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

Hussein Hazimeh * and Rahul Mazumder †

Massachusetts Institute of Technology

March, 2018

Abstract

We consider the canonical L_0-regularized least squares problem (aka best subsets) which is generally perceived as a 'gold-standard' for many sparse learning regimes. In spite of worst-case computational intractability results, recent work has shown that advances in mixed integer optimization can be used to obtain near-optimal solutions to this problem for instances where the number of features $p \approx 10^3$. While these methods lead to estimators with excellent statistical properties, often there is a price to pay in terms of a steep increase in computation times, especially when compared to highly efficient popular algorithms for sparse learning (e.g., based on L_1-regularization) that scale to much larger problem sizes. Bridging this gap is a main goal of this paper. We study the computational aspects of a family of L_0-regularized least squares problems with additional convex penalties. We propose a hierarchy of necessary optimality conditions for these problems. We develop new algorithms, based on coordinate descent and local combinatorial optimization schemes, and study their convergence properties. We demonstrate that the choice of an algorithm determines the quality of solutions obtained; and local combinatorial optimization-based algorithms generally result in solutions of superior quality. We show empirically that our proposed framework is relatively fast for problem instances with $p \approx 10^6$ and works well, in terms of both optimization and statistical properties (e.g., prediction, estimation, and variable selection), compared to simpler heuristic algorithms. A version of our algorithm reaches up to a three-fold speedup (with p up to 10^6) when compared to state-of-the-art schemes for sparse learning such as `glmnet` and `ncvreg`.

1 Introduction

We consider the usual linear regression setup with response $y \in \mathbb{R}^n$, model matrix $X \in \mathbb{R}^{n \times p}$, and regression coefficients $\beta \in \mathbb{R}^p$. We will assume that the columns of X are mean-centered and standardized to have unit ℓ_2-norm and y is centered. In the high-dimensional learning framework with $p \gg n$, it is desirable to estimate β under the assumption that it is sparse, i.e., β has few nonzero entries [8, 18]. With least squares as the data-fidelity term, this leads to the well-known best-subset selection problem [25] in constrained form:

$$\widehat{\beta}_{\text{L0}} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \quad \frac{1}{2}\|y - X\beta\|_2^2 \qquad \text{s.t.} \qquad \|\beta\|_0 \leq k, \tag{1}$$

*H. Hazimeh's research was partially supported by ONR-N000141512342. email: `hazimeh@mit.edu`

†R. Mazumder's research was partially supported by NSF-IIS-1718258 and ONR-N000141512342 . email: `rahulmaz@mit.edu`

where, $\|\beta\|_0 \stackrel{\text{def}}{=} \sum_{i\in[p]} \mathbb{1}[\beta_i \neq 0]$ denotes the L_0 pseudo-norm of β and k controls the model size. The statistical properties of $\widehat{\beta}_{\text{L0}}$ are well-understood, see for example [13, 14, 30, 37] (and references therein), and this estimator is widely considered as a 'gold-standard' for sparse regression (assuming it can be computed). Suppose data is generated from a true linear model $y = X\beta^0 + \epsilon$ where β^0 is sparse and $\epsilon_i \stackrel{\text{iid}}{\sim} \text{N}(0, \sigma^2)$. It is well-known that $\widehat{\beta}_{\text{L0}}$ has excellent statistical properties (variable selection, estimation and prediction error) when the signal-to-noise ratio (SNR)[1] is high. Indeed, in several regimes, $\widehat{\beta}_{\text{L0}}$ has superior statistical properties compared to computationally friendlier schemes (e.g., based on L_1-regularization) [5, 38, 4]. However, when the SNR becomes low, $\widehat{\beta}_{\text{L0}}$ suffers from over-fitting—its performance deteriorates in terms of variable selection and prediction error [24, 12]. Recently [24, 17] explored the relatively less-known characteristic of $\widehat{\beta}_{\text{L0}}$ in the low SNR regime, wherein continuous-shrinkage based estimators like the ridge/Lasso are found to deliver better predictive models compared to $\widehat{\beta}_{\text{L0}}$. [24] propose to circumvent this unfavorable behavior of $\widehat{\beta}_{\text{L0}}$ by considering a regularized variant of the form:

$$\min_{\beta\in\mathbb{R}^p} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_q \qquad \text{s.t.} \qquad \|\beta\|_0 \leq k \tag{2}$$

where $\|\beta\|_q, q \in \{1, 2\}$ is the L_q norm of β, and λ controls the amount of shrinkage. [24] demonstrate (theoretically and empirically) that estimator (2) has superior or comparable predictive accuracy compared to Lasso/ridge, and it often leads to solutions with much fewer nonzeros. Thusly motivated, in this paper, we consider a penalized version[2] of regularized subset-selection estimators introduced above:

$$\min_{\beta\in\mathbb{R}^p} \quad F(\beta) \stackrel{\text{def}}{=} f(\beta) + \lambda_0\|\beta\|_0 \tag{3}$$

where, $\lambda_0 > 0$ and $f(\beta)$ is the least squares term with additional convex penalties:

$$f(\beta) \stackrel{\text{def}}{=} \frac{1}{2}\|y - X\beta\|^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2, \tag{4}$$

and $\lambda_1, \lambda_2 \geq 0$ are a-priori known tuning parameters that control the amount of continuous shrinkage. In this paper, we will set at least one of the tuning parameters λ_1, λ_2 to zero, and focus on the following cases: (i) $\lambda_1 = 0$ and $\lambda_2 > 0$, with Problem (3) denoted as (L_0L_2) (ii) $\lambda_1 > 0$ and $\lambda_2 = 0$, with (3) denoted as (L_0L_1) and (iii) $\lambda_1 = \lambda_2 = 0$, with (3) denoted as (L_0).

Current computational landscape: We discuss a key aspect of Problem (3): its computational properties. Computing $\widehat{\beta}_{\text{L0}}$ is known to be NP-hard [26] – in fact, the well-known R package `leaps` can compute solutions to Problem (1) for $n \geq p \approx 30$. Recently, [5] have shown that significant computational advances in mixed integer optimization (MIO) can be leveraged to compute near-optimal solutions to Problem (1) for instances much larger than what was considered possible in the wider statistics community since [11]. [5] demonstrated that for problem instances with $p \approx 1000$, high-quality solutions can be obtained for the best-subset problem within a few minutes with Gurobi's (commercial-grade) MIO solver, when initialized with good warm-starts available from discrete first order algorithms similar to iterative hard thresholding (IHT)[3]. What sets this MIO-based framework apart from other heuristic approaches is its ability to deliver certificates of (approximate) optimality via dual bounds, at the cost of longer computation times. [24] adapt the approach of [5] to Problem (2). [4] propose an impressive cutting plane method (using Gurobi)

[1] For a linear model with $y_i = \mu_i + \epsilon_i, i \in \{1, \ldots, n\}$ we define SNR=Var(μ)/Var(ϵ).

[2] The penalized version is chosen primarily from a computational viewpoint.

[3] Usually, Gurobi can take anywhere between 5-30 minutes to get a near-optimal solution for a particular subset size when warm-started with a solution obtained via the discrete first order methods

for subset selection that works well with mild sample correlations and a sufficiently large n. [22] demonstrate the use of MIO in the context of an L_0-variant of the Dantzig Selector. Sophisticated mathematical optimization-based techniques such as MIO seem to be useful for applications where near real-time training is not of primary importance, but being able to obtain high quality solutions with optimality-certificates is of foremost importance. On the other end, the extremely efficient and optimized implementations of solvers for L_1-regularization (Lasso) (e.g., `glmnet` [10]) can compute an entire regularization path (with a hundred values of the tuning parameter) in usually less than a second. Indeed, it seems that there is a steep price to pay in terms of computation time for using best-subsets [5] versus popular learning schemes like Lasso. As pointed out by [17], the increased computation time might discourage practitioners from adopting global optimization-based solvers for Problem (1) for daily data-analysis purposes. However, it is well known [20, 5, 38, 23, 32] that there is a significant gap in the statistical quality of solutions that can be achieved via Lasso (and its variants) and near-optimal solutions to nonconvex subset-selection type procedures. Furthermore, as we explore in this paper, the choice of an algorithm can significantly affect the quality of solutions obtained: on many instances, algorithms that perform a better job in optimizing the nonconvex subset-selection criterion (3) result in superior-quality statistical estimators (for example, in terms of support recovery). In our experiments, we observed that when the underlying statistical problem is difficult, almost all state-of-the-art algorithms for sparse learning (Lasso, iterative hard thresholding, stepwise regression, MCP penalized regression, etc), as implemented in popular packages, fail to recover the support of β^0. However, a better optimization of Problem (3), using local combinatorial optimization methods we develop herein, seems to ameliorate this problem – in comparable run times.

Our contributions: The above discussion suggests that it is critical to develop an algorithmic framework for subset-selection type problems, leading to near-optimal solutions for problems with $p \approx 10^3$-10^6, in times that are comparable to fast coordinate-wise algorithms for Lasso/nonconvex (MCP) penalized regression, for example. To this end, we revisit popular heuristic approaches for Problem (1) that rely on greedy stepwise methods [16] and/or iterative hard thresholding (IHT) methods [6, 5]. We then ask the following questions: (i) Can we use more advanced techniques from computational mathematical optimization to develop efficient algorithms for Problem (3) with associated optimality properties, suitably defined? (ii) Can the algorithms be made comparable in speed with efficient algorithms for Lasso, for example? Addressing these questions is the primary goal of this paper.

We draw inspiration from the efficiency of coordinate descent (CD) based algorithms popularly used in the context of sparse regression [7, 23, 10, 27]. However, unlike convex problems like Lasso, Problem (3) is nonconvex, and therefore, investigating the quality of solutions is more delicate. Since we seek to create algorithms with run times comparable to `glmnet` and `ncvreg` (for example), it seems that the notion of global optimality, which is closely linked to the global-optimization based MIO framework, is practically unrealistic. We will thus need to investigate weaker notions of optimality. To this end, conditions necessary for a solution β to be optimal for Problem (3) motivate various notions of stationarity or local optimality (as we define in this paper). Indeed, as we discuss in this paper, the notion of a stationary solution is closely linked to the type of algorithm used to obtain solutions to Problem (3). In other words, the quality of a solution depends upon the algorithm used for Problem (3). For example (see Section 2), if $S \subset \{1, 2, \ldots, p\}$ then $\min\{f(\beta) \mid \beta_i = 0, i \notin S\}$ leads to a stationary solution for Problem (3). However, as we show, it is possible to obtain better (in terms of smaller objective value) solutions with more advanced optimization routines. Since Problem (3) is the sum of a smooth convex loss and a non-smooth regularizer, which is separable across the coordinates of β, one can apply IHT-type methods [24, 6];

and even coordinate-wise algorithms [29] to get good solutions to Problem (3). The fixed points of these algorithms correspond to restricted notions of stationary solutions for Problem (3)— fixed points associated with an IHT-type algorithms always include the fixed points associated with coordinate-wise algorithms (see Section 2). In fact, the class of coordinate-wise stationary solutions are sensitive to whether we perform a complete minimization for every coordinate or not. One can obtain even more refined classes of stationary solutions by drawing inspiration from local combinatorial optimization algorithms. To this end, we introduce the notion of *swap inescapable minima* (Section 2) — in words, these are stationary solutions that cannot be improved by (locally) perturbing the current support of a stationary solution and optimizing over the new support. This introduces a hierarchy of necessary optimality conditions for Problem (3) and inspires new algorithms for Problem (3). We demonstrate that as one moves up the hierarchy, it is possible to obtain (i) better solutions for Problem (3) with marginally increasing run times and (ii) estimators with superior statistical properties, when compared to several other sparse regularization techniques with similar computational scalability and run times.

