrid of lasso and ridge regression. *Contemporary Mathematics*, 443(7):59–72, 2007.

[137] FICO White Paper. Introduction to model builder scorecard. https://www.fico.com/en/latest-thinking/white-paper/introduction-model-builder-scorecard.

[138] Neal Parikh and Stephen Boyd. *Proximal Algorithms*. Now Foundations and Trends, 2014.

[139] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[140] A. Patrascu and I. Necoara. Random coordinate descent methods for ℓ_0 regularized convex optimization. *IEEE Transactions on Automatic Control*, 60(7):1811–1824, July 2015.

[141] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[142] Bo Peng, Lan Wang, and Yichao Wu. An error bound for l1-norm support vector machine coefficients in ultra-high dimension. *Journal of Machine Learning Research*, 17:1–26, 2016.

[143] Mert Pilanci, Martin J Wainwright, and Laurent El Ghaoui. Sparse learning via boolean relaxations. *Mathematical Programming*, 151(1):63–87, 2015.

[144] Yaniv Plan and Roman Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *IEEE Transactions on Information Theory*, 59(1):482–494, 2013.

[145] P. Radchenko and G. M. James. Variable selection using adaptive nonlinear interaction structures in high dimensions. *Journal of the American Statistical Association*, 105:1541–1553, 2010.

[146] Peter Radchenko and Gareth M. James. Variable selection using adaptive nonlinear interaction structures in high dimensions. *Journal of the American Statistical Association*, 105(492):1541–1553, 2010.

[147] Garvesh Raskutti, Martin Wainwright, and Bin Yu. Minimax rates of estimation for high-dimensional linear regression over l_q-balls. *IEEE transactions on information theory*, 57(10):6976–6994, 2011.

[148] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Minimax-optimal rates for sparse additive models over kernel classes via convex programming. *Journal of Machine Learning Research*, 13(Feb):389–427, 2012.

[149] P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society, B.*, 71:1009–1030, 2009.

[150] Pradeep Ravikumar, Martin J Wainwright, and John D Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

[151] Philippe Rigollet. 18.s997: High dimensional statistics. *Lecture Notes, Cambridge, MA, USA: MIT OpenCourseWare*, 2015.

[152] Owais Sarwar, Benjamin Sauk, and Nikolaos V. Sahinidis. A Discussion on Practical Considerations with Sparse Regression Methodologies. *Statistical Science*, 35(4):593 – 601, 2020.

[153] Toshiki Sato, Yuichi Takano, Ryuhei Miyashiro, and Akiko Yoshise. Feature subset selection for logistic regression via mixed integer optimization. *Computational Optimization and Applications*, 64(3):865–880, 2016.

[154] Shai Shalev-Shwartz and Tong Zhang. Proximal stochastic dual coordinate ascent. *arXiv preprint arXiv:1211.2717*, 2012.

[155] Yiyuan She, Zhifeng Wang, and He Jiang. Group regularized estimation under structural hierarchy. *Journal of the American Statistical Association*, 113(521):445–454, 2018.

[156] IBM ILOG CPLEX Optimization Studio-CPLEX. Users manual-version 12 release 6. *IBM ILOG CPLEX Division: Incline Village, NV, USA*, 2013.

[157] T. Suzuki and M. Sugiyama. Fast learning rate of multiple kernel learning: Trade-off between sparsity and smoothness. *Annals of Statistics*, 41:1381–1405, 2013.

[158] Zhiqiang Tan and Cun-Hui Zhang. Doubly penalized estimation in additive regression with high-dimensional data. *The Annals of Statistics*, 47(5):2567–2600, 2019.

[159] Bernadetta Tarigan and Sara A Van De Geer. Classifiers of support vector machine type with l1 complexity regularization. *Bernoulli*, 12(6):1045–1076, 2006.

[160] Mohit Tawarmalani and Nikolaos V Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical programming*, 103(2):225–249, 2005.

