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

Hussein Hazimeh*, Rahul Mazumder†, and Ali Saab‡

Massachusetts Institute of Technology

April, 2021

Abstract

We consider the least squares regression problem, penalized with a combination of the ℓ_0 and squared ℓ_2 penalty functions (a.k.a. $\ell_0\ell_2$ regularization). Recent work shows that the resulting estimators are of key importance in many high-dimensional statistical settings. However, exact computation of these estimators remains a major challenge. Indeed, modern exact methods, based on mixed integer programming (MIP), face difficulties when the number of features $p \sim 10^4$. In this work, we present a new exact MIP framework for $\ell_0\ell_2$-regularized regression that can scale to $p \sim 10^7$, achieving speedups of at least 5000x, compared to state-of-the-art exact methods. Unlike recent work, which relies on modern commercial MIP solvers, we design a specialized nonlinear branch-and-bound (BnB) framework, by critically exploiting the problem structure. A key distinguishing component in our framework lies in efficiently solving the node relaxations using a specialized first-order method, based on coordinate descent (CD). Our CD-based method effectively leverages information across the BnB nodes, through using warm starts, active sets, and gradient screening. In addition, we design a novel method for obtaining dual bounds from primal CD solutions, which certifiably works in high dimensions. Experiments on synthetic and real high-dimensional datasets demonstrate that our framework is not only significantly faster than the state of the art, but can also deliver certifiably optimal solutions to statistically challenging instances that cannot be handled with existing methods. We open source the implementation through our toolkit L0BnB.

*MIT Operations Research Center. Email: hazimeh@mit.edu

†MIT Sloan School of Management, Operations Research Center and MIT Center for Statistics. Email: rahulmaz@mit.edu

‡AJO. Email: saab@mit.edu

1 Introduction

We consider the sparse linear regression problem with additional ℓ_2 regularization [13, 32]. 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:

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

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 [31, 39, 32] for further discussions on this matter. Statistical properties of ℓ_0-based estimators have been extensively studied in the statistics literature [26, 46, 54, 19, 52, 39]. Specifically, under suitable assumptions on the underlying data and appropriate choices of tuning parameters, global solutions of (1) have optimal support recovery properties [52, 23]; and optimal prediction error bounds that do not depend upon X [46]. Appealing prediction error bounds available from optimal solutions of (1) for the low-signal regime are discussed in [39]. Such strong guarantees are generally not satisfied by heuristic solutions to (1)—see [54] for theoretical support and [32] 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 (1) is labeled as NP-Hard [42] and poses computational challenges. Recently, there has been exciting work in developing Mixed Integer Programming (MIP)-based approaches to solve (1), e.g., [17, 10, 41, 21, 13, 32, 12, 53, 6]. Specifically, [10] 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 [13]; and the method in [20] for the classification variant of (1). The approaches of [13, 20] 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 [31, 32]. 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` [25], and local optimization methods for (1), such as `L0Learn` [32], can handle much larger instances, and they typically terminate in the order of milliseconds to seconds.

Our goal in this paper is to advance the computational methodology for the global optimization of Problem (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 (1), which rely on commercial MIP solvers, e.g., Gurobi and CPLEX. These MIP solvers are also based on a BnB

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

framework, but they are equipped with general-purpose relaxation solvers and heuristics that do not take into account the specific structure in Problem (1).

Our BnB solves a mixed integer second order cone program (MISOCP) that is based on a perspective reformulation [24, 2, 27] of Problem (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 [8]. 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 [25, 38, 32]. 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 (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 2).
- To solve the MISOCP, we design a specialized nonlinear BnB, with the following main contributions (see Section 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

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

statistical instances which were virtually unsolvable before (See Section 4). We open source the implementation through our toolkit L0BnB:

`https://github.com/alisaab/L0BnB`

Related Work: As mentioned earlier, an impressive line of recent work considers solving Problem (1), or its cardinality-constrained variant, to optimality. [10] used Gurobi on a Big-M formulation, which can handle $n \sim p \sim 10^3$ in the order of minutes to hours. [13] scale the problem even further by applying outer-approximation (using Gurobi) on a boolean reformulation [45] 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 [11]. [53] consider solving a perspective formulation [24] of the problem directly using Gurobi and reported timings that compare well with [13]—the largest problem instances they consider have $p \sim 10^3$. [20] show that a variant of Problem (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., [45, 21, 5, 53], and algorithms that locally optimize an ℓ_0-based objective [14, 7, 32].

There is also a rich literature on solving mixed integer nonlinear programs (MINLPs) using BnB, e.g., see [36, 8]. Our approach is based on the nonlinear BnB framework [18], 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 [36], e.g., they are used in MOSEK [3] and are also one of the supported options in CPLEX [47]. Generally, interior point based nonlinear solvers are not as effective in exploiting warm starts and sparsity as linear programming solvers [8], which led to an alternative approach known as outer-approximation (OA) [22]. 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 [48] and Gurobi [28] apply OA on extended formulations—[48] 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 [50, 37, 51] and the references therein. In this paper, 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 Supplementary Material: 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.

2 MIP Formulations and Relaxations

In this section, we present MIP formulations for Problem (1) and study their corresponding relaxations.

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 (1), say β^*, satisfies: $\|\beta^*\|_\infty \leq M$. This allows for modeling (1) as a mixed integer quadratic program (MIQP) using the following Big-M formulation:

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

where each binary variable z_i controls whether β_i is zero or not via the first constraint in (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., [10, 39, 32, 53]. See [10, 53] 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 [24, 2, 27] 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 (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 (2):

$$\begin{aligned} \mathrm{PR}(M): \quad \min_{\beta,z,s} \quad & \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0 \sum_{i\in[p]} z_i + \lambda_2 \sum_{i\in[p]} s_i \\ \text{s.t.} \quad & \beta_i^2 \leq s_i z_i, \ i \in [p] \\ & -Mz_i \leq \beta_i \leq Mz_i, \ i \in [p] \\ & z_i \in \{0,1\}, s_i \geq 0, \ i \in [p]. \end{aligned} \tag{3}$$

Problem (3) can be expressed as a MISOCP. Similar to (2), formulation (3) is equivalent to Problem (1) (as long as M is suitably chosen). Algorithms for formulation (3) will be the main focus of our paper.

If we set $M = \infty$ in (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 (1)—this holds since $\beta_i^2 \leq s_i z_i$ enforces $z_i = 0 \implies \beta_i = 0$. We note that [21] have studied Problem (3) and focused on the special case of PR(∞). [21] shows that PR(∞) is equivalent to the pure binary formulations considered in [45, 13]. We also note that [53] have considered a similar perspective formulation for

the cardinality constrained variant of Problem (1). In Proposition 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).

2.2 Relaxation of the Perspective Formulation (3)

In this section, we study the *interval relaxation* of Problem (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 1 shows that the interval relaxation of (3) can be reformulated purely in the β space. This leads to a regularized least squares criterion, where the regularizer involves the reverse Huber [44] 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:

$$\mathcal{B}(t) = \begin{cases} |t| & |t| \leq 1 \\ (t^2 + 1)/2 & |t| \geq 1. \end{cases} \tag{4}$$

Theorem 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) is equivalent to:

$$\min_{\beta \in \mathbb{R}^p} \; F(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i \in [p]} \psi(\beta_i; \lambda_0, \lambda_2, M) \quad s.t. \quad \|\beta\|_\infty \leq M, \tag{5}$$

and we let $V_{PR(M)}$ denote the optimal objective value of (5).

The reduced formulation (5) has an important role in both the subsequent analysis and the algorithmic development in Section 3. Theorem 1 shows that the conic and Big-M constraints in the relaxation of (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 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) 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 1.

Next, we analyze the tightness of the perspective relaxation in (5).