We summarize our contributions below:

1. We introduce a family of necessary optimality conditions for Problem (3), leading to a hierarchy of classes of stationary solutions. Classes higher up in the hierarchy are of higher quality and include stationary solutions that cannot be improved by local perturbations to the support of the current solution.

2. We develop an algorithmic framework, relying on cyclic CD and local combinatorial optimization that enables us to attain these classes of stationary solutions. We explore how the choice of an optimization algorithm for Problem (3) affects the quality of solutions obtained. We establish a novel convergence analysis of a variant of a coordinate-wise algorithm that performs full optimization per coordinate. Our local combinatorial optimization algorithms are based on highly structured MIO formulations that can run in the order of seconds to minutes when p is in the order of 10^3 to 10^5.

3. We provide an open-source and extensible C++ toolkit `L0Learn` with an R interface[4], implementing the algorithms in this paper. Our implementation pays careful attention to several delicate computational details and exploits problem-structure to achieve run times that are often faster than efficient implementations of Lasso (`glmnet`), nonconvex penalized regression (`ncvreg`). Typical speedups of a version of our algorithm on real and synthetic datasets are between 25% to 300% for p up to 10^6 and $n \approx 10^3$.

4. In a series of experiments on real and synthetic datasets, we demonstrate that the algorithms proposed herein, do a better job at optimizing Problem (3) and achieve superior statistical performance in terms of estimation, prediction, and variable selection accuracy compared to popularly used state-of-the-art methods for sparse learning. The quality of solutions obtained by our algorithms is empirically found to be similar to those available via MIO on the full problem [5], but with significantly smaller and practical run times.

1.1 Related work

There is a large literature on algorithms for sparse linear regression – see for example [5, 1], for an overview. A popular heuristic for best-subset selection (i.e., Problem (1)) is (greedy) stepwise

[4] Available at `http://github.com/hazimehh/L0Learn`

regression [17, 16] – however, this becomes prohibitively expensive as soon as the number of features is of the order of tens of thousands. IHT or proximal gradient type algorithms [6, 5] are also a popular choice for Problem (1) and its Lagrangian version. However, IHT-type methods require full gradient evaluations at every iteration making it computationally less appealing than coordinate descent (CD)-type methods—we have also experienced a similar observation for Problem (3). Indeed, in the case of Lasso, CD-type algorithms are computationally more attractive compared to proximal gradient methods [10, 27]. Furthermore, we show in Section 2 that stationary solutions associated with IHT-type algorithms strictly contain those associated with CD-type algorithms. Similar observations on IHT and CD-type algorithms have also been made by [1] for Problem (1) and [29]. Moreover, for the least squares loss [23, 7] propose efficient cyclic CD algorithms for continuous (nonconvex) regularizers (like MCP, SCAD), adapting the framework of [35], which uses full minimization in every coordinate block. We note that the objective function in Problem (3) is not quasiconvex in every coordinate, and hence the convergence of CD with complete minimization cannot be guaranteed [35]. [29] have used random CD for the penalized version of Problem (1) with a conservative step size. The conservative step size and the random choice of coordinates facilitate the convergence analysis of the algorithms. However, a conservative step size leads to a class of stationary solutions containing that generated by complete minimization (see Section 3). Moreover, based on our empirical evidence, a (partially greedy) cyclic CD with complete minimization for every coordinate is found to be superior to random CD methods with a conservative step-size in both solution quality and running time. Due to its competitive performance, we use a (partially greedy) cyclic CD algorithm to obtain coordinate-wise stationary solutions for Problem (3). However, proving the convergence of this algorithm to a coordinate-wise stationary solution is not straightforward – rigorously establishing convergence is an important contribution of our work. We note that [2] advocates the use of a cyclic CD rule over randomized CD for convex problems due to their faster convergence. Furthermore, in the context of random CD algorithm, calls to the random number generator for large-scale problems can become a computational burden [27].

An important aspect of our work that makes it different from earlier works on CD-like algorithms for best-subset selection [1, 29] is the exploration of local combinatorial optimization schemes to define finer classes of stationary solutions. A special case of this, in the form of single coordinate-swaps, was proposed by by [1] for Problem (1) — however, there are important differences in our approaches as we consider much smaller classes of stationary solutions, described via (local) combinatorial optimization schemes. In addition, our work carefully considers computational efficiency and scalability to large problems, an aspect which has not been explored previously by similar algorithms (see the discussion in Sections 2 and 5).

1.2 Notation and Preliminaries

We make use of the following notation in this paper. We denote the set $\{1, 2, \ldots, p\}$ by $[p]$, the canonical basis for $\mathbb{R}^p$ by $e_1, \ldots, e_p$, and the standard ℓ_2 Euclidean norm by $\|\cdot\|$. For $\beta \in \mathbb{R}^p$, Supp(β) denotes its support, i.e., the indices with non-zero entries. For $S \subseteq [p]$, $\beta_S \in \mathbb{R}^{|S|}$ denotes the subvector of β with indices in S. Similarly, X_S denotes the submatrix of X with column indices S. U^S denotes a $p \times p$ matrix whose ith row being e_i if $i \in S$ and zero otherwise. Thus, for any $\beta \in \mathbb{R}^p$, $(U^S\beta)_i = \beta_i$ if $i \in S$ and $(U^S\beta)_i = 0$ if $i \notin S$.

We use the shorthands: (i) L_0L_2 to denote Problem (3) with $\lambda_1 = 0$ and $\lambda_2 > 0$; (ii) L_0L_1 to denote Problem (3) with $\lambda_1 > 0$ and λ_0; and (iii) L_0 to denote Problem (3) with $\lambda_1 = \lambda_2 = 0$. Furthermore, for Problem (3) we assume that $\lambda_0 > 0$.

2 Necessary Optimality Conditions

We study different notions of necessary optimality conditions for Problem (3). We start with the basic notion of *stationary solutions*, and we subsequently refine this notion in Sections 2.2 and 2.3.

2.1 Stationary Solutions

For a function $g : \mathbb{R}^p \to \mathbb{R}$ and a vector $d \in \mathbb{R}^p$, we denote the (lower) directional derivative[3, 35] of g at β in the direction d by:

$$g'(\beta; d) \stackrel{\text{def}}{=} \liminf_{\alpha \downarrow 0} \left(\frac{g(\beta + \alpha d) - g(\beta)}{\alpha} \right).$$

Directional derivatives play an important role in describing necessary optimality conditions for optimization problems [3]. For example, let us consider the unconstrained minimization of g when it is continuously differentiable. Here, the well-known first-order stationarity condition, $\nabla g(\beta) = 0$ follows from imposing $g'(\beta; d) = \nabla g(\beta)^T d \geq 0$ for any d. Note that $\beta \to f(\beta)$ is convex with any subgradient denoted by $\nabla f(\beta) \in \mathbb{R}^p$. If β has support S, then, the function $u_S \to f(u_S)$ is differentiable at β_S. Thus, $\nabla_S f(\beta) \in \mathbb{R}^{|S|}$ is the gradient of $f(u_S)$ at β_S.

Although the objective $F(\beta)$ (of Problem (3)) is not continuous, it is insightful to use the notion of a directional derivative to arrive at a basic definition of stationarity for Problem (3)

Definition 1. *(Stationary Solution) A vector $\beta \in \mathbb{R}^p$ is a stationary solution for Problem (3) if for every direction vector $d \in \mathbb{R}^p$, the lower directional derivative satisfies: $F'(\beta; d) \geq 0$.*

The next lemma gives a more explicit characterization of $F'(\beta; d)$ being non-negative.

Lemma 1. *Let $\beta \in \mathbb{R}^p$ have support S. Then, β is a stationary solution of Problem (3) iff $\nabla_S f(\beta) = 0$.*

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

$$F^{'}(\beta; d) = \begin{cases} \langle \nabla_S f(\beta), d_S \rangle & \text{if } d_{S^c} = 0 \\ \infty & \text{o.w.} \end{cases} \tag{5}$$

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

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

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

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

We now consider the case when $d_{S^c} \neq 0$. In this case, $\lim_{\alpha \downarrow 0} \text{Term III} = \infty$; and since the limit of Term I is bounded, we have $F^{'}(\beta; d) = \infty$. Thus, we have shown that (5) holds. From (5), we have $F^{'}(\beta; d) \geq 0$ for all d iff $\nabla_S f(\beta) = 0$. □

Note that $\nabla_i f(\beta) = 0$ for every $i \in \text{Supp}(\beta)$ can be equivalently written as:

$$\beta_i = \text{sign}(\widetilde{\beta}_i) \frac{|\widetilde{\beta}_i| - \lambda_1}{1 + 2\lambda_2} \;\text{ and }\; |\widetilde{\beta}_i| > \lambda_1, \qquad \text{for all } i \in \text{Supp}(\beta), \tag{6}$$

where, $\widetilde{\beta}_i \stackrel{\text{def}}{=} \langle y - \sum_{j \neq i} X_j \beta_j, X_i \rangle$. Characterization (6) suggests that a stationary solution β, does not depend on the regularization parameter λ_0. Moreover, (6) does not impose any condition on the coordinates outside the support of β. In the next remark, we show that a stationary solution is also a local minimum for Problem (3).

Remark 1. *We note that a stationary solution β^* is a local minimum for Problem* (3)*. We present a proof of this result below.*

By the continuity of f, there exists a positive scalar δ and a non-empty ball $R = \{\beta \in \mathbb{R}^p \mid ||\beta^* - \beta|| < \delta\}$ such that for every $\beta \in R$, we have $|f(\beta^*) - f(\beta)| < \lambda_0$. Let $S = Supp(\beta^*)$. We assume w.l.o.g. that δ is small enough so that if $i \in S$, then for every $\beta \in R$, we have $i \in Supp(\beta)$. For any $\beta \in R$, if $\beta_i \neq 0$ for some $i \notin S$, we have $(||\beta^*||_0 - ||\beta||_0) \leq -1$. This implies

$$F(\beta^*) - F(\beta) = f(\beta^*) - f(\beta) + \lambda_0(||\beta^*||_0 - ||\beta||_0) \leq |f(\beta^*) - f(\beta)| + \lambda_0(-1) < \lambda_0 - \lambda_0 = 0.$$

Otherwise, if $Supp(\beta) = S$, then the stationarity of β^* and convexity of f imply that $f(\beta^*) \leq f(\beta)$ and consequently $F(\beta^*) \leq F(\beta)$. Therefore, for any $\beta \in R$, we have $F(\beta^*) \leq F(\beta)$.

We now introduce refinements of the class of stationary solutions introduced above.

2.2 Coordinate-wise Minima

We consider a class of stationary solutions inspired by coordinate-wise algorithms [35, 1, 3]. A stationary point β, is a coordinate-wise minimum for Problem (3) if optimization with respect to every individual coordinate cannot improve the objective. The definition is given below:

Definition 2. *(Coordinate-wise (CW) Minimum) A vector $\beta^* \in \mathbb{R}^p$ is a CW minimum for Problem* (3) *if for every $i \in [p]$, β_i^* is a minimizer of $F(\beta^*)$ w.r.t. the ith coordinate (with others held fixed), i.e.,*

$$\beta_i^* \in \arg\min_{\beta_i \in \mathbb{R}} F(\beta_1^*, \ldots, \beta_{i-1}^*, \beta_i, \beta_{i+1}^*, \ldots, \beta_p^*). \tag{7}$$

Let $i \in [p]$ and $\widetilde{\beta}_i$ be a scalar defined as $\widetilde{\beta}_i \stackrel{\text{def}}{=} \langle y - \sum_{j \neq i} X_j \beta_j^*, X_i \rangle$. As every column of X has unit ℓ_2-norm, solutions to Problem (7) are given by the following thresholding operator $\widetilde{T}$:

$$\widetilde{T}(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2) \stackrel{\text{def}}{=} \arg\min_{\beta_i \in \mathbb{R}} \Big\{ \frac{1 + 2\lambda_2}{2} \Big(\beta_i - \frac{\widetilde{\beta}_i}{1 + 2\lambda_2} \Big)^2 + \lambda_1 |\beta_i| + \lambda_0 \mathbb{1}[\beta_i \neq 0] \Big\}, \tag{8}$$

where, (tuning parameters) $\{\lambda_i\}_0^2$ and $\widetilde{\beta}_i$ are all fixed, and $\widetilde{T}(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2)$ is set-valued. Lemma 2 provides an explicit characterization of $\widetilde{T}(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2)$.