[161] Gian-Andrea Thanei, Nicolai Meinshausen, and Rajen D Shah. The xyz algorithm for fast interaction search in high-dimensional data. *The Journal of Machine Learning Research*, 19(1):1343–1384, 2018.

[162] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[163] Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):245–266, 2012.

[164] Ryan J Tibshirani. The lasso problem and uniqueness. *Electronic Journal of Statistics*, 7:1456–1490, 2013.

[165] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[166] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016.

[167] Sara Van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, 2000.

[168] Sara A Van de Geer. High-dimensional generalized linear models and the lasso. *The Annals of Statistics*, pages 614–645, 2008.

[169] Juan Pablo Vielma, Shabbir Ahmed, and George L Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs. *INFORMS Journal on Computing*, 20(3):438–450, 2008.

[170] Juan Pablo Vielma, Iain Dunning, Joey Huchette, and Miles Lubin. Extended formulations in mixed integer conic quadratic programming. *Mathematical Programming Computation*, 9(3):369–418, 2017.

[171] G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, 1990.

[172] Martin J Wainwright. Information-theoretic limits on sparsity recovery in the high-dimensional and noisy setting. *IEEE Transactions on Information Theory*, 55(12):5728–5741, 2009.

[173] Jie Wang and Jieping Ye. Two-layer feature reduction for sparse-group lasso via decomposition of convex sets. In *Advances in Neural Information Processing Systems*, pages 2132–2140, 2014.

[174] Jie Wang, Jiayu Zhou, Peter Wonka, and Jieping Ye. Lasso screening rules via dual polytope projection. In *Advances in Neural Information Processing Systems*, pages 1070–1078, 2013.

[175] Rui Wang, Naihua Xiu, and Shenglong Zhou. Fast newton method for sparse logistic regression. *arXiv preprint arXiv:1901.02768*, 2019.

[176] F. Wei and J. Huang. Consistent group selection in high-dimensional linear regression. *Bernoulli*, 16:1369–1384, 2010.

[177] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

[178] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[179] Jing Wu, Bernie Devlin, Steven Ringquist, Massimo Trucco, and Kathryn Roeder. Screen and clean: a tool for identifying interactions in genome-wide association studies. *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society*, 34(3):275–285, 2010.

[180] Weijun Xie and Xinwei Deng. Scalable algorithms for the sparse ridge regression. *SIAM Journal on Optimization*, 30(4):3359–3386, 2020.

[181] Xiaohan Yan and Jacob Bien. Hierarchical sparse modeling: A choice of two group lasso formulations. *Statistical Science*, 32(4):531–560, 2017.

[182] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

[183] Ming Yuan and Ding-Xuan Zhou. Minimax optimal rates of estimation in high dimensional additive models. *The Annals of Statistics*, 44(6):2564–2593, 2016.

[184] X. Yuan and Q. Liu. Newton-type greedy selection methods for ℓ_0 -constrained minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2437–2450, 2017.

[185] Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, February 2010.

[186] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594, 2008.

[187] Cun-Hui Zhang and Tong Zhang. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

[188] Xiang Zhang, Yichao Wu, Lan Wang, and Runze Li. Variable selection for support vector machines in moderately high dimensions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(1):53–76, 2016.

[189] Yuchen Zhang, Martin J Wainwright, and Michael I Jordan. Lower bounds on the performance of polynomial-time algorithms for sparse linear regression. In *Conference on Learning Theory*, pages 921–948. PMLR, 2014.

[190] Yuchen Zhang, Martin J Wainwright, and Michael I Jordan. Optimal prediction for sparse linear models? Lower bounds for coordinate-separable M-estimators. *Electronic Journal of Statistics*, 11(1):752–799, 2017.

[191] Peng Zhao and Bin Yu. On model selection consistency of lasso. *Journal of Machine learning research*, 7(Nov):2541–2563, 2006.

[192] Shenglong Zhou, Naihua Xiu, and Hou-Duo Qi. Global and quadratic convergence of newton hard-thresholding pursuit. *Journal of Machine Learning Research*, 22(12):1–45, 2021.

[193] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320, 2005.