Tightness of the Perspective Relaxation (5): [21] has shown that the interval relaxation of PR(∞) can be written as:

$$V_{\mathrm{PR}(\infty)} = \min_{\beta \in \mathbb{R}^p} G(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i \in [p]} \psi_1(\beta_i; \lambda_0, \lambda_2), \tag{6}$$

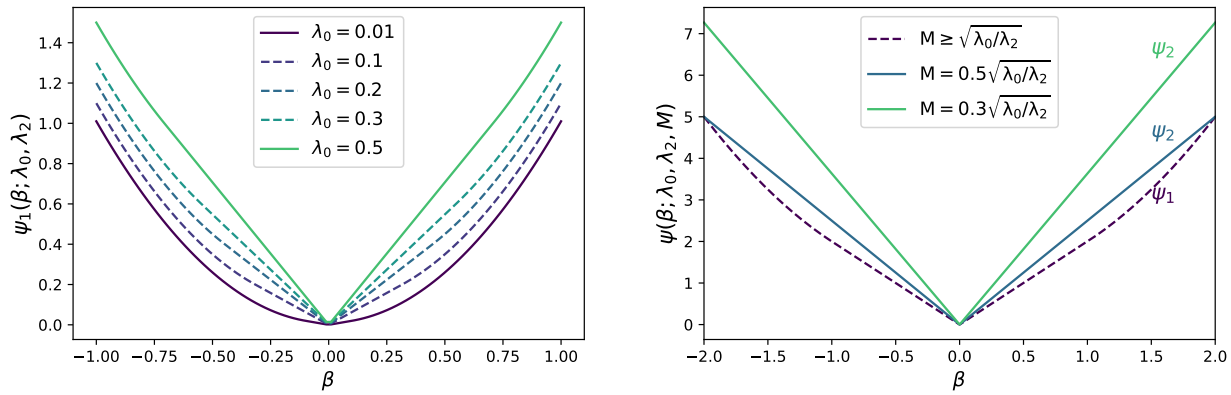

Figure 1: [Left]: Plot of $\psi_1(\beta; \lambda_0, 1)$ for different values of λ_0. [Right]: Plot of $\psi(\beta; 1, 1, M)$ for different values of M.

where $\psi_1(\beta_i; \lambda_0, \lambda_2)$ is defined in Theorem 1. Note that (6) can also be obtained from Theorem 1 (with $M = \infty$). For $\sqrt{\lambda_0/\lambda_2} \leq M$, the relaxation of PR($M$) in Theorem 1 matches that in (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 1.

Proposition 1. *Let $V_{B(M)}$ denote the optimal objective of the interval relaxation of $B(M)$. Let β^* be an optimal solution to* (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 $\sqrt{\lambda_0/\lambda_2} > M$:*

$$V_{PR(M)} \geq V_{B(M)} + \lambda_2(M\|\beta^*\|_1 - \|\beta^*\|_2^2) \tag{7}$$
$$V_{PR(M)} \geq V_{PR(\infty)} + h(\lambda_0, \lambda_2, M)\|\beta^*\|_1. \tag{8}$$

We make a couple of remarks. For bound (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 (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 4.2.1, our experiments empirically validate Proposition 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 1 does not directly compare $V_{\mathrm{PR}(\infty)}$ with $V_{\mathrm{B}(M)}$. In Proposition 2, we establish a new result which compares $V_{\mathrm{PR}(\infty)}$ with $V_{\mathrm{B}(M)}$. Before we present Proposition 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

$$\mathcal{L}(M) := \{\lambda_2 > 0 \mid \exists \beta \in \mathcal{S}(\lambda_2) \text{ s.t. } \|\beta\|_\infty \leq M\}. \tag{9}$$

Proposition 2. *Let $\mathcal{L}(M)$ be as defined in* (9)*. Then, the following holds:*

$$V_{B(M)} \geq V_{PR(\infty)}, \text{ if } M \leq \tfrac{1}{2}\sqrt{\lambda_0/\lambda_2} \tag{10}$$
$$V_{B(M)} \leq V_{PR(\infty)}, \text{ if } M \geq \sqrt{\lambda_0/\lambda_2} \text{ and } \lambda_2 \in \mathcal{L}(M). \tag{11}$$

Proposition 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 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 (1). In contrast, $V_{\mathrm{PR}(\infty)}$ is always a valid relaxation of (1).

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). 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), 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). 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). 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 [8], we rely solely on a primal method which consists of a highly scalable CD-based algorithm. The algorithm solves the node relaxations of (3) in the β-space as opposed to the extended (β, s, z) space—these relaxations are variants of the reduced relaxation introduced in (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 3.1.

[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 3.2.
- **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.

For simplicity of exposition, in the remainder of Section 3, we assume that the columns of X and y have unit ℓ_2 norm.

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 (5). We operate on the reduced formulation compared to the interval relaxation of (3) in the extended (β, s, z) space due to computational reasons. After solving (5), we use the resulting solution, say β^*, to construct a corresponding solution (β^*, s^*, z^*) to the interval relaxation of (3)—see the proof of Theorem 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) to 0 or 1, so their subproblems can be obtained by minor modifications to the root relaxation. For completeness, in Appendix B.2, we discuss how to formulate and solve these node subproblems.

Problem (5) is of the composite form [43]: 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 (5) amenable to cyclic CD [49]. To our knowledge, the use of cyclic CD for Problem (5) is novel. We also emphasize that a direct application of cyclic CD to Problem (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 1.

Algorithm 1: Cyclic CD for Relaxation (5)

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

$$\hat{\beta}_i \leftarrow \underset{\beta_i \in \mathbb{R}}{\arg\min}\, F(\hat{\beta}_1, \ldots, \beta_i, \ldots, \hat{\beta}_p) \quad \text{s.t.} \quad |\beta_i| \leq M. \tag{12}$$

Every limit point of Algorithm 1 is a global minimizer of (5) [49], and the algorithm has a sublinear rate of convergence [34]. We will show that the solution of (12) can be computed in closed-form. To this end, since the columns of X have unit ℓ_2-norm, we note that (12) is equivalent to:

$$\min_{\beta_i \in \mathbb{R}} \quad \frac{1}{2}(\beta_i - \tilde{\beta}_i)^2 + \psi(\beta_i; \lambda_0, \lambda_2, M) \quad \text{s.t.} \quad |\beta_i| \leq M, \tag{13}$$

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 (13) is given by:

$$\hat{\beta}_i = \begin{cases} T(\tilde{\beta}_i; 2\sqrt{\lambda_0\lambda_2}, M) & \text{if } |\tilde{\beta}_i| \leq 2\sqrt{\lambda_0\lambda_2} + \sqrt{\lambda_0/\lambda_2} \\ T(\tilde{\beta}_i(1 + 2\lambda_2)^{-1}; 0, M) & \text{otherwise} \end{cases} \tag{14}$$

and for $\sqrt{\lambda_0/\lambda_2} > M$, the solution of (13) is:

$$\hat{\beta}_i = T(\tilde{\beta}_i; \lambda_0/M + \lambda_2 M, M). \tag{15}$$

Thus, update (12) in Algorithm 1 can be computed in closed-form using expressions (14) and (15). See Appendix 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 [25, 32]. 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 3.1.1.

3.1.1 Active Sets

Update (12) in Algorithm 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 1 (assuming that the regularization parameters are chosen so that the optimal solution of the relaxation is sparse, and good warm starts are used). Motivated by this observation, we run Algorithm 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:

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, F(\beta) \quad \text{s.t.} \quad \|\beta\|_\infty \leq M, \;\; \beta_{\mathcal{A}^c} = 0. \tag{16}$$

After solving (16), we augment the active set with any variable $i \in \text{Supp}(\hat{\beta})^c$ that violates the following optimality condition:

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

[4]Suppose the parent node branches on z_i. After reformulating the node relaxations in the β space (as discussed in Appendix 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.

We repeat this procedure of solving the restricted problem (16) and then augmenting $\mathcal{A}$ with violating variables, until there are no more violations. The algorithm is summarized below.