Lemma 2. *(Univariate Minimization) Let $\widetilde{T}$ be the thresholding operator defined in (8). Then,*

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

Proof. Let $g(u)$ denote the objective function minimized in (8), i.e.,

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

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

Lemma 2 and Definition 2 provide an explicit characterization of CW minima in Lemma 3

Lemma 3. *Let $\beta^* \in \mathbb{R}^p$ and define $\widetilde{\beta}_i = \langle y - \sum_{j \neq i} X_j \beta_j^*, X_i \rangle$ for all $i \in [p]$. Then, β^* is a CW minimum iff*

$$\begin{aligned} & \beta_i^* = \operatorname{sign}(\widetilde{\beta}_i) \tfrac{|\widetilde{\beta}_i| - \lambda_1}{1+2\lambda_2} \quad \text{and} \quad |\beta_i^*| \geq \sqrt{\tfrac{2\lambda_0}{1+2\lambda_2}}, && \text{for every } i \in \mathit{Supp}(\beta^*) \\ \text{and} \quad & \tfrac{|\widetilde{\beta}_i| - \lambda_1}{1+2\lambda_2} \leq \sqrt{\tfrac{2\lambda_0}{1+2\lambda_2}} && \text{for every } i \notin \mathit{Supp}(\beta^*). \end{aligned} \tag{9}$$

Comparing (9) to characterization (6) of stationary solutions, we see that the class of stationary solutions contains the class of CW minima, and the containment is strict for a general X.

2.3 Swap Inescapable Minima

We now consider stationary solutions that further refine the class of CW minima, using notions from local combinatorial optimization. Given a CW minimum β^*, one might consider obtaining another candidate solution with a lower objective by a "swapping" operation described as follows: We set some non-zeros in β^* to zero and some entries from outside the support of β^* to non-zero. Then, we optimize over the new support using one of the following rules:

- **Partial Optimization**: Here we optimize only w.r.t the coordinates added from outside the support–this leads to *Partial Swap Inescapable Minima*, as described in Section2.3.1.
- **Full Optimization:** Here we optimize w.r.t all the coordinates in the new support–this leads to *Full Swap Inescapable Minima*, as described in Section 2.3.2.

If the resulting solution leads to a lower objective compared to β^*, then we have successfully escaped from β^* to a better solution. If β^* cannot be improved via the above strategy, we call it a *Swap Inescapable* minimum. Our proposed notion is inspired by the work of [1] who introduced an interesting special case of swap inescapable minima for the cardinality-constrained problem, where their

swapping operation is done with respect to single coordinates using partial optimization. However, the problem studied herein, i.e., Problem (3) is different — we consider an L_0-penalized version and $f(\beta)$ is non-smooth. Furthermore, our classes of swap inescapable minima allow for multiple coordinates to be swapped. We also allow for partial and full optimization for the subproblems.

2.3.1 Partial Swap Inescapable (PSI) Minima

We formally introduce Partial Swap Inescapable (PSI) minima. In words, these stationary solutions (or minima) cannot be escaped by swapping any two subsets of coordinates (from inside and outside the support) and performing partial optimization over the new support (i.e., we optimize only w.r.t the new coordinates added to the support). We recall that for any $L \subseteq [p]$, the notation $U^L\beta$ denotes a vector with ith coordinate $(U^L\beta)_i = \beta_i$ if $i \in L$ and $(U^L\beta)_i = 0$ if $i \notin L$.

Definition 3. *(PSI Minima) Let k be a positive integer. A vector β^* with support S is a Partial Swap Inescapable minimum of order k, denoted by PSI(k), if it is a stationary solution and for every $S_1 \subseteq S$ and $S_2 \subseteq S^c$, such that $|S_1| \leq k$ and $|S_2| \leq k$, the following holds*

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

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

Lemma 4. *A vector $\beta^* \in \mathbb{R}^p$ is a PSI(1) minimum iff*

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

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

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

Proof. The result can be readily derived from Lemma 2 and Definition 3. □

Lemmas 3 and 4 suggest that PSI(1) minima, when compared to CW minima, impose additional restrictions on the magnitude of nonzero coefficients. We also note that the class of CW minima contains PSI minima for any k. Furthermore, as k increases, the class of PSI(k) minima becomes smaller — till it coincides with the class of global minimizers of Problem (3). Section 4.1 introduces an algorithm that combines coordinate descent and local combinatorial optimization based on MIO to achieve PSI(k) minima for any given $k \in [p]$.

2.3.2 Full Swap Inescapable (FSI) Minima

We define Full Swap Inescapable (FSI) minima – they differ from PSI minima in that a *full* optimization is allowed on the new support after swapping the coordinates.

Definition 4. *(FSI Minima) Let k be a positive integer. A vector β^* with support S is a FSI minimum of order k, denoted by FSI(k), if for every $S_1 \subseteq S$ and $S_2 \subseteq S^c$, such that $|S_1| \leq k$ and $|S_2| \leq k$, the following holds*

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

We note that for a fixed k, the class of FSI(k) minima is contained inside the class of PSI(k) minima (this is a consequence of the definition). As k increases, the class of FSI(k) minima becomes smaller till it coincides with the set of global minimizers of Problem (3). Section 4.2 introduces an algorithm that combines coordinate descent with MIO to generate FSI(k) minima.

2.4 Stationarity Motivated by Iterative Hard Thresholding (IHT)

Proximal-gradient type algorithms such as IHT are popularly used for L_0-constrained and L_0-penalized least squares problems [6]. It is insightful to consider the class of stationary solutions associated with IHT and study how they compare to CW minima. Let $f_d(\beta) := \frac{1}{2}\|y - X\beta\|^2 + \lambda_2\|\beta\|^2$. The gradient of $f_d(\beta)$, i.e., $\nabla f_d(\beta)$ is Lipschitz with parameter L (say), i.e., $\|\nabla f_d(\beta) - \nabla f_d(\alpha)\| \leq L\|\beta - \alpha\|$ for all $\beta, \alpha \in \mathbb{R}^p$. Then it follows that [28]

$$Q_L(\beta;\alpha) := \frac{L}{2}\|\beta-\alpha\|_2^2 + \langle \nabla f_d(\alpha), \beta - \alpha\rangle + f_d(\alpha) \geq f(\beta) \qquad \forall \beta, \alpha \in \mathbb{R}^p.$$

If β^k denotes the value of β at the kth iteration, then, to obtain the $(k+1)$th iterate, IHT minimizes an upper bound to Problem (3) of the form: $Q_L(\beta;\beta^k)+\lambda_1\|\beta\|_1+\lambda_0\|\beta\|_0$. This leads to the following sequence of updates:

$$\beta^{k+1} \in \underset{\beta\in\mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2\tau}\|\beta - (\beta^k - \tau\nabla f_d(\beta^k))\|^2 + \lambda_1\|\beta\|_1 + \lambda_0\|\beta\|_0 \right\}, \tag{10}$$

where, $\tau > 0$ is a fixed step size. We say that $\alpha \in \mathbb{R}^p$ is a fixed point of update (10) if $\beta^k = \alpha$ leads to $\beta^{k+1} = \alpha$. This also defines another notion of stationarity for Problem (3), which is different from the CW minima described previously. To this end, we consider the following theorem which establishes the convergence of β^k to a fixed point of update (10).

Theorem 1. *Let L be defined as above. The sequence $\{\beta^k\}$ defined in (10) converges to a fixed point β^* of update (10) for any $\tau < \frac{1}{L}$. Note that β^* is a fixed point iff*

$$\begin{aligned} &\beta_i^* = \text{sign}(\widetilde{\beta}_i)\tfrac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} \quad \textit{and} \quad |\beta_i^*| \geq \sqrt{2\lambda_0\tau} && \textit{for } i \in \textit{Supp}(\beta^*) \\ \textit{and} \quad &\tfrac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} \leq \sqrt{\tfrac{2\lambda_0}{(1+2\lambda_2)^2\tau}} && \textit{for } i \notin \textit{Supp}(\beta^*) \end{aligned} \tag{11}$$

where $\widetilde{\beta}_i = \langle y - \sum_{j\neq i} X_j\beta_j^*, X_i\rangle$.

We do not provide a proof of the above theorem as it is similar to the proof of [24, 5] (which considers the cardinality constrained version of Problem (3)) – see also [21] for a proof of IHT for a cardinality constrained optimization problems when the objective is differentiable. Characterization (11) suggests a class of minimizers inspired by IHT as defined below.

Definition 5. *A vector β^* is an IHT minimum for Problem (3) if it satisfies (11) for $\tau < \frac{1}{L}$.*

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

Remark 2. *Let M be the largest eigenvalue of X^TX and take $L = M+2\lambda_2$. By Theorem 1, any $\tau < \frac{1}{M+2\lambda}$ ensures the convergence of updates (10). Moreover, since the columns of X are normalized, we have $M \geq 1$. Comparing characterization (11) (of IHT minima) to characterization (9) (of CW minima) we see that the class of IHT minima includes the class of CW minima. Indeed, typically for high-dimensional problems, M is much larger than 1, making the class of IHT minima much larger than CW minima (See Section 6.2 for numerical examples).*

Below we summarize a hierarchy of classes of stationary solutions introduced in this section.

$$\textsc{Hierarchy} \qquad \begin{matrix}\text{FSI}(k)\\ \text{Minima}\end{matrix} \subseteq \begin{matrix}\text{PSI}(k)\\ \text{Minima}\end{matrix} \subseteq \begin{matrix}\text{CW}\\ \text{Minima}\end{matrix} \subseteq \begin{matrix}\text{IHT}\\ \text{Minima}\end{matrix} \subseteq \begin{matrix}\text{Stationary}\\ \text{Solutions}\end{matrix} \tag{12}$$

Finally, we note that when k is large, the classes of FSI(k) minima and PSI(k) minima coincide with the global minimizers of Problem (3). Section 2 introduced a hierarchical class of stationary conditions for Problem (3). We now discuss optimization algorithms that converge to these stationary classes: CW minima, PSI minima and FSI minima. Section 3 discusses coordinate-wise algorithms that guarantee convergence to CW minima; and Section 4 discusses local combinatorial optimization algorithms that reach stationary points corresponding to FSI(k)/PSI(k) minima.

3 Cyclic Coordinate Descent: Algorithmic Convergence

In this section, we introduce a variant of cyclic CD that performs a full minimization for every coordinate [3]. We analyze its convergence behavior—in particular, we prove a new result establishing convergence to a unique CW minimum (which depends upon the initialization) and an asymptotic linear rate of convergence. We note that if we avoid complete minimization and use a conservative step size, the proofs of convergence become straightforward by virtue of a sufficient decrease of the objective value after every coordinate update[5]. However, using CD with a conservative step size for Problem 3 has a detrimental effect on the solution quality. Indeed, by characterizing the fixed points, it can be shown that a sub-optimal step size leads to a class of stationary solutions that contains CW minima. While cyclic CD has been studied before with continuous regularizers [23, 7] with a least-squares data fidelity term, to our knowledge, the study of cyclic CD (with associated convergence analysis) for Problem (3) is novel.