Algorithm 2: The Active-set Algorithm[5]

- **Input:** Initial solution $\hat{\beta}$ and initial active set $\mathcal{A}$.
- **Repeat**:
  - Step 1: Solve (16) using Algorithm 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, $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{V}$.

The quality of the initial active set affects the number of iterations in Algorithm 2. For example, if $\mathcal{A}$ is a superset of the support of an optimal solution to (5), then $\mathcal{V}$ in the first iteration of Algorithm 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 3.2, we present a novel method that makes use of the updates in Algorithm 2 to obtain provably high-quality dual bounds. We note that our approach goes beyond standard active-set methods [25]. 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 2 can be performed inexactly.

Remark 1. *In practice, we solve the restricted optimization problem in Step 1 of Algorithm 2 inexactly. Specifically, in Step 1, we terminate Algorithm 1 when the relative change in the objective values is below a small numerical tolerance*[6]*. 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 3.2.*

3.1.2 Gradient Screening

Algorithm 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 [33] for a different problem: learning sparse hierarchical interactions via a convex optimization formulation. In the current paper, the optimality checks

[5] Algorithm 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 B.2), and all the variables fixed to zero at the current node should be excluded from the set $\mathcal{V}$ in Step 2.

[6] If upon termination, an integral z is obtained, we solve the relaxation to optimality, as discussed in the introduction of Section 3.

in Step 2 of Algorithm 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.

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

Proposition 3. *Let $\hat{\beta}$ and $\mathcal{V}$ be as defined in Algorithm 2, and define $\hat{r} = y - X\hat{\beta}$. Then, the set $\mathcal{V}$ can be equivalently written as follows:*

$$\mathcal{V} = \{i \in \mathit{Supp}(\hat{\beta})^c \mid |\langle \hat{r}, X_i \rangle| > c(\lambda_0, \lambda_2, M)\}, \tag{17}$$

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, 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 (17) to $\hat{\mathcal{V}}$ instead of $\mathrm{Supp}(\hat{\beta})^c$, i.e., the following holds:

$$\mathcal{V} = \{i \in \hat{\mathcal{V}} \mid |\langle \hat{r}, X_i \rangle| > c(\lambda_0, \lambda_2, M)\}. \tag{18}$$

Assuming $\hat{\mathcal{V}}$ is available, the cost of computing $\mathcal{V}$ in (18) is $\mathcal{O}(n|\hat{\mathcal{V}}|)$. This cost can be significantly smaller than that of (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 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 4. *Let $\hat{\beta}$ and $\mathcal{V}$ be as defined in Algorithm 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:*

$$\mathcal{V} \subseteq \hat{\mathcal{V}} := \Big\{i \in \mathit{Supp}(\hat{\beta})^c \mid |\langle r^0, X_i \rangle| > c(\lambda_0, \lambda_2, M) - \epsilon\Big\}. \tag{19}$$

The set $\hat{\mathcal{V}}$ defined in (19) depends solely on ϵ and the quantities $|\langle r^0, X_i \rangle|$, $i \in \mathrm{Supp}(\hat{\beta})^c$, 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 (17), which requires the costly computation of the terms $|\langle \hat{r}, X_i \rangle|$, $i \in \mathrm{Supp}(\hat{\beta})^c$. Given the sorted values $|\langle r^0, X_i \rangle|$, $i \in \mathrm{Supp}(\hat{\beta})^c$, we can identify $\hat{\mathcal{V}}$ in (19) with $\mathcal{O}(\log p)$ operations, using a variant of binary search. Thus, the overall cost of computing $\mathcal{V}$ using (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 (19) is strict.

Proposition 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 (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 2 (as detailed below). Let $\epsilon_{\mathrm{gs}} \in (0, 1)$ be a pre-specified parameter for the gradient screening procedure. The procedure, which replaces Step 2 of Algorithm 2, is defined as follows:

Gradient Screening

1. If this is the root node and the first iteration of Algorithm 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 (17); and skip all the steps below.

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

3. Compute $\hat{\mathcal{V}}$ using (19) and then $\mathcal{V}$ using (18)[7].

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 $\epsilon_{\text{gs}} = 0.05$ 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).

3.2 Dual Bounds

In practice, we use Algorithm 2 to obtain inexact primal solutions for relaxation (5), as discussed in Remark 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 (5) in Theorem 2.

Theorem 2. *For $\sqrt{\lambda_0/\lambda_2} \leq M$, a dual of Problem (5) is given by:*

$$\max_{\alpha \in \mathbb{R}^n, \gamma \in \mathbb{R}^p} \; h_1(\alpha, \gamma) := -\frac{1}{2}\|\alpha\|_2^2 - \alpha^T y - \sum_{i \in [p]} v(\alpha, \gamma_i), \tag{20}$$

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

$$v(\alpha, \gamma_i) := \Big[\frac{(\alpha^T X_i - \gamma_i)^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i|. \tag{21}$$

Otherwise, if $\sqrt{\lambda_0/\lambda_2} > M$, a dual of Problem (5) is given by

$$\begin{aligned} \max_{\rho \in \mathbb{R}^n, \mu \in \mathbb{R}^p} \; & h_2(\rho, \mu) := -\frac{1}{2}\|\rho\|_2^2 - \rho^T y - M\|\mu\|_1 \\ s.t. \quad & |\rho^T X_i| - \mu_i \leq \lambda_0/M + \lambda_2 M, \; i \in [p]. \end{aligned} \tag{22}$$

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

Let β^* be an optimal solution to Problem (5) *and define $r^* = y - X\beta^*$. The optimal dual variables for* (20) *are given by:*

$$\alpha^* = -r^* \quad \textit{and} \quad \gamma_i^* = \mathbb{1}_{[|\beta_i^*|=M]}(\alpha^{*T}X_i - 2M\lambda_2 \operatorname{sign}(\alpha^{*T}X_i)), i \in [p]. \tag{23}$$

Moreover, the optimal dual variables for (22) *are:*

$$\rho^* = -r^* \quad \textit{and} \quad \mu_i^* = \mathbb{1}_{[|\beta_i^*|=M]}(|\rho^{*T}X_i| - \lambda_0/M - \lambda_2 M), i \in [p]. \tag{24}$$

Note that strong duality holds for relaxation (5) since it satisfies Slater's condition [9], so the optimal objective of the dual in Theorem 2 matches that of (5).

Dual Feasible Solutions: Let $\hat{\beta}$ be an inexact primal solution obtained using Algorithm 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 2.

Case of $\sqrt{\lambda_0/\lambda_2} \leq M$: Here, we construct a dual solution $(\hat{\alpha}, \hat{\gamma})$ for Problem (20) as follows:

$$\hat{\alpha} = -\hat{r} \quad \text{and} \quad \hat{\gamma} \in \operatorname*{arg\,max}_{\gamma \in \mathbb{R}^p} h_1(\hat{\alpha}, \gamma). \tag{25}$$

The choice $\hat{\alpha} = -\hat{r}$ is motivated by the optimality conditions in Theorem 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 (20) is unconstrained. Since (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:

$$\hat{\gamma}_i = T(\hat{\alpha}^T X_i; 2M\lambda_2, \infty). \tag{26}$$

Thus, $\hat{\gamma}$ can be obtained in closed-form using (26). Since $\hat{\beta}$ is the output of Algorithm 2, we know that the corresponding set $\mathcal{V}$ in Algorithm 2 is empty. Thus, by Proposition 3, we have $|\hat{r}^T X_i| \leq 2\sqrt{\lambda_0\lambda_2}$ for any $i \in \operatorname{Supp}(\hat{\beta})^c$. 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 (26), we get that $\hat{\gamma}_i = 0$ for any $i \in \operatorname{Supp}(\hat{\beta})^c$. However, for $i \in \operatorname{Supp}(\hat{\beta})$, $\hat{\gamma}_i$ can be potentially nonzero.

Case of $\sqrt{\lambda_0/\lambda_2} > M$: We construct a dual feasible solution $(\hat{\rho}, \hat{\mu})$ for (22) as follows:

$$\hat{\rho} = -\hat{r} \quad \text{and} \quad \hat{\mu} \in \operatorname*{arg\,max}_{\mu \in \mathbb{R}^p} \; h_2(\hat{\rho}, \mu) \quad \text{s.t.} \quad (\hat{\rho}, \mu) \text{ is feasible for (22).} \tag{27}$$

Similar to (27), the choice $\hat{\rho} = -\hat{r}$ is motivated by the optimality conditions in Theorem 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:

$$\hat{\mu}_i = \Big[|\hat{\rho}^T X_i| - \lambda_0/M - \lambda_2 M\Big]_+. \tag{28}$$

Note that for any $i \in \operatorname{Supp}(\hat{\beta})^c$, we have $|\hat{\rho}^T X_i| = |\hat{r}^T X_i| \leq \lambda_0/M + \lambda_2 M$ (by Proposition 3), which implies that $\hat{\mu}_i = 0$. However, for $i \in \operatorname{Supp}(\hat{\beta})$, $\hat{\mu}_i$ can be potentially nonzero.

Quality of the Dual Bounds: In Theorem 3, we quantify the tightness of the dual bounds obtained from the dual feasible solutions (25) and (27).

Theorem 3. *Let α^*, γ^*, ρ^*, and μ^* be the optimal dual variables defined in Theorem 2. Let β^* be an optimal solution to (5), and $\hat{\beta}$ be an inexact solution obtained using Algorithm 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 (5). For a fixed (M, λ_2), the following holds for the dual solution $(\hat{\alpha}, \hat{\gamma})$ defined in (25):*

$$h_1(\hat{\alpha}, \hat{\gamma}) \geq h_1(\alpha^*, \gamma^*) - k\mathcal{O}(\epsilon) - k\mathcal{O}(\epsilon^2), \tag{29}$$

and for the dual solution $(\hat{\rho}, \hat{\mu})$ defined in (27), we have:

$$h_2(\hat{\rho}, \hat{\mu}) \geq h_2(\rho^*, \mu^*) - k\mathcal{O}(\epsilon) - \mathcal{O}(\epsilon^2). \tag{30}$$

Interestingly, the bounds established in Theorem 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 2, which performs optimality checks on the coordinates outside the support. If vanilla CD was used instead of Algorithm 2, then the term k appearing in the bounds (29) and (30) will be replaced by p, making the bounds loose[8].

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 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 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:

$$h_1(\hat{\alpha}, \hat{\gamma}) = -\frac{1}{2}\|\hat{\alpha}\|_2^2 - \hat{\alpha}^T y - \sum_{i \in \text{Supp}(\hat{\beta})} v(\hat{\alpha}, \hat{\gamma}_i). \tag{31}$$

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 (27) is dual feasible, which means that the constraints in (22) need not be checked when computing the bound. Moreover, $\hat{\mu}_i = 0$ for every $i \in \text{Supp}(\hat{\beta})^c$ (see the discussion after (28)). Thus, the dual bound can be expressed as follows:

$$h_2(\hat{\rho}, \hat{\mu}) = -\frac{1}{2}\|\hat{\rho}\|_2^2 - \hat{\rho}^T y - M \sum_{i \in \text{Supp}(\hat{\beta})} |\hat{\mu}_i|. \tag{32}$$

The expressions in (31) and (32) can be computed in $\mathcal{O}(n + n\|\hat{\beta}\|_0)$ operations.

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 [8, 15, 4, 1]. Among these strategies, strong branching has proven to be very effective in minimizing the size of the search tree [4, 15, 8]. 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 (5). To

[8]This holds by using the argument in the proof of Theorem 3 for Algorithm 1 instead of Algorithm 2.

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` [32], which uses CD and efficient local search algorithms to obtain good quality feasible solutions for Problem (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 [29].

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 [32] for empirical studies on statistical properties.

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 [32, 40] 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 $\tilde{\beta}^\dagger$:

$$\beta(\lambda_2) \in \underset{\beta\in\mathbb{R}^p}{\arg\min} \frac{1}{2}\|\tilde{y} - \tilde{X}\beta\|_2^2 + \lambda_2\|\beta\|_2^2 \quad \text{s.t.} \quad \beta_{(S^\dagger)^c} = 0, \tag{33}$$

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

$$\lambda_2^* \in \underset{\lambda_2\geq 0}{\arg\min} \|\tilde{\beta}^\dagger - \beta(\lambda_2)\|_2. \tag{34}$$

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 (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). 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`[9] [32]. 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 4.2.3, presents an experiment where λ_0 is varied over a range that is independent of `L0Learn`.
We obtain warm starts from `L0Learn`[10] and use them for all the MIP solvers considered.

Solvers and Settings: Our solver, L0BnB[11], is written in Python with critical code sections optimized using Numba [35]. We compare L0BnB with Gurobi (GRB), MOSEK (MSK), and BARON (B) on formulation (3). We also compare with [13] who solve the cardinality-constrained variant of (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 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 3.1.2, 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 4.2.1 (Table 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 C). Moreover, for all approaches, we report the number of BnB nodes explored in Appendix C.

[9] `L0Learn` computes a data-dependent grid of λ_0 values. When CD is used to optimize (1), each λ_0 in the grid leads to a different support size. See Section 4.1 of [32] for more details.

[10] 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.

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

4.2 Comparison with State-of-the-art solvers

4.2.1 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[12] 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 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 $M = 1.5M^*(\lambda_2)$ (where λ_2^* and M^* are defined in Section 4.1). In Table 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 1 are without gradient screening; in Table 6 in Appendix C.1, we report the timings with gradient screening enabled. Moreover, in Appendix C.1, we report the values of λ_0^* and λ_2^*, along with the number of BnB nodes explored.

Table 1: (Sensitivity to λ_2 and M) Running time in seconds for solving (1) (via formulation (3)) by our proposal (L0BnB), Gurobi (GRB), MOSEK (MSK), and BARON (B). The method [13] solves the cardinality-constrained variant of (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	[13]	L0BnB	GRB	MSK	B	[13]	L0BnB	GRB	MSK	B	[13]
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 1 indicate significant speed-ups, reaching over 200,000x compared to Gurobi, 28,000x compared to MOSEK, and 20,000x compared to [13]. 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

[12]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).

optimization problem seems to become easier: L0BnB can be more than 100x faster compared to λ_2^*, and [13] 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[13]. 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 (1) over a range of λ_2 values, which includes values in the neighborhood of λ_2^*. However, the results in Table 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 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 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 1 considers both the two settings in Theorem 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.

4.2.2 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 2. The corresponding values of λ_0^* and λ_2^*, and the number of BnB nodes are reported in Appendix C.2. 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.

4.2.3 Varying λ_0 and λ_2

In the experiment of Section 4.2.1, 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 [32][14], we use $\lambda_0^{\mathrm{m}} = (2 + 4\lambda_2)^{-1}\|X^T y\|_\infty^2$. Next, we describe,

[13] 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.

[14] [32] shows that λ_0^{m} is the smallest choice of λ_0 for which $\beta = 0$ is a coordinate-wise minimizer for Problem (1). In this experiment, we verified numerically that $\beta = 0$ is a global minimizer at λ_0^{m}.

Table 2: Running time (seconds) for solving (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	-

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 (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) with $M = \infty$. We then compute an 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) for $M(\lambda_0, \lambda_2) = 1.5\tilde{M}(\lambda_0, \lambda_2)$. In Table 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 C.3.

In Table 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) 13 out of the 20 instances in the 1 hour limit. In contrast, Gurobi could not solve any of the instances and MOSEK could only solve 6.

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 C.4.