Recall that cyclic CD updates coordinates in the order dictated by a a-priori specified permutation of $\{1, 2, \ldots, p\}$. Before diving into a formal treatment of our algorithm, we briefly discuss why CD, and in particular cyclic CD, seems to be well-suited for our problem – especially in the light of its excellent computational performance in our experiments.

Why Cyclic CD? Cyclic CD has been practically shown to be among the fastest algorithms for Lasso [10] and continuous nonconvex regularizers (such as MCP, SCAD, etc) [23, 7]. The coordinate updates in cyclic CD have low cost and can exploit sparsity–specifically, through sparse residual updates and active set convergence[10]. This makes it well-suited for high-dimensional problems with $n \ll p$ and p of the order of tens-of-thousands to million(s). On the other hand, for problem instances with a similar size, methods that require the evaluation of the full gradient (e.g., proximal gradient descent, greedy coordinate descent, etc) are computationally more expensive. For example, proximal gradient descent methods do not exploit sparsity-based structure as well as CD-based methods [10, 27]. We also note that based on our empirical experiments, random CD seems to exhibit slower convergence in practice, (see Section 6.2)–see also related discussions by [2] for convex problems.

To complement the aforementioned computational advantages reported in earlier research, our numerical experience suggests that cyclic CD has an edge over competing algorithms, both in terms of optimization objective (Section 6.2) and statistical performance (see Sections 6.3,6.4, and 6.6). Solutions to the (L_0), (L_0L_1), and (L_0L_2) problems are generally expected to have fewer

[5]Such an observation appears in establishing convergence of IHT-type algorithms–see for example,[1, 21, 5]

non-zeros than the Lasso. Since the speed of CD is sensitive to the number of non-zeros in the solution, we expect CD-like algorithms for Problem (3) to lead to faster run times–indeed this is also validated in our experiments, where we observe up to a three-fold improvement in run time compared to `glmnet` and `ncvreg`.

3.1 Convergence Analysis

We first present a formal description of our algorithm – it is essentially the cyclic CD algorithm, with some additional tweaks for reasons we discuss shortly. With initialization β^0, we update the first coordinate (with others fixed) to get β^1, and continue the updates as per a cyclic rule. Let β^k denote the iterate obtained after performing k coordinate updates. Then, β^{k+1} is obtained by updating its ith coordinate (with others held fixed) via:

$$\beta_i^{k+1} \in \underset{\beta_i \in \mathbb{R}}{\arg\min}\, F(\beta_1^k, \ldots, \beta_{i-1}^k, \beta_i, \beta_{i+1}^k, \ldots, \beta_p^k), \tag{13}$$

where $i = (k+1) \bmod p$. Recall that the operator $\widetilde{T}(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2)$ (defined in (8)) describes solutions of Problem (13). Specifically, it returns two solutions when $\frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. In such a case, we consistently choose one of these solutions[6], namely the non-zero solution. Thus, we use the new operator (note the use of T instead of $\widetilde{T}$):

$$T(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2) \overset{\text{def}}{=} \begin{cases} \text{sign}(\widetilde{\beta}_i)\frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} & \text{if } \frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ 0 & \text{if } \frac{|\widetilde{\beta}_i|-\lambda_1}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \end{cases} \tag{14}$$

for update (13). In addition to the above modification in the thresholding operator, we introduce "spacer steps" that are occasionally performed during the course of the algorithm to *stabilize* its behavior[7]—spacer steps are commonly used in the context of general nonlinear optimization problems—see [3] for a discussion. Every spacer step is performed as follows. We keep track of the supports encountered so far; and when a certain support S (say) appears for Cp-many times, where C is an a-priori fixed positive integer, we perform one pass of cyclic CD on $f(\beta_S)$. This entails updating every coordinate in S via the operator: $T(\widetilde{\beta}_i, 0, \lambda_1, \lambda_2)$ (See Subroutine 1).

Algorithm 1 summarizes the above procedure. Count[S] is an associative array that stores the number of times support S appears–it takes S as a key and the number of times S has appeared so far as a value. Count[S] is initialized to 0 for all choices of S.

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

Below we introduce several lemmas that describe the behavior of Algorithm 1. Finally, Theorem 2 establishes the convergence of Algorithm 1 to a CW minimum.

The following lemma states that Algorithm 1 is a descent algorithm.

[6] We note that this convention is used for a technical reason and is needed for our proof of Theorem 2

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

Algorithm 1: Coordinate Descent with Spacer Steps (CDSS)

Input : Initial Solution β^0, Positive Integer C
$k \leftarrow 0$
while *Not Converged* **do**
  for i *in* 1 *to* p **do**
    $\beta^{k+1} \leftarrow \beta^k$
    $\beta_i^{k+1} \leftarrow \arg\min_{\beta_i \in \mathbb{R}} F(\beta_1^k, \ldots, \beta_i, \ldots, \beta_p^k)$ using (14) `// Non-spacer Step`
    $k \leftarrow k+1$
    Count[Supp(β^k)] $\leftarrow$ Count[Supp(β^k)] + 1
    if *Count[Supp(β^k)]* $= Cp$ **then**
      $\beta^{k+1} \leftarrow$ SpacerStep(β^k) `// Spacer Step`
      Count[Supp(β^k)] $= 0$
      $k \leftarrow k+1$
return (β)

Subroutine 1: SpacerStep(β)

Input : β
for i *in* *Supp*(β) **do**
  $\beta_i \leftarrow \arg\min_{\beta_i \in \mathbb{R}} f(\beta_1, \ldots, \beta_i, \ldots, \beta_p)$ using (14) with $\lambda_0 = 0$
return (β)

Lemma 5. *Algorithm 1 is a descent algorithm and $F(\beta^k) \downarrow F^*$ for some $F^* \geq 0$.*

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

For the remainder of the section, we will make the following minor assumptions towards showing convergence to a unique limit for the (L_0) and (L_0L_1) problems. No such assumptions are needed for the (L_0L_2) problem.

Assumption 1. *Let $m = \min\{n, p\}$. Every m columns in X are linearly independent.*
Assumption 2. *(Initialization) If $p > n$, we assume that the initial estimate β^0 satisfies*

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

The following remark demonstrates that Assumption 2 is rather minor.

Remark 3. *Suppose $p > n$ and Assumption 1 holds. For the (L_0) problem, let $S \subseteq [p]$ such that $|S| = n$. If β^0 is defined such that β^0_S is the least squares solution on the support S with $\beta^0_{S^c} = 0$; then $F(\beta^0) = \lambda_0 n$ (with the least squares loss being zero). This satisfies Assumption 2. For the (L_0L_1) problem, we note that there always exists an optimal lasso solution $\widehat{\beta}$ such that $\|\widehat{\beta}\| \leq n$ (see, for e.g., [34]). Therefore, $\widehat{\beta}$ satisfies Assumption 2.*

In what follows, we assume that Assumptions 1 and 2 hold for the (L_0) and (L_0L_1) problems, and we make no assumptions for the (L_0L_2) problem.

The following lemma shows that in the (L_0) and (L_0L_1) problems, the support size of any β^k obtained by Algorithm 1 cannot exceed the minimum of n and p.

Lemma 6. *For the (L_0) and (L_0L_1) problems, $\{\beta^k\}$ satisfies $\|\beta^k\|_0 \leq \min\{n, p\}$ for all k.*

Proof. The result holds trivially if $p \leq n$. Suppose $p > n$. In the (L_0) problem, Assumption 2 states that $F(\beta^0) \leq \lambda_0 n$. Since Algorithm 1 is a descent method (by Lemma 5) we have $F(\beta^k) \leq \lambda_0 n$ for every k, which implies $f(\beta^k) + \lambda_0\|\beta^k\|_0 \leq \lambda_0 n$ and hence, $\lambda_0\|\beta^k\|_0 \leq \lambda_0 n$. Therefore, $\|\beta^k\|_0 \leq n$ for all k. Similarly, for the (L_0L_1) problem, Assumption 2 and the descent property imply $F(\beta^k) \leq f(\beta^{\ell_1}) + \lambda_0 n$ which can be equivalently written as $f(\beta^k) - f(\beta^{\ell_1}) \leq \lambda_0(n - \|\beta^k\|_0)$. But the optimality of the lasso solution implies $f(\beta^k) - f(\beta^{\ell_1}) \geq 0$, which leads to $\|\beta^k\|_0 \leq n$. □

The following lemma shows that the sequence generated by Algorithm 1 is bounded.

Lemma 7. *The sequence $\{\beta^k\}$ is bounded.*

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

We now study the L_0 problem. Firstly, if $p \leq n$, then the objective function for the (L_0) problem is coercive (under Assumption 1), and the previous argument used for $(L_0L_1)/(L_0L_2)$ applies. Otherwise, suppose that $p > n$. Recall that from Lemma 6, we have $\|\beta^k\|_0 \leq n$ for all $k \geq 0$; and from Assumption 2, we have $F(\beta^0) \leq \lambda_0 n$. In addition, by Lemma 5, we have $F(\beta^k) \leq \lambda_0 n$ for every k. Therefore, it follows that $\beta^k \in A$ where,

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

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

The next lemma characterizes the limit points of Algorithm 1. The proof is in Section A.

Lemma 8. *Let S be a support that is generated infinitely often by the non-spacer steps, and let $\{\beta^l\}_{l\in L}$ be the sequence of spacer steps generated by S. Then, the following hold true:*

1. *There exists an integer N such that for all $l \in L$ and $l \geq N$ we have $Supp(\beta^l) = S$.*
2. *There exists a subsequence of $\{\beta^l\}_{l\in L}$ that converges to a stationary solution β^*, where, β^*_S is the unique minimizer of $\min_{\beta_S} f(\beta_S)$ and $\beta^*_{S^c} = 0$.*
3. *Every subsequence of $\{\beta^k\}_{k\geq 0}$ with support S converges to β^* (as in Part 2, above).*
4. *β^* satisfies $|\beta^*_j| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$ for every j in S.*

Lemma 9 shows that the support corresponding to any limit point of $\{\beta^k\}$ appears infinitely often.

Lemma 9. *Let B be a limit point of $\{\beta^k\}$ with $Supp(B) = S$, then $Supp(\beta^k) = S$ for infinitely many k.*

Proof. We prove this result by using a contradiction argument. To this end, suppose support S occurs only finitely many times. Since there are only finitely many supports, there is a support S' with $S' \neq S$; and a subsequence $\{\beta^{k'}\}$ of $\{\beta^k\}$ which satisfies: $\text{Supp}(\beta^{k'}) = S'$ for all k'; and $\beta^{k'} \to B$ as $k' \to \infty$. However, this is not possible by Part 3 of Lemma 8. □

Lemma 10 is technical and will be needed in the proof of convergence of Algorithm 1. The proof is in Section A.

Lemma 10. *Let $B^{(1)}$ and $B^{(2)}$ be two limit points of the sequence $\{\beta^k\}$, with supports S_1 and S_2, respectively. Suppose that $S_2 = S_1 \cup \{j\}$ for some $j \notin S_1$. Then, exactly one of the following holds:*

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

The following establishes a lower bound on the decrease in objective value, when a non-zero coordinate is set to zero during Algorithm 1.