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[15]. The timings for $n \in \{10^2, 10^3, 10^4, 10^5\}$ are in Table 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

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

Table 3: Running time for solving (3) over a grid of λ_0 and λ_2 values. "NNZ" is the number of nonzeros in the solution to (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%)	-

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 $p \in \{10^4, 10^5\}$, $n = 10^3$, $k^\dagger = 10$, 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 [55]. 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 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 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 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%)

4.4 Real Data and Ablation Studies (Algorithm Settings)

High-dimensional Real Data: Here we investigate the run time of L0BnB on the Riboflavin dataset [16]—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 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 4.2.1). The run times for different choices of p are reported in Table 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.

Table 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%)

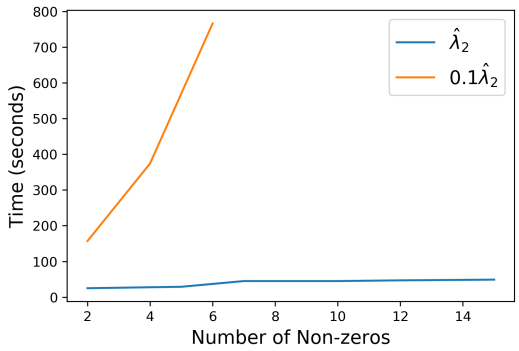

Figure 2: L0BnB run time on a real genomics dataset (Riboflavin) with $n = 71$ and $p \approx 8.3 \times 10^6$.

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 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. [6] 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.

Acknowledgements

Hussein Hazimeh acknowledges research support from the Office of Naval Research ONR-N000141812298. Rahul Mazumder acknowledges research funding from the Office of Naval Research ONR-N000141812298 (Young Investigator Award), the National Science Foundation (NSF-IIS-1718258) and IBM. The authors would like to thank the referees for their constructive comments, which led to an improvement in the paper.

References

[1] Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Operations Research Letters **33**(1), 42–54 (2005)

[2] Aktürk, M.S., Atamtürk, A., Gürel, S.: A strong conic quadratic reformulation for machine-job assignment with controllable processing times. Operations Research Letters **37**(3), 187–191 (2009)

[3] Andersen, E.D., Roos, C., Terlaky, T.: On implementing a primal-dual interior-point method for conic quadratic optimization. Mathematical Programming **95**(2), 249–277 (2003)

[4] Applegate, D., Bixby, R., Cook, W., Chvátal, V.: On the solution of traveling salesman problems (1998)

[5] Atamturk, A., Gomez, A.: Rank-one convexification for sparse regression. arXiv preprint arXiv:1901.10334 (2019)

[6] Atamturk, A., Gomez, A.: Safe screening rules for l0-regression from perspective relaxations. In: H.D. III, A. Singh (eds.) Proceedings of the 37th International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 119, pp. 421–430. PMLR (2020). URL `http://proceedings.mlr.press/v119/atamturk20a.html`

[7] Beck, A., Eldar, Y.C.: Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. SIAM Journal on Optimization **23**(3), 1480–1509 (2013). DOI 10.1137/120869778. URL `https://doi.org/10.1137/120869778`

[8] Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A.: Mixed-integer nonlinear optimization. Acta Numerica **22**, 1–131 (2013)

[9] Bertsekas, D.: Nonlinear Programming. Athena scientific optimization and computation series. Athena Scientific (2016). URL `https://books.google.com/books?id=TwOujgEACAAJ`

[10] Bertsimas, D., King, A., Mazumder, R.: Best subset selection via a modern optimization lens. The Annals of Statistics **44**(2), 813–852 (2016)

[11] Bertsimas, D., Pauphilet, J., Van Parys, B.: Sparse classification: a scalable discrete optimization perspective. arXiv preprint arXiv:1710.01352 (2017)

[12] Bertsimas, D., Pauphilet, J., Van Parys, B.: Sparse regression: Scalable algorithms and empirical performance. arXiv preprint arXiv:1902.06547 (2019)

[13] Bertsimas, D., Van Parys, B., et al.: Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. The Annals of Statistics **48**(1), 300–323 (2020)

[14] Blumensath, T., Davies, M.E.: Iterative hard thresholding for compressed sensing. Applied and computational harmonic analysis **27**(3), 265–274 (2009)

[15] Bonami, P., Lee, J., Leyffer, S., Wächter, A.: More branch-and-bound experiments in convex nonlinear integer programming. Preprint ANL/MCS-P1949-0911, Argonne National Laboratory, Mathematics and Computer Science Division (2011)

[16] Bühlmann, P., Kalisch, M., Meier, L.: High-dimensional statistics with a view toward applications in biology (2014)

[17] Cozad, A., Sahinidis, N.V., Miller, D.C.: Learning surrogate models for simulation-based optimization. AIChE Journal **60**(6), 2211–2227 (2014). DOI https://doi.org/10.1002/aic.14418. URL `https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.14418`

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

[19] David, G., Ilias, Z.: High dimensional regression with binary coefficients. estimating squared error and a phase transtition. In: Conference on Learning Theory, pp. 948–953 (2017)

[20] Dedieu, A., Hazimeh, H., Mazumder, R.: Learning sparse classifiers: Continuous and mixed integer optimization perspectives. arXiv preprint arXiv:2001.06471 (2020)

[21] Dong, H., Chen, K., Linderoth, J.: Regularization vs. Relaxation: A conic optimization perspective of statistical variable selection. ArXiv e-prints (2015)

[22] Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical programming **36**(3), 307–339 (1986)

[23] Fletcher, A.K., Rangan, S., Goyal, V.K.: Necessary and sufficient conditions for sparsity pattern recovery. IEEE Transactions on Information Theory **55**(12), 5758–5772 (2009)

[24] Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0–1 mixed integer programs. Mathematical Programming **106**(2), 225–236 (2006)

[25] Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software **33**(1), 1–22 (2010). URL `http://www.jstatsoft.org/v33/i01/`

[26] Greenshtein, E., 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)

[27] Günlük, O., Linderoth, J.: Perspective reformulations of mixed integer nonlinear programs with indicator variables. Mathematical programming **124**(1-2), 183–205 (2010)

[28] Gurobi Optimization, I.: Gurobi optimizer reference manual. URL http://www. gurobi. com (2020)

[29] Hammarling, S., Lucas, C.: Updating the qr factorization and the least squares problem. Manchester Institute for Mathematical Sciences, University of Manchester (2008)

[30] Hart, W.E., Watson, J.P., Woodruff, D.L.: Pyomo: modeling and solving mathematical programs in python. Mathematical Programming Computation **3**(3), 219–260 (2011)

[31] Hastie, T., Tibshirani, R., Tibshirani, R.J.: Extended comparisons of best subset selection, forward stepwise selection, and the lasso. arXiv preprint arXiv:1707.08692 (2017)

[32] Hazimeh, H., Mazumder, R.: Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. Operations Research **68**(5), 1517–1537 (2020). DOI 10.1287/opre.2019.1919. URL `https://doi.org/10.1287/opre.2019.1919`

[33] Hazimeh, H., Mazumder, R.: Learning hierarchical interactions at scale: A convex optimization approach. In: International Conference on Artificial Intelligence and Statistics, pp. 1833–1843. PMLR (2020)

[34] Hong, M., Wang, X., Razaviyayn, M., Luo, Z.Q.: Iteration complexity analysis of block coordinate descent methods. Mathematical Programming **163**(1-2), 85–114 (2017)

[35] Lam, S.K., Pitrou, A., Seibert, S.: Numba: A llvm-based python jit compiler. In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, pp. 1–6 (2015)

[36] Lee, J., Leyffer, S.: Mixed integer nonlinear programming, vol. 154. Springer Science & Business Media (2011)

[37] Lubin, M., Yamangil, E., Bent, R., Vielma, J.P.: Extended formulations in mixed-integer convex programming. In: International Conference on Integer Programming and Combinatorial Optimization, pp. 102–113. Springer (2016)