Lemma 11. *Let β^k be an iterate of Algorithm 1 with $\beta_j^k \neq 0$ for some $j \in [p]$. Let β^{k+1} correspond to a non-spacer step which updates coordinate j to 0, i.e., $\beta_j^{k+1} = 0$. Then, the following holds:*

$$F(\beta^k) - F(\beta^{k+1}) \geq \frac{1+2\lambda_2}{2}\left(|\beta_j^k| - \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}\right)^2. \tag{15}$$

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

$$\begin{aligned} F(\beta^k) - F(\beta^{k+1}) &= -\widetilde{\beta}_j^k \beta_j^k + \frac{1+2\lambda_2}{2}(\beta_j^k)^2 + \lambda_0 + \lambda_1|\beta_j^k| \\ &\geq -|\widetilde{\beta}_j^k||\beta_j^k| + \frac{1+2\lambda_2}{2}(\beta_j^k)^2 + \lambda_0 + \lambda_1|\beta_j^k| \\ &\geq -|\beta_j^k|(|\widetilde{\beta}_j^k| - \lambda_1) + \frac{1+2\lambda_2}{2}(\beta_j^k)^2 + \lambda_0, \end{aligned} \tag{16}$$

where $\widetilde{\beta}_j^k = \langle y - \sum_{i \neq j} X_j \beta_i^k, X_j \rangle$. Since β^{k+1} is a non-spacer step which sets coordinate j to 0, the definition of the thresholding operator (14) implies $|\widetilde{\beta}_j^k| - \lambda_1 < \sqrt{2\lambda_0(1+2\lambda_2)}$. Plugging this bound into (16) and factorizing, we arrive to the result of the lemma. □

Finally, the following theorem (for proof see Section A) establishes the convergence of Algorithm 1 to a unique CW minimum.

Theorem 2. *The following holds for Algorithm 1:*

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

3.2 Rate of Convergence

In this section, we will show that Algorithm 1 exhibits an asymptotic linear rate of convergence. Theorem 2 implies that the support of the iterates stabilizes in a finite number of iterations. After the support stabilizes to a support S (say), the spacer and non-spacer steps lead to the same coordinate updates. Thus Algorithm 1 can be viewed as a cyclic CD (with full optimization per coordinate), where we cyclically apply the operator: $T(\widetilde{\beta}_i, 0, \lambda_1, \lambda_2)$ defined in (14) for every $i \in S$. We will use some new notation for the exposition of the theory in Section 3.2—the term *full cycle* will refer to a single pass of vanilla CD over all the coordinates in S. We use β^K to denote the iterate generated after performing K full cycles of CD.

Theorem 3. *Suppose the same assumptions of Theorem 2 hold. Let $\{\beta^K\}$ be the full-cycle iterates generated by Algorithm 1 and B be the limit with support S. Let m_S and M_S denote the smallest and largest eigenvalues of $X_S^T X_S$, respectively. Then, there is an integer N such that for all $K \geq N$ the following holds:*

$$\frac{F(\beta^{K+1}) - F(B)}{F(\beta^K) - F(B)} \leq 1 - \frac{m_S + 2\lambda_2}{2(1+2\lambda_2)\Big(1 + |S|(\frac{M_S+2\lambda_2}{1+2\lambda_2})^2\Big)} \tag{17}$$

Proof. By Theorem 2, we have $\beta^K \to B$, and $\exists M$ such that for all $K \geq M$, we have $\text{Supp}(\beta^K) = S$ and $\text{Supp}(B) = S$. Therefore, there exists an integer $N \geq M$ such that for $K \geq N$, $\text{sign}(\beta_i^K) = \text{sign}(B_i)$ for every $i \in S$. For $K \geq N$, it can be readily seen that the iterates β^K are the same as those generated by minimizing the following objective

$$g(\beta_S) = \frac{1}{2}\|y - X_S\beta_S\|^2 + \lambda_1 \sum_{i \in S, B_i > 0} \beta_i - \lambda_1 \sum_{i \in S, B_i < 0} \beta_i + \lambda_2 \|\beta_S\|^2, \tag{18}$$

using coordinate descent with step size $\frac{1}{1+2\lambda_2}$ and starting from the initial solution β^N. The function $\beta_S \mapsto g(\beta_S)$ is continuously differentiable and its gradient is Lipschitz continuous with parameter $L = M_S + 2\lambda_2$. Moreover, it is strongly convex [28] with strong-convexity parameter $\sigma_S = m_S + 2\lambda_2$. [2] (see Theorem 3.9) has proven a linear rate of convergence for cyclic CD when applied to strongly convex and continuously differentiable functions. Applying [2]'s result in our context leads to the conclusion of the theorem. □

4 Local Combinatorial Optimization Algorithms

Motivated by the classes of Swap Inescapable minima introduced in Section 2.3, we present algorithms to achieve solutions belonging to these classes.

4.1 Algorithms for PSI minima

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

$$\min_{\beta, S_1, S_2} \quad F(\beta^\ell - U^{S_1}\beta^\ell + U^{S_2}\beta) \quad \text{s.t.} \quad S_1 \subseteq S,\ S_2 \subseteq S^c, |S_1| \leq k,\ |S_2| \leq k \tag{19}$$

where, $S = \text{Supp}(\beta^\ell)$. Note that if there exists a feasible solution $\widehat{\beta}$ to Problem (19) satisfying $F(\widehat{\beta}) < F(\beta^\ell)$, then $\widehat{\beta}$ may not be a CW minimum. To this end, $\widehat{\beta}$ can be used to initialize Algorithm 1 to obtain a better solution for Problem (3). Otherwise, if $\widehat{\beta}$ does not exist, then β^ℓ is a PSI(k) minimum since it satisfies Definition 3. Algorithm 2 (aka CD-PSI(k)) summarizes the algorithm.

Algorithm 2: CD-PSI(k)

$\widehat{\beta}^0 \leftarrow \beta^0$
for $\ell = 0, 1, \ldots$ **do**
  $\beta^{\ell+1} \leftarrow$ Output of Algorithm 1 initialized with $\widehat{\beta}^\ell$
  if *Problem (19) has a feasible solution* $\widehat{\beta}$ *satisfying* $F(\widehat{\beta}) < F(\beta^{\ell+1})$ **then**
    $\widehat{\beta}^{\ell+1} \leftarrow \widehat{\beta}$
  else
    Stop
return β^l

Theorem 4. *Let $\{\beta^\ell\}$ be the sequence of iterates generated by Algorithm 2. For the (L_0) and $(L_0 L_1)$ problems, suppose that Assumptions 1 and 2 hold. Then, Algorithm 2 terminates in a finite number of iterations and the output is a PSI(k) minimum.*

Proof. Algorithm 2 leads to a sequence $\{\beta^i\}_0^\ell$ such that $F(\beta^\ell) < F(\beta^{\ell-1}) < \cdots < F(\beta^0)$. Since $\beta^\ell, \beta^{\ell-1}, \ldots, \beta^0$ are all outputs of Algorithm 1, they are all CW minima (by Theorem 2). Any CW minimum on a support S, is stationary for the problem: $\min_{\beta_S} f(\beta_S)$. By the convexity of $\beta_S \to f(\beta_S)$, all stationary solutions on support S have the same objective (since they all correspond to the minimum of $\min_{\beta_S} f(\beta_S)$). Thus, we have $\text{Supp}(\beta^i) \neq \text{Supp}(\beta^j)$ for any $i, j \leq \ell$ such that $i \neq j$. Therefore, a support can appear at most once during the course of Algorithm 2. Since the number of possible supports is finite, we conclude that Algorithm 2 terminates in a finite number of iterations. Finally, we note that Algorithm 2 terminates iff there is no feasible solution $\widehat{\beta}$ for (19) satisfying $F(\widehat{\beta}) < F(\beta^\ell)$. This implies that β^ℓ is a minimizer of (19) and thus a PSI(k) minimum (by Definition 3). □

We now discuss formulations and algorithms for computing a solution to the combinatorial optimization Problem (19).

MIO formulation for Problem (19)**:** Problem (19) admits a mixed integer quadratic optimiza-

tion (MIQO) formulation given by:

$$\min_{\theta,\beta,z} \quad f(\theta) + \lambda_0 \sum_{i\in[p]} z_i \tag{20a}$$

$$\text{s.t.} \quad \theta = \beta^\ell - \sum_{i\in S} e_i \beta_i^\ell (1 - z_i) + \sum_{i\in S^C} e_i \beta_i \tag{20b}$$

$$-\mathcal{M} z_i \le \beta_i \le \mathcal{M} z_i, \quad \forall i \in S^c \tag{20c}$$

$$\sum_{i\in S^c} z_i \le k \tag{20d}$$

$$\sum_{i\in S} z_i \ge |S| - k \tag{20e}$$

$$\beta_i \in \mathbb{R}, \quad \forall i \in S^c \tag{20f}$$

$$z_i \in \{0,1\}, \quad \forall i \in [p], \tag{20g}$$

where the optimization variables are $\theta \in \mathbb{R}^p$, $\beta_i, i \in S^c$ and $z \in \{0,1\}^p$. In (20), $S = \text{Supp}(\beta^\ell)$, where β^ℓ is fixed, and $\mathcal{M}$ is a Big-M parameter (a-priori specified) controlling the ℓ_∞-norm of β_{S^c}. Any sufficiently large value of $\mathcal{M}$ will lead to a solution for Problem (19); however, a tight choice for $\mathcal{M}$ affects the run time of the MIO solver–see [5] for additional details. We note that the $\|\theta\|_1$ term included in $f(\theta)$ can be expressed via linear inequalities using auxiliary variables. Thus, Problem (20) minimizes a quadratic objective with linear constraints – the optimization variables include both integer and continuous variables.

We now explain the constraints in Problem (20) and how they relate to Problem (19). To this end, let S_1 and S_2 be subsets defined in (19). Let $\theta = \beta^\ell - U^{S_1}\beta^\ell + U^{S_2}\beta$ and this relation is expressed in (20b). Let us consider any binary variable z_i where $i \in S$. If $z_i = 0$ then β_i^ℓ is removed from S, and we have $\theta_i = 0$ (see (20b)). If $z_i = 1$, then β_i^ℓ is not removed from θ, and we have $\theta_i = \beta_i^\ell \neq 0$ (see (20b)). Note that $|S_1| = \sum_{i\in S}(1 - z_i) = |S| - \sum_{i\in S} z_i$. The condition $|S_1| \le k$, is thus encoded in the constraint $\sum_{i\in S} z_i \ge |S| - k$ in (20e). Thus we have that $\|\theta_S\|_0 = \sum_{i\in S} z_i$.

Now consider any binary variable z_i where $i \in S^c$. If $z_i = 1$, then by (20c) we observe that β_i is free to vary in $[-\mathcal{M}, \mathcal{M}]$. This implies that $\theta_i = \beta_i$. If $z_i = 0$ then $\theta_i = \beta_i = 0$. Note $\sum_{i\in S^c} z_i = |S_2|$, and the constraint $|S_2| \le k$ is expressed via $\sum_{i\in S^c} z_i \le k$ in (20d). It also follows that $\|\theta_{S^c}\|_0 = \sum_{i\in S^c} z_i$. Finally, we note that the function appearing in the objective (20a) is $F(\theta)$, since $\lambda_0 \sum_{i\in[p]} z_i = \lambda_0 \|\theta\|_0$.

Remark 4. *We note that the MIO problem* (20) *has a much reduced (combinatorial) search space compared to an MIO formulation for the full Problem* (3)*. Thus, solving* (20) *for small values of* k *is usually much faster than Problem* (3)*. Furthermore, note that we use the MIO framework* (20) *so that it can quickly deliver a feasible solution with a smaller objective than the current solution – usually this is achieved within very short run times when compared to that taken towards establishing optimality via matching dual bounds. To this end, if a feasible solution with smaller objective value does not exist, the MIO-framework can certify the non-existence via dual bounds.*

Section 6 presents examples where the MIO-framework above leads to higher quality solutions in terms of the quality of solutions obtained — both from an optimization and statistical performance viewpoint.

Section 4.1.1 discusses a special case of the above MIO formulation with $k = 1$, where we can derive efficient algorithms for solving Problem (19).

4.1.1 An efficient algorithm for computing PSI(1) minima

Subroutine 2 presents an algorithm for Problem (19) with $k = 1$. That is, we search for a feasible solution $\widehat{\beta}$ of Problem (19) satisfying $F(\widehat{\beta}) < F(\beta^\ell)$.

Subroutine 2: Naive Implementation of Problem (19) with $k = 1$.

$S \leftarrow \text{Supp}(\beta^\ell)$
for $i \in S$ **do**
  for $j \in S^c$ **do**
    $v_j^* \leftarrow \underset{v_j \in \mathbb{R}}{\arg\min}\, F(\beta^\ell - e_i\beta_i^\ell + e_j v_j)$ (21)
    $F_j^* \leftarrow F(\beta^\ell - e_i\beta_i^\ell + e_j v_j^*)$ (22)
  $\vartheta \leftarrow \underset{j \in S^c}{\arg\min}\, F_j^*$ (23)
  if $F_\vartheta^* < F(\beta^*)$ **then**
    $\widehat{\beta} \leftarrow \beta^\ell - e_i\beta_i^\ell + e_\vartheta v_\vartheta^*$ (24)
    BREAK

The two for loops in Subroutine 2 can run for a total of $(p - \|\beta^\ell\|_0)\|\beta^\ell\|_0$ iterations, where every iteration requires $O(n)$ operations to perform the minimization in (21) and evaluate the new objective in (22). Therefore, Subroutine 2 entails an overall cost of $O\Big(n(p - \|\beta^\ell\|_0)\|\beta^\ell\|_0\Big)$. However, we show below that a careful implementation can reduce the cost by a factor of n; leading to a cost of $O\Big((p - \|\beta^\ell\|_0)\|\beta^\ell\|_0\Big)$-many operations.

Note that a solution v_j^* of Problem (21) is given by

$$v_j^* = \begin{cases} \text{sign}(\bar{\beta}_j)\frac{|\bar{\beta}_j| - \lambda_1}{1+2\lambda_2} & \text{if} \quad \frac{|\bar{\beta}_j| - \lambda_1}{1+2\lambda_2} \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ 0 & \text{if} \quad \frac{|\bar{\beta}_j| - \lambda_1}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \end{cases} \tag{25}$$

where

$$\bar{\beta}_j = \langle r + X_i\beta_i^\ell, X_j\rangle = \langle r, X_j\rangle + \langle X_i, X_j\rangle\beta_i^\ell, \tag{26}$$

and $r = y - X\beta^\ell$. We note that in Algorithm 2, solving (19) in iteration $\ell - 1$ is preceded by a call to Algorithm 1 in iteration $\ell - 2$. The quantities $\langle r, X_j\rangle$ and $\langle X_i, X_j\rangle$ appearing to the right of (26) are already computed during the run time of Algorithm 1. By reusing these two stored quantities, we can compute every $\bar{\beta}_j$ and consequently v_j^* in $O(1)$ arithmetic operations.

Furthermore, the following equivalences can be shown to hold

$$\underset{j \in S^c}{\arg\min}\, F_j^* \iff \underset{j \in S^c}{\arg\max}\, |v_j^*| \tag{27}$$

$$F_k^* < F(\beta^l) \iff |v_k^*| > |\beta_k^l|. \tag{28}$$

Thus, we can avoid the computation of the objective F_j^* in (22) and replace (23) with $k \leftarrow \arg\max_{j \in S^c} |v_j^*|$. Furthermore, we can replace $F_k^* < F(\beta^*)$ (before equation (24)) with $|v_k^*| > |\beta_i^l|$. We summarize these changes in Subroutine 3, which is the efficient counterpart of Subroutine 2. Clearly, Subroutine 3 has a cost of $O\Big((p - \|\beta^l\|_0)\|\beta^l\|_0\Big)$ operations.

Subroutine 3: Efficient Implementation of Problem (19) with $k = 1$.

$S \leftarrow \text{Supp}(\beta^\ell)$
for $i \in S$ **do**
  for $j \in S^c$ **do**
    Compute v_j^* in $O(1)$ using (25)
  $\vartheta \leftarrow \arg\max_{j \in S^c} |v_j^*|$
  if $|v_k^*| > |\beta_k^\ell|$ **then**
    $\widehat{\beta} \leftarrow \beta^\ell - e_i \beta_i^\ell + e_\vartheta v_\vartheta^*$
    BREAK

Remark 5. *Since CD-PSI(1) (Algorithm 2 with $k = 1$) is computationally efficient, in Algorithm 2 (with $k > 1$), CD-PSI(1) may be used to replace Algorithm 1. In our numerical experiments, this is found to work well in terms of lower run times and also in obtaining higher-quality solutions (in terms of objective values). This modification also guarantees convergence to a PSI(k) minimum (as the proof of Theorem 4 still applies to this modified version).*

4.2 Algorithm for FSI minima

To obtain a FSI(k) minimum, Problem (19) needs to be modified – we replace optimization w.r.t the variable $U^{S_2}\beta$ by that of $U^{(S\setminus S_1)\cup S_2}\beta$. This leads to the following problem:

$$\min_{\beta, S_1, S_2} \quad F(\beta^\ell - U^{S_1}\beta^\ell + U^{(S\setminus S_1)\cup S_2}\beta) \quad \text{s.t.} \quad S_1 \subseteq S,\ S_2 \subseteq S^c, |S_1| \le k,\ |S_2| \le k, \tag{29}$$

where, $S = \text{Supp}(\beta^l)$. Similarly, Algorithm 2 gets modified by considering Problem (29) instead of Problem (19). By the same argument used in the proof of Theorem 4, this modification guarantees that Algorithm 2 converges in a finite number of iterations to a FSI(k) minimum.

Problem (29) can be expressed as a MIQO problem. To this end, Problem (20) needs to be modified in lines (20c), (20b) and (20f) with the following constraints:

$$\begin{aligned}
&\theta = \beta^\ell - \textstyle\sum_{i\in S} e_i\beta_i^\ell(1 - z_i) + \sum_{i\in S^C} e_i\beta_i + \sum_{i\in S} e_i\beta_i \\
&-\mathcal{M}z_i \le \beta_i \le \mathcal{M}z_i, \qquad i \in S \cup S^c = [p] \\
&\beta_i \in \Re, \qquad i \in S \cup S^c = [p]
\end{aligned}$$

With the above modification, Problem (29) can be expressed as the following MIQO problem:

$$\min_{\theta, z} \quad f(\theta) + \lambda_0 \sum_{i\in[p]} z_i \tag{30a}$$

$$\text{s.t.} \quad -\mathcal{M}z_i \le \theta_i \le \mathcal{M}z_i, \quad \forall i \in [p] \tag{30b}$$

$$\sum_{i\in S^c} z_i \le k \tag{30c}$$

$$\sum_{i\in S} z_i \ge |S| - k \tag{30d}$$

$$z_i \in \{0, 1\}, \quad \forall i \in [p] \tag{30e}$$

In words, the above formulation removes variables indexed by S_1 from the current support S, adds variables corresponding to indices $S_2 \in S^c$, and then optimizes over the new support $(S \setminus S_1) \cup S_2$ – the selection of coordinates is expressed via the inequality constraints on the binary variables, appearing in Problem (30). Problem (30) has p binary variables and p continuous variables.

Remark 6. *Formulation* (30) *has a larger search space compared to formulation* (20) *of PSI minima, due to the additional number of continuous variables. This leads to increased running times compared to formulation* (20)*. However, we note that this formulation can be solved significantly faster than an MIO formulation for Problem* (3) *(for the same reasons discussed in Remark 4).*

In Section 6.5, we present experiments where we compare the quality of FSI(k) minima, for different values of k, to the other classes of minima.

5 Efficient Computation of the Regularization Path

We designed `L0Learn` [8]: an extensible C++ toolkit with an R interface that implements all the algorithms discussed in this paper. The toolkit achieves lower running times[9] compared to other popular sparse learning toolkits (e.g., `glmnet` and `ncvreg`) by utilizing a series of computational tricks such as continuation, an adaptive choice of the grid of tuning parameters, active set updates, (partially) greedy cyclic ordering of coordinates, correlation screening, and a careful accounting of floating point operations exploiting the least-squares loss and sparsity in β. We emphasize that in our experience, the aforementioned computational heuristics are found to play a critical role–they together influence the quality of solutions obtained and also result in fast run times. We note that the toolkit utilizes the fast linear algebra library Armadillo [31] which makes direct calls to the BLAS (Basic Linear Algebra Subprograms), leading to significant speedups for linear algebra operations. Below we provide a more detailed account of the strategies mentioned above.

Continuation: From a statistical viewpoint, it is desirable to obtain solutions to Problem (3) for a grid of regularization parameters $\{\lambda_0^i\}_1^m$, for every choice of λ_1, λ_2. If the λ_0 values are sorted as: $\lambda_0^1 > \lambda_0^2 > \cdots > \lambda_0^m$, we use the solution obtained from λ_0^i to initialize the algorithms (both Algorithms 1 and 2) for λ_0^{i+1}. This helps in speeding up convergence of the algorithm and also *encourages* the algorithm to avoid low-quality stationary solutions. We note that we do not use continuation across λ_1, λ_2; and restrict ourselves to continuation across λ_0.

Adaptive Selection of Tuning Parameters: While using continuation, the selection of a good grid of λ_0 values is very important. The chosen grid may depend upon the choice of λ_1, λ_2. If successive values of λ_0^i are far away, one might miss good solutions. On the other hand, if two values of λ_0^i are too close, they may lead to identical solutions (due to the combinatorial nature of the problem). To avoid this problem, we derive conditions (which to our knowledge are novel) on the choice of λ_0 values which ensure that Algorithm 1 leads to different solutions when using continuation. To this end, we present the following lemma, wherein we assume that λ_1, λ_2 are a-priori fixed.

Lemma 12. *Suppose* $\beta^{(i)}$ *is the output of Algorithm 1 with* $\lambda_0 = \lambda_0^i$*. Let* $S = Supp(\beta^{(i)})$*,* $r =$

[8] `https://github.com/hazimehh/L0Learn`

[9] We also note that Problem (3) is expected to lead to solutions that are more sparse (fewer non-zeros) than those available via Lasso and MCP penalized regression. This also contributes towards reduced run times.

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

$$M^i = \frac{1}{2(1+2\lambda_2)} \max_{j \in S^c} \Big(|\langle r, X_j \rangle| - \lambda_1 \Big)^2. \tag{31}$$

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

Proof. Let us consider the case where, $\lambda_0^{i+1} < M^i$. It follows from (31) that:

$$\max_{j \in S^c} \frac{|\ |\langle r, X_j \rangle| - \lambda_1|}{1+2\lambda_2} > \sqrt{\frac{2\lambda_0^{i+1}}{1+2\lambda_2}}, \tag{32}$$

which implies that $\beta^{(i)}$ is not a CW minimum for the given λ_0^{i+1} (see (9)). By Theorem 2, Algorithm 1 converges to a CW minimum. Therefore, Algorithm 1 initialized with $\beta^{(i)}$ leads to $\beta^{(i+1)} \neq \beta^{(i)}$.