[38] Mazumder, R., Friedman, J.H., Hastie, T.: Sparsenet: Coordinate descent with nonconvex penalties. Journal of the American Statistical Association **106**(495), 1125–1138 (2011). DOI 10.1198/jasa.2011.tm09738. URL `https://doi.org/10.1198/jasa.2011.tm09738`. PMID: 25580042

[39] Mazumder, R., Radchenko, P., Dedieu, A.: Subset selection with shrinkage: Sparse linear modeling when the snr is low. arXiv preprint arXiv:1708.03288 (2017)

[40] Mazumder, R., et al.: Discussion of "best subset, forward stepwise or lasso? analysis and recommendations based on extensive comparisons". Statistical Science **35**(4), 602–608 (2020)

[41] Miyashiro, R., Takano, Y.: Subset selection by mallows' cp: A mixed integer programming approach. Expert Systems with Applications **42**(1), 325–331 (2015)

[42] Natarajan, B.K.: Sparse approximate solutions to linear systems. SIAM journal on computing **24**(2), 227–234 (1995)

[43] Nesterov, Y.: Gradient methods for minimizing composite functions. Mathematical Programming **140**(1), 125–161 (2013)

[44] Owen, A.B.: A robust hybrid of lasso and ridge regression. Contemporary Mathematics **443**(7), 59–72 (2007)

[45] Pilanci, M., Wainwright, M.J., El Ghaoui, L.: Sparse learning via boolean relaxations. Mathematical Programming **151**(1), 63–87 (2015)

[46] Raskutti, G., Wainwright, M.J., Yu, B.: Minimax rates of estimation for high-dimensional linear regression over l_q-balls. IEEE transactions on information theory **57**(10), 6976–6994 (2011)

[47] Studio-CPLEX, I.I.C.O.: Users manual-version 12 release 6. IBM ILOG CPLEX Division: Incline Village, NV, USA (2013)

[48] Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical programming **103**(2), 225–249 (2005)

[49] Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. Journal of Optimization Theory and Applications **109**(3), 475–494 (2001). DOI 10.1023/A:1017501703105. URL `http://dx.doi.org/10.1023/A:1017501703105`

[50] Vielma, J.P., Ahmed, S., Nemhauser, G.L.: A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs. INFORMS Journal on Computing **20**(3), 438–450 (2008)

[51] Vielma, J.P., Dunning, I., Huchette, J., Lubin, M.: Extended formulations in mixed integer conic quadratic programming. Mathematical Programming Computation **9**(3), 369–418 (2017)

[52] Wainwright, M.J.: Information-theoretic limits on sparsity recovery in the high-dimensional and noisy setting. IEEE Transactions on Information Theory **55**(12), 5728–5741 (2009)

[53] Xie, W., Deng, X.: Scalable algorithms for the sparse ridge regression. SIAM Journal on Optimization **30**(4), 3359–3386 (2020)

[54] Zhang, Y., Wainwright, M.J., Jordan, M.I.: Lower bounds on the performance of polynomial-time algorithms for sparse linear regression. In: Conference on Learning Theory, pp. 921–948. PMLR (2014)

[55] Zhao, P., Yu, B.: On model selection consistency of lasso. Journal of Machine learning research **7**(Nov), 2541–2563 (2006)

A Proofs of Technical Results

Proof of Theorem 1: The interval relaxation of (3) can be expressed as:

$$\min_{\beta} \quad \Big\{\frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i\in[p]} \min_{s_i,z_i}(\lambda_0 z_i + \lambda_2 s_i)\Big\} \tag{35a}$$

$$\beta_i^2 \le s_i z_i,\ i \in [p] \tag{35b}$$

$$-Mz_i \le \beta_i \le Mz_i,\ i \in [p] \tag{35c}$$

$$z_i \in [0,1], s_i \ge 0,\ i \in [p] \tag{35d}$$

Let $\omega(\beta_i; \lambda_0, \lambda_2, M) := \min_{s_i,z_i}(\lambda_0 z_i + \lambda_2 s_i)$ s.t. (35b), (35c), (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 (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 (35b) and (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 (35b) and (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:

$$\omega(\beta_i; \lambda_0, \lambda_2, M) = \min_{s_i} \max\Big\{\underbrace{\lambda_0\frac{\beta_i^2}{s_i} + \lambda_2 s_i}_{\text{Case I}}, \underbrace{\lambda_0\frac{|\beta_i|}{M} + \lambda_2 s_i}_{\text{Case II}}\Big\} \ \text{ s.t. } \ s_i \ge \beta_i^2,\ |\beta_i| \le M. \tag{36}$$

Suppose that Case I attains the maximum in (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 $\sqrt{\lambda_0/\lambda_2} \le |\beta_i| \le M$. Plugging s_i^* into (36) leads to $\omega(\beta_i; \lambda_0, \lambda_2, M) = 2\lambda_0\mathcal{B}(\beta_i\sqrt{\lambda_2/\lambda_0})$, assuming $\sqrt{\lambda_0/\lambda_2} \le M$.

Now suppose Case II attains the maximum in (36). This holds if $s_i \ge |\beta_i|M$. 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 $|\beta_i|M \ge \beta_i^2$ (since $|\beta_i| \le M$), which implies that the optimal solution is $s_i^* = |\beta_i|M$. Substituting s_i^* into (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 1: First, we show (7). Note that $V_{\mathrm{B}(M)}$ can be simplified to:

$$V_{\mathrm{B}(M)} = \min_{\|\beta\|_\infty \le M} H(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i\in[p]}\Big(\frac{\lambda_0}{M}|\beta_i| + \lambda_2\beta_i^2\Big). \tag{37}$$

Recalling Theorem 1 and the definition of $F(\beta)$ in (5), note that for $\sqrt{\lambda_0/\lambda_2} > M$:

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

where the inequality holds since β^*, an optimal solution to (5), is feasible for (37). This establishes (7).

We now show (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 (4)). Also recall that $V_{\mathrm{PR}(\infty)} = \min_\beta G(\beta)$, where $G(\beta)$ is defined in (6) (this follows from applying Theorem 1 with $M = \infty$). The following then holds:

$$\begin{aligned} V_{\mathrm{PR}(M)} - V_{\mathrm{PR}(\infty)} \geq F(\beta^*) - G(\beta^*) =& \sum\nolimits_{i\in[p]} \Big\{\psi_2(\beta_i^*; \lambda_0, \lambda_2, M) - \psi_1(\beta_i^*; \lambda_0, \lambda_2)\Big\} \\ =& h(\lambda_0, \lambda_2, M)\|\beta^*\|_1, \end{aligned} \tag{38}$$

where the inequality holds since β^*, an optimal solution to (5), is feasible for (6).

Proof of Proposition 2: First, we recall that $V_{\mathrm{PR}(\infty)} = \min_\beta G(\beta)$, where $G(\beta)$ is defined in (6), and that $V_{\mathrm{B}(M)}$ is defined in (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 (10).

Proof of (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:

$$t(\beta_i) = \Big(2\sqrt{\lambda_0\lambda_2} - \tfrac{\lambda_0}{M}\Big)|\beta_i| - \lambda_2\beta_i^2 \leq 0, \tag{39}$$

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 (37). Then, the following holds:

$$V_{\mathrm{B}(M)} - V_{\mathrm{PR}(\infty)} \geq H(\beta^\dagger) - G(\beta^\dagger) = -\sum\nolimits_{i\in[p]} t(\beta_i^\dagger) \geq 0, \tag{40}$$

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 (40) follows from (39). This establishes (10).

To show (11), we will need the following lemma.

Lemma 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 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 1. □

Proof of (11)**:** Define $v^*(M) = \min_{\|\beta\|_\infty \leq M} G(\beta)$ (recall, $G(\beta)$ is defined in (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:

$$v^*(M) - V_{\mathrm{B}(M)} \geq G(\beta^*) - H(\beta^*) = \sum_{i\in[p]} t(\beta_i^*) \geq 0 \tag{41}$$

where the first inequality holds since β^* is a feasible solution to (37) so $V_{\mathrm{B}(M)} \leq H(\beta^*)$, and the last inequality is due to Lemma 1.