We now consider the case where, $\lambda_0^{i+1} \in (M^i, \lambda_0^i]$. Then (31) implies

$$\max_{j \in S^c} \frac{|\ |\langle r, X_j \rangle| - \lambda_1|}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0^{i+1}}{1+2\lambda_2}} \leq \sqrt{\frac{2\lambda_0^{i}}{1+2\lambda_2}}. \tag{33}$$

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

$$|\beta_j^{(i)}| \geq \sqrt{\frac{2\lambda_0^{i}}{1+2\lambda_2}} \geq \sqrt{\frac{2\lambda_0^{i+1}}{1+2\lambda_2}}, \tag{34}$$

where, the second inequality follows from $\lambda_0^{i+1} \leq \lambda_0^i$. The condition $\nabla_S f(\beta_S^{(i)}) = 0$ along with inequalities (33) and (34) imply that $\beta^{(i)}$ is a CW minimum for Problem (3) at $\lambda_0 = \lambda_0^{i+1}$. Therefore, $\beta^{(i)}$ is a fixed point for Algorithm 1. □

Lemma 12 suggests a simple scheme to compute the grid of tuning parameters $\{\lambda_0^i\}$. Suppose we have computed the regularization path up to $\lambda_0 = \lambda_0^i$, then λ_0^{i+1} can be computed as $\lambda_0^{i+1} = \alpha M^i$, where α is a fixed scalar in $(0, 1)$. Moreover, we note that M^i (defined in (31)) can be computed without explicitly calculating $\langle r, X_i \rangle$ for every $i \in S^c$, as these dot products can be maintained in memory while running Algorithm 1 with $\lambda_0 = \lambda_0^i$. Therefore, computing M^i, and consequently λ_0^{i+1}, requires only $O(|S^c|)$ operations.

(Partially) Greedy Cyclic Order: Suppose Algorithm 1 is initialized with a solution β^0 and let $r^0 = y - X\beta^0$. Before running Algorithm 1, we sort the coordinates based on arranging the quantities $|\langle r^0, X_i \rangle|$ for $i \in [p]$ in descending order[10]. We note that this ordering is performed *once* before the start of Algorithm 1, making this different from greedy CD which requires finding the maximum of $|\langle r^0, X_i \rangle|$ after every coordinate update.

This ordering of the coordinates encourages Algorithm 1 to give high priority to coordinates that are highly correlated (in absolute value) with the residuals. Furthermore, we note that when using continuation, the quantities $|\langle r^0, X_i \rangle|$ can be maintained in memory by Algorithm 1 when

[10]Since the columns of X have unit ℓ_2-norm, updating index $\arg\max_i |\langle r^0, X_i \rangle|$ will lead to the maximal decrease in the objective function.

computing the solution at the previous value of λ_0 (in the grid). Instead of fully sorting the p values $|\langle r^0, X_i\rangle|, i \in [p]$; we have observed that it is computationally beneficial to perform a *partial sorting*, in which only the top t coordinates are sorted, while the rest maintain their initial order. Partial sorting can be done in $O\Big(p\log(t)\Big)$ operations using a heap-based implementation. In our experiments, we found that setting t to 5% of p leads to results similar to that of full sorting, with much reduced computation time. We call this scheme a (partially) greedy cyclic order.

In Section 6.2, we compare Algorithm 1 with the aforementioned (partially) greedy cyclic order, the vanilla version of Algorithm 1 (where we cycle across the p coordinates in an a-priori fixed order), and random CD [29]. Our findings seem to indicate that the (partially) greedy cycling rule proposed herein, has a significant advantage both in terms of running time and optimization performance.

Correlation Screening: When using continuation, we perform screening [33] by restricting the updates of Algorithm 1 to the support of the warm start in addition to a small portion (e.g., 5%) of other coordinates that are highly correlated with the current residuals — these highly correlated coordinates are readily available as a byproduct of the (partially) greedy cyclic ordering rule, described above. After convergence on the screened support, we check if any of the coordinates outside the support violates the conditions of a CW minimum. If there is a violation, we rerun the algorithm from the current solution. Typically, the solution obtained from the screened set turns out to be a CW minimum, and only one pass is done over all the coordinates—this is often found to reduce the overall running time.

Active Set Updates: Empirically, we observe that the iterates generated by Algorithm 1 can typically achieve support stabilization in less than 10 full cycles[11]. This is further supported by Theorem 2 which guarantees that the support must stabilize in a finite number of iterations. If the support does not change across multiple consecutive full cycles, then we restrict the subsequent updates of Algorithm 1 to the current support. After convergence on the restricted support, we check whether any of the coordinates from outside the support violates the conditions of a CW minimum. If there is a violation, we run the algorithm once again initialized from the current iterate—-this is continued until we reach a CW minimum. This heuristic turns out to be very effective in reducing the computation times, especially when $p \gg n$.

Fast Coordinate Updates: Let $r^k = y - \sum_{j\in[p]} X_j\beta_j^k$ be the residuals corresponding to the current iterate β^k of Algorithm 1. To update coordinate i in iteration $k+1$, we need to compute $\widetilde{\beta}_i = \langle r^k + X_i\beta_i^k, X_i\rangle = \langle r^k, X_i\rangle + \beta_i^k$ before applying the thresholding operator (14). Updating $\widetilde{\beta}_i$ can be done using either one of the following rules that exploit sparsity:

(i) **Residual Updates:** We maintain the residuals r^k throughout the algorithm, and we compute $\widetilde{\beta}_i = \langle r^k, X_i\rangle + \beta_i^k$ using $O(n)$ operations. Once β_i^{k+1} is computed (upon updating the ith coordinate), we update the residuals by: $r^{k+1} \leftarrow r^k + X_i(\beta_i^k - \beta_i^{k+1})$ with cost $O(n)$ operations. Since β^k is sparse, many of the coordinates remain at zero during the algorithm, i.e., $\beta_i^k = \beta_i^{k+1} = 0$ implying that $r^{k+1} = r^k$. Thus, updating the residuals for one full pass costs $O((p-t)n)$, where t is the number of coordinates that stay at 0 during the full cycle — note that $p - t \approx |S|$, where S is the support obtained at the end of the full cycle.

(ii) **Materialization Updates:** We present another method to update $\widetilde{\beta}_i$ that does not require using the precomputed r^k. Note that Algorithm 1, at the start computes $\langle y, X_i\rangle$ for all $i \in [p]$ (this is done only once when using continuation). If a coordinate ℓ enters the support for the

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

first time, we "materialize ℓ" by computing and storing the quantity $\langle X_\ell, X_j \rangle$ for all $j \in [p]$. Thus, in iteration $k+1$, we compute $\widetilde{\beta}_i$ by utilizing the sparsity of the support as follows:

$$\widetilde{\beta}_i = \langle y, X_i \rangle - \sum\nolimits_{j:\beta_j^k \neq 0} \langle X_i, X_j \rangle \beta_j^k. \tag{35}$$

The update in (35) costs $O(\|\beta^k\|_0)$ operations. This cost is smaller than the $O(n)$ cost of computing $\widetilde{\beta}_i$ using rule (i), above; provided $\|\beta^k\|_0 < n$. If $\beta_i^{k+1} \neq 0$ and coordinate i has not been materialized before, then we materialize it with cost $O(np)$ (stemming from dot-product computations). Computing $\widetilde{\beta}_i$ for every $i \in [p]$ requires a cost of $O(|S|p)$, where S is the largest support encountered. In addition, there is a cost of $O(|S|np)$ for computing the dot-products arising from materializing new coordinates. However, these computations can be stored during the algorithm and need not be repeated at every iteration.

Scheme (ii) is useful when the supports encountered by Algorithm 1 are small w.r.t to n. It is also useful for an efficient implementation of the PSI(1) algorithm as it stores the dot products required in (26). However, when the supports encountered by CD are relatively large compared to n (e.g., 10% of n), then Scheme (i) can become significantly faster since the dot product computations can be accelerated using calls to the BLAS library. We recommend keeping an option for both the above schemes and selecting the one that can run faster.

6 Computational Experiments

In this section, we investigate both the optimization and statistical performance of our proposed algorithms and compare them to other popular sparse learning algorithms. For convenience, we provide a roadmap of this section.

Section 6.2 presents comparisons among our proposed algorithms and other variants of CD and IHT. Section 6.3 empirically studies the statistical properties of estimators available from our proposed algorithms versus others with varying sample sizes. Section 6.4 studies phase transitions as the SNR varies. Section 6.5 performs an in-depth investigation among the PSI(k)/FSI(k) algorithms, for different values of k. Section 6.6 presents empirical studies including timing comparisons for some large-scale instances, including real-datasets.

6.1 Experimental Setup

Data generation: We consider a series of experiments on synthetic datasets for a wide range of problem sizes and designs. We generate a multivariate Gaussian data matrix $X_{n\times p} \sim \text{MVN}(0, \Sigma)$. We generate $\beta^\dagger$ with $k^\dagger$ non-zero entries with $\beta_i^\dagger \in \{0, 1\}$ and $\beta_i^\dagger = 1$ for $k^\dagger$ equi-spaced indices in $[p]$. We then generate the response vector $y = X\beta^\dagger + \epsilon$, where $\epsilon_i \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$. We define the signal-to-noise ratio (SNR) by $\text{SNR} = \frac{\beta^{\dagger T}\Sigma\beta^\dagger}{\sigma^2}$. We consider the following instances of $\Sigma := ((\sigma_{ij}))$:

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

We select the tuning parameters by minimizing the prediction error on a separate validation set, which is generated under the fixed design setting as $y' = X\beta^\dagger + \epsilon'$, where, $\epsilon_i' \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$.

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

- **Lasso**: In the figures we denote it by "L1".
- **Relaxed Lasso**: This is the version of relaxed Lasso considered in [17]. Given the Lasso solution β^{lasso}, the relaxed Lasso solution is defined as $\beta^{\text{relaxed}} = \gamma\beta^{\text{lasso}} + (1-\gamma)\beta^{\text{LS}}$ where the non-zero components of β^{LS} are given by the least squares solution on the support of β^{lasso} and $\gamma \in [0,1]$ is a tuning parameter. We use our own implementation for relaxed lasso and denote it in the experiments by "L1Relaxed".
- **MCP**: This is the MCP penalty of [36]. We use the coordinate descent-based implementation provided by the R package "`ncvreg`" [7].
- **Forward Stepwise Selection**: We use the implementation provided by [17], and in the experiments we denote it by "FStepwise".
- **IHT**: We use our implementation of IHT, and we select a constant step size, which is equal to the reciprocal of the maximum eigenvalue of X^TX.

For all the methods involving one tuning parameter, we tune over a grid of 100 parameter values, except for forward stepwise selection which we allow to run for up to 250 steps. For the methods with two parameters, we tune over a 2-dimensional grid of 100×100 values. For our algorithms, the tuning parameter λ_0 is generated as per Section 5. For the (L_0L_2) penalty, we sweep λ_2 between 0.0001 and 10 if SNR ≥ 1 and 100 if SNR ≤ 1. For the (L_0L_1) penalty, we sweep λ_1 from $\|X^Ty\|_\infty$ down to $0.0001 \times \|X^Ty\|_\infty$. For Lasso, we sweep λ_1 from $\|X^Ty\|_\infty$ down to $0.0001 \times \|X^Ty\|_\infty$. For Relaxed Lasso, we use the same values of λ_1 as for the Lasso and we sweep γ between 0 and 1. For MCP, the range of the first parameter λ is chosen by `ncvreg`, and we sweep the second parameter γ between 1.5 and 25.

Performance Measures: We use the following metrics to evaluate the quality of a solution ($\widehat{\beta}$, say) obtained by a method.

- **Prediction Error**: This is the same measure used in [5] and is defined by

$$\text{Prediction Error} = \|X\widehat{\beta} - X\beta^\dagger\|^2 / \|X\beta^\dagger\|^2.$$

  The prediction error of a perfect model is 0 and that of the null model ($\beta = 0$) is 1.
- L_∞ **Norm**: This the L_∞-norm of the estimation error, i.e., $\|\widehat{\beta} - \beta^\dagger\|_\infty$.
- **Full support recovery**: We study if the support of $\beta^\dagger$ is completely recovered by β, i.e., $1[\text{Supp}(\beta^\dagger) = \text{Supp}(\widehat{\beta})]$, where, $1[\cdot]$ is the indicator function – we look at the average value of this quantity across multiple replications, leading to an estimate for probability of full support recovery.
- **True Positives**: The number of common elements between the supports of $\widehat{\beta}$ and $\beta^\dagger$.
- **False Positives**: The number of elements in the support of $\widehat{\beta}$ but not in the support of $\beta^\dagger$.
- **Support Size**: The number of non-zeros in $\widehat{\beta}$.
- **Objective**: The value of the objective function $F(\widehat{\beta})$.