Now suppose that $\lambda_2 \in \mathcal{L}(M)$ (defined in (9)) and let $\hat{\beta} \in \mathcal{S}(\lambda_2)$ (i.e., $\hat{\beta}$ is optimal for (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 (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 (11) (since by definition $G(\hat{\beta}) = V_{\mathrm{PR}(\infty)}$); and completes the proof of Proposition 2.

Proof of Proposition 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 (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 $\sqrt{\lambda_0/\lambda_2} \leq M$, (14) implies that the solution of (13) is nonzero iff $|\tilde{\beta}_i| > 2\sqrt{\lambda_0 \lambda_2}$. Similarly, for $\sqrt{\lambda_0/\lambda_2} > M$, by (15), the solution of (13) is nonzero iff $|\tilde{\beta}_i| > \lambda_0/M + \lambda_2 M$. Using these observations in the definition of $\mathcal{V}$ in Algorithm 2, leads to the result of the proposition.

Proof of Proposition 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 (19).

Proof of Theorem 2: Problem (5) can be written equivalently as:

$$\min_{\beta \in \mathbb{R}^p, r \in \mathbb{R}^n} \quad \frac{1}{2}\|r\|_2^2 + \sum_{i \in [p]} \psi(\beta_i; \lambda_0, \lambda_2, M) \;\; \text{s.t.} \;\; r = y - X\beta, \;\; |\beta_i| \leq M, \; \forall i \in [p]. \tag{42}$$

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 (42), leading to the following Lagrangian dual:

$$\max_{\alpha \in \mathbb{R}^n, \eta \in \mathbb{R}^p_{\geq 0}} \quad \min_{\beta \in \mathbb{R}^p, r \in \mathbb{R}^n} \quad L(\beta, r, \alpha, \eta) \tag{43}$$

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 (43). Let us write:

$$\min_{\beta, r} \quad L(\beta, r, \alpha, \eta) = \min_r \left\{ \frac{1}{2}\|r\|_2^2 + \alpha^T r \right\} + \sum_{i \in [p]} \min_{\beta_i} \left\{ D_i(\beta_i) \right\} - \alpha^T y - \sum_{i \in [p]} M\eta_i \tag{44}$$

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

$$\tilde{\beta}_i^* = \arg\min_{\beta_i} \; D_i(\beta_i) = \begin{cases} 0 & \text{if} \quad 2\sqrt{\lambda_0\lambda_2} + \eta_i - |\alpha^T X_i| \geq 0 \\ -\sqrt{\lambda_0/\lambda_2}\,\text{sign}(\alpha^T X_i) & \text{otherwise} \end{cases} \tag{45}$$

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

$$\tilde{\beta}_i^* = \arg\min_{\beta_i} D_i(\beta_i) = \arg\min_{\beta_i} \lambda_2\beta_i^2 + \alpha^T X_i\beta_i + \eta_i|\beta_i| = T(-\alpha^T X_i; \eta_i, \infty)/(2\lambda_2) \tag{46}$$

where, T above is the soft-thresholding operator [25]. But $\tilde{\beta}_i^*$ in (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:

$$D_i(\tilde{\beta}_i^*) = -\frac{1}{4\lambda_2}\Big(|\alpha^T X_i| - \eta_i\Big)^2 + \lambda_0 \quad \text{if} \quad |\alpha^T X_i| - \eta_i \geq 2\sqrt{\lambda_0\lambda_2}. \tag{47}$$

Combining (45) and (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 (43), we get the following dual problem:

$$\max_{\alpha \in \mathbb{R}^n, \eta \in \mathbb{R}^p_{\geq 0}} \; -\frac{1}{2}\|\alpha\|_2^2 - \alpha^T y - \sum_{i=1}^p \Big[\frac{(|\alpha^T X_i| - \eta_i)^2}{4\lambda_2} - \lambda_0\Big]_+ - \sum_{i=1}^p M\eta_i. \tag{48}$$

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 (48) can be reformulated as:

$$\max_{\alpha \in \mathbb{R}^n, \gamma \in \mathbb{R}^p} \; -\frac{1}{2}\|\alpha\|_2^2 - \alpha^T y - \sum_{i=1}^p \Big[\frac{(\alpha^T X_i - \gamma_i)^2}{4\lambda_2} - \lambda_0\Big]_+ - \sum_{i=1}^p M|\gamma_i| \tag{49}$$

which is the formulation in (20).

We now show (23). Let (α^*, η^*) be an optimal solution of (48). First, it is easy to see $r^* = \arg\min_r L(\beta^*, r, \alpha^*, \eta^*) = -\alpha^*$. For the second part of (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 (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 (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.

Lemma 2. *Suppose $\sqrt{\lambda_0/\lambda_2} \leq M$. Let $\hat{\beta}$ be a solution from Algorithm 2 and $(\hat{\alpha}, \hat{\gamma})$ be the corresponding dual solution defined in* (25)*. Let β^* and r^* be as defined in Theorem 2, and define the primal gap $\epsilon = \|X(\beta^* - \hat{\beta})\|_2$. Then, for every $i \in Supp(\hat{\beta})^c$, we have $v(\hat{\alpha}, \hat{\gamma}_i) = 0$ (see* (21) *for definition of v); and for every $i \in Supp(\hat{\beta})$, we have*

$$v(\hat{\alpha}, \hat{\gamma}_i) \leq c_i\epsilon + (4\lambda_2)^{-1}\epsilon^2 + v(\alpha^*, \gamma_i^*), \tag{50}$$

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

Proof of Lemma 2. Fix some $i \in \text{Supp}(\hat{\beta})^c$. Since $\hat{\beta}$ is the output of Algorithm 2, the set $\mathcal{V}$ in Algorithm 2 must be empty. Thus, using Proposition 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 (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 (25) and the definition of h_1 in (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:

$$\begin{aligned}
v(\hat{\alpha}, \hat{\gamma}_i) \leq v(\hat{\alpha}, \gamma_i^*) &= \Big[\frac{(\hat{\alpha}^T X_i - \gamma_i^*)^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\
&= \Big[\frac{((\hat{\alpha} - \alpha^*)^T X_i + \alpha^{*T} X_i - \gamma_i^*)^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\
&= \Big[\frac{a^2 + 2ab + b^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\
&\leq \frac{a^2 + 2|a||b|}{4\lambda_2} + \Big[\frac{b^2}{4\lambda_2} - \lambda_0\Big]_+ + M|\gamma_i^*| \\
&\leq \frac{a^2 + 2|a||b|}{4\lambda_2} + v(\alpha^*, \gamma_i^*). \qquad (51)
\end{aligned}$$

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 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 (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 (23), $\gamma_i^* = \alpha^{*T} X_i - 2M\lambda_2 \,\text{sign}(\alpha^{*T} X_i)$, which leads to $|b| \leq 2M\lambda_2$.

Plugging $|a| \leq \epsilon$ and the bounds on $|b|$ (above) into (51), we arrive to (50). □

Proof of Theorem 3: We consider two cases based on $\sqrt{\lambda_0/\lambda_2} \leq M$ or $\sqrt{\lambda_0/\lambda_2} > M$.
Showing bound (29)**:** First, we consider the case of $\sqrt{\lambda_0/\lambda_2} \leq M$, i.e., we will establish the bound in (29). Note that the following holds:

$$\begin{aligned}
\frac{1}{2}\|\hat{\alpha}\|_2^2 + \hat{\alpha}^T y &= \frac{1}{2}\|\hat{\alpha} - \alpha^* + \alpha^*\|_2^2 + (\hat{\alpha} - \alpha^* + \alpha^*)^T y \\
&= \frac{1}{2}\|\hat{\alpha} - \alpha^*\|_2^2 + (\hat{\alpha} - \alpha^*)^T \alpha^* + \frac{1}{2}\|\alpha^*\|_2^2 + (\hat{\alpha} - \alpha^*)^T y + \alpha^{*T} y \\
&\leq \frac{1}{2}\epsilon^2 + \epsilon\|\alpha^*\|_2 + \frac{1}{2}\|\alpha^*\|_2^2 + \epsilon\|y\|_2 + \alpha^{*T} y, \qquad (52)
\end{aligned}$$

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 2) in (52), we get:

$$\frac{1}{2}\|\hat{\alpha}\|_2^2 + \hat{\alpha}^T y \leq \frac{1}{2}\epsilon^2 + 2\epsilon + \frac{1}{2}\|\alpha^*\|_2^2 + \alpha^{*T} y. \tag{53}$$

Plugging (53) into the objective function in (20) (with $\alpha = \hat{\alpha}$ and $\gamma = \hat{\gamma}$):

$$h_1(\hat{\alpha}, \hat{\gamma}) \geq -\frac{1}{2}\|\alpha^*\|_2^2 - \alpha^{*T} y - 2\epsilon - \frac{1}{2}\epsilon^2 - \sum_{i \in [p]} v(\hat{\alpha}, \hat{\gamma}_i). \tag{54}$$

By Lemma 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 (50) for every $i \in \text{Supp}(\hat{\beta})$, in (54), we get:

$$\begin{aligned} h_1(\hat{\alpha}, \hat{\gamma}) &\geq -\frac{1}{2}\|\alpha^*\|_2^2 - \alpha^{*T} y - \sum_{i \in [p]} v(\alpha^*, \gamma_i^*) - 2\epsilon - \frac{1}{2}\epsilon^2 - \sum_{i \in \text{Supp}(\hat{\beta})} \Big(c_i \epsilon + (4\lambda_2)^{-1}\epsilon^2\Big) \\ &= h_1(\alpha^*, \gamma^*) - 2\epsilon - \frac{1}{2}\epsilon^2 - \sum_{i \in \text{Supp}(\hat{\beta})} \Big(c_i \epsilon + (4\lambda_2)^{-1}\epsilon^2\Big). \end{aligned} \tag{55}$$

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 2) in (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 (29).

Showing bound (30)**:** Now, we consider the case of $\sqrt{\lambda_0/\lambda_2} > M$, where we will establish the bound in (30). By the same argument used in deriving (54), we have:

$$h_2(\hat{\rho}, \hat{\mu}) \geq -\frac{1}{2}\|\rho^*\|_2^2 - \rho^{*T} y - 2\epsilon - \frac{1}{2}\epsilon^2 - M\|\hat{\mu}\|_1. \tag{56}$$

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

$$\begin{aligned} |\hat{\mu}_i| = \Big[|\hat{\rho}^T X_i| - \lambda_0/M - \lambda_2 M\Big]_+ &\leq \Big[|(\hat{\rho} - \rho^*)^T X_i| + |\rho^{*T} X_i| - \lambda_0/M - \lambda_2 M\Big]_+ \\ &\leq |(\hat{\rho} - \rho^*)^T X_i| + \Big[|\rho^{*T} X_i| - \lambda_0/M - \lambda_2 M\Big]_+ \\ &\leq \epsilon + |\mu_i^*|, \end{aligned} \tag{57}$$

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 2 along with the following:

$$|\rho^{*T} X_i| \leq \lambda_0/M + \lambda_2 M \;\text{ if }\; |\beta_i^*| < M \;\text{ and }\; |\rho^{*T} X_i| \geq \lambda_0/M + \lambda_2 M \;\text{ if }\; |\beta_i^*| = M. \tag{58}$$

Conditions in (58) follow from the optimality of (ρ^*, μ^*) for (22). (For $|\beta_i^*| < M$, (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 (22). When $|\beta_i^*| = M$, we will establish (58) via contradiction: If $|\rho^{*T} X_i| < \lambda_0/M + \lambda_2 M$ holds true, then (24) implies $\mu_i^* < 0$, which would violate optimality for (22).)

For $i \in \text{Supp}(\hat{\beta})^c$, we have $\hat{\mu}_i = 0$ (see the discussion after (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 (57) (for all $i \in \text{Supp}(\hat{\beta})$) in (56), and simplifying, we get:

$$h_2(\hat{\rho}, \hat{\mu}) \geq h_2(\rho^*, \mu^*) - \epsilon(2 + Mk) - \epsilon^2/2, \tag{59}$$

which leads to (30).

B Additional Technical Details

B.1 Derivation of solutions to Problem (13)

First, we consider the setting of $\sqrt{\lambda_0/\lambda_2} \leq M$. From Theorem 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 (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 (13) is given by a capped version (due to the box constraint on β_i) of the soft thresholding operator [25]: $\beta_i^* = T(\tilde{\beta}_i; 2\sqrt{\lambda_0\lambda_2}, M)$; 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 (14).

Finally, we consider the setting of $\sqrt{\lambda_0/\lambda_2} > M$. By Theorem 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 (15).

B.2 Node Subproblems

In Theorem 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:

$$\begin{aligned} \min_{\beta,z,s} \quad & \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0 \sum_{i\in[p]} z_i + \lambda_2 \sum_{i\in[p]} s_i \\ \text{s.t.} \quad & \beta_i^2 \le s_i z_i, \ i \in [p] \\ & -Mz_i \le \beta_i \le Mz_i, \ i \in [p] \\ & s_i \ge 0, \ i \in [p] \\ & z_i = 0, i \in \mathcal{Z}, \quad z_i = 1, i \in \mathcal{N}, \quad z_i \in [0,1], i \in [p] \setminus (\mathcal{Z} \cup \mathcal{N}). \end{aligned} \tag{60}$$

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 1, Problem (60) can be equivalently expressed in the β space as:

$$\begin{aligned} \min_{\beta\in\mathbb{R}^p} \ \tilde{F}(\beta) := \frac{1}{2}\|y - X\beta\|_2^2 + \sum_{i\in\mathcal{N}^c} \psi(\beta_i; \lambda_0, \lambda_2, M) + \sum_{i\in\mathcal{N}} \tilde{\psi}(\beta_i; \lambda_0, \lambda_2) \\ \text{s.t.} \ \ \|\beta\|_\infty \le M, \quad \beta_i = 0, i \in \mathcal{Z}. \end{aligned} \tag{61}$$

Note that Problem (61) is similar to Problem (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 (61) using cyclic CD, the coordinates in $(\mathcal{N} \cup \mathcal{Z})^c$ are updated exactly as described in Section 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 (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: $T(\tilde{\beta}_i(1 + 2\lambda_2)^{-1}; 0, M)$.

C Additional Experimental Results and Details

C.1 Experiment of Section 4.2.1

In Table 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 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.

C.2 Experiment of Section 4.2.2

The parameters λ_0 and λ_2, and the number of BnB nodes explored are reported in Table 8.

C.3 Experiment of Section 4.2.3

The parameter $\lambda_2^* = 0.0409$. The parameter λ_0 and the number of BnB nodes explored are reported in Table 9.

Table 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 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**

Table 7: Number of BnB nodes explored in the experiment of Section 4.2.1. For [13], 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	[13]	L0BnB	GRB	MSK	B	[13]	L0BnB	GRB	MSK	B	[13]
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 8: The parameters λ_0 and λ_2, and the number of nodes explored by the different solvers in the experiment of Section 4.2.2. 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

C.4 Experiment of Section 4.3

The number of nodes for all experiments of Section 4.3, and the parameters for varying n and k are presented in Table 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 $\lambda_0^* = 0.0304$.

C.5 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 [30], and [13]'s OA approach using the SubsetSelectionCIO toolkit [11] 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 9: The parameter λ_0 and the number of BnB nodes explored in the experiment of Section 4.2.3.

$p = 10^3$					
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 10: Problem parameters and number of nodes for the experiment of Section 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