6.2 Comparison among CD variants and IHT: Optimization performance

We investigate the optimization performances of the different algorithms. We study the objective values of solutions obtained by IHT and the following variants of CD:

- **Cyclic CD**: This is Algorithm 1 with default cyclic order.
- **Random CD**: This is a randomized version of CD where, the coordinates to be updated are chosen uniformly at random from $[p]$. This is the version considered in [29].
- **Greedy Cyclic CD**: This is our proposed Algorithm 1 with a partially greedy cyclic ordering of coordinates, described in Section 5.

We generated a dataset with Exponential Correlation, $\rho = 0.5$, $n = 500$, $p = 2000$, SNR $= 10$, and a support size $k^\dagger = 100$. We generated 50 random initializations each with a support size of 100, where the non-zero indices are selected uniformly at random from the range 1 to p and assigned values that are drawn from Uniform$(0, 1)$. For every initial solution, we ran Cyclic CD, Greedy Cyclic CD, and IHT and recorded the value of the objective function of the solution along with the number of iterations till convergence. For Random CD, we ran the algorithm 10 times for every initialization and averaged the objective values and number of iterations. For all the algorithms above, we declare convergence when the relative change in the objective is $< 10^{-7}$. Figure 1 presents the results. Figure 1 shows that the objective values resulting from Greedy Cyclic CD

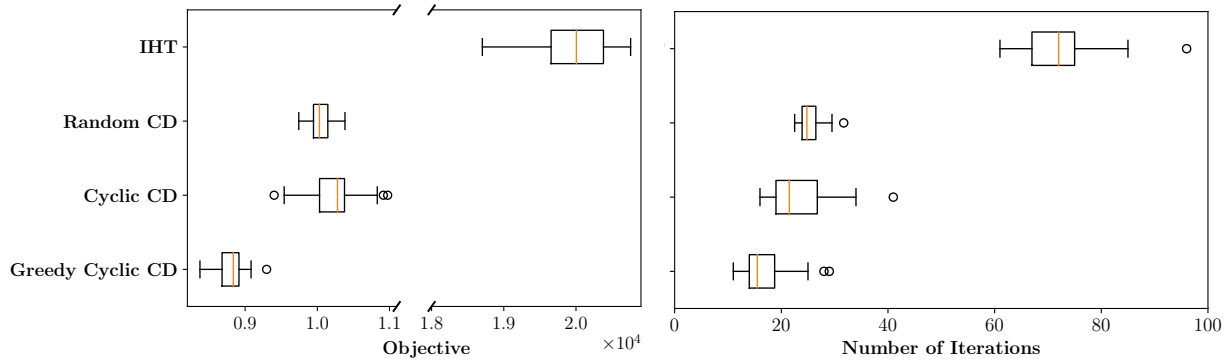

Figure 1: Box plots showing the distribution of the objective values and the number of iterations (here, one full pass over all p coordinates is defined as one iteration) till convergence for different variants of CD and IHT, for each algorithm we used 50 random initializations (as described in the text). The ticks of the box plots represent 1.5 times the interquartile range.

are significantly lower than the other methods; on average we have roughly a 12% improvement in the objective from Random CD and 55% improvement over IHT. This finding can be partially explained in the light of our discussion in Section 2.4 where, we observed that the Lipschitz constant L controls the quality of the solutions returned by IHT. In this high-dimensional setting, $L \approx 11$ which is far from 1, and thus IHT can get stuck in relatively weak local minima. The number of iterations till convergence is also in favor of Greedy Cyclic CD, which requires roughly 28% less iterations than Random CD and 75% less iterations than IHT.

6.3 Statistical Performance for Varying Number of Samples

In this section, we study how the different statistical metrics change with the number of samples, with other factors $(p, k^\dagger, \text{SNR}, \Sigma)$ remaining the same. We consider Algorithm 1 and CD-PSI(1) for

the (L_0), (L_0L_1), and (L_0L_2) problems; in addition to Lasso, Relaxed Lasso, MCP, and Forward Stepwise selection. In Experiments 1 and 2 (below), we sweep n between 100 and 1000 in increments of 100. For every value of n, we generated 20 random training and validation datasets having Exponential Correlation, $p = 1000$, $k^\dagger = 25$, and SNR=10.

6.3.1 Experiment 1: High Correlation

Here we select $\rho = 0.9$ – this is a difficult problem due to the high correlations among features in the sample. Figure 2 summarizes the results. In the top panel of Figure 2 we show Algorithm 1 for (L_0), CD-PSI(1) for (L_0L_2), and other competing algorithms. In the bottom panel, we present a detailed comparison among the results available from Algorithm 1 and CD-PSI(1) for all the cases (L_0), (L_0L_1), and (L_0L_2).

From the top panel (Figure 2), we can see that CD-PSI(1) (L_0L_2) (i.e., algorithm CD-PSI(1) applied to the (L_0L_2) problem) achieves the best performance in terms of all the considered measures and for all values of n. The probability of full recovery for CD-PSI(1) (L_0L_2) increases steeply when n is in the range of 200 and 300 and achieves a probability of 1 at $n = 500$ (which is half of p). Note that for all values of n, Lasso and Relaxed Lasso never achieve full support recovery—and the corresponding lines are aligned with the horizontal axis. Moreover, MCP and FStepwise have a probability less than 0.5 even when $n = p = 1000$ – suggesting that they fail to do correct support recovery in this regime. A similar phenomenon occurs for the prediction error and the L_∞ norm where CD-PSI(1) (L_0L_2) clearly dominates.

The bottom figure shows that CD-PSI(1) (L_0), CD-PSI(1) (L_0L_1), and CD-PSI(1) (L_0L_2) behave similarly; and they significantly outperform their cousins that perform no swaps. It is clear that the local combinatorial optimization schemes introduced herein, have an edge in performance. This finding suggests that in the presence of highly correlated features, CD can get easily stuck in poor-quality solutions, and hence a combinatorial searching scheme plays an important role in guiding it towards better solutions. Swaps seem to help achieve the true underlying solution, even when the underlying statistical problem becomes relatively difficult–with small values of n.

6.3.2 Experiment 2: Mild Correlation

In this experiment we keep exactly the same setup as the previous experiment, but we reduce the correlation parameter ρ to 0.5. In Figure 3, we show the results for Algorithm 1 for (L_0), CD-PSI(1) for (L_0L_2), and the other competing methods. We note that the results for the rest of our methods have the same profiles as Algorithm 1 (L_0) and CD-PSI(1) (L_0L_2), however, we do not include the plots for space constraints. Intuitively, this setup is relatively easier from a statistical perspective, when compared to Experiment 1 where $\rho = 0.9$. Thus, we expect all the methods to perform better (overall) and display a phase-transition at a smaller sample size (compared to Experiment 1). Indeed, as shown in Figure 3, Algorithm 1 (L_0) and CD-PSI(1) (L_0L_2) have roughly the same profiles, and they outperform the other methods; they fully recover the true support using only 200 samples. The swap variants of our methods in this case do not seem to lead to significant improvements over the non-swap variants, and this can be explained by our hypothesis: when the statistical problem is easy, Algorithm 1 is successful in recovering the true underlying signal – swaps are not necessary. MCP and FStepwise also exhibit good performance, but their transition occurs for much larger values of n and MCP does not seem to be robust. Lasso in this case never

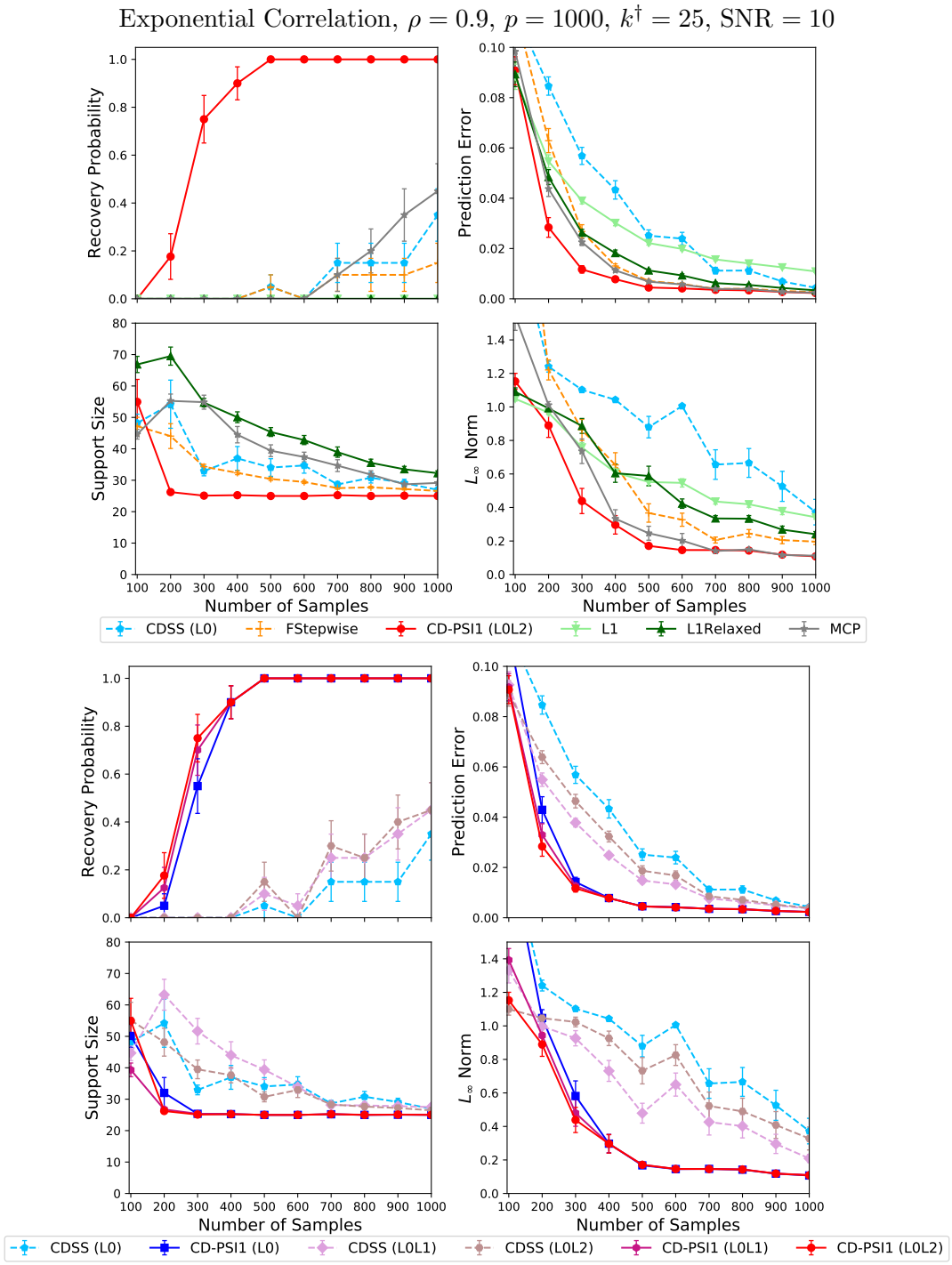

Figure 2: Performance measures as the number of samples n varies between 100 and 1000. The top figure compares two of our methods (a) CDSS(L0) (i.e., Algorithm 1 for the (L_0) problem), (b) CD-PSI(1) for the (L_0L_2) problem, and other state-of-the-art algorithms. The bottom figure compares Algorithm 1 and CD-PSI(1) for all three problems.

recovers the true support, and this property is inherited by Relaxed Lasso which requires at least 900 samples for full support recovery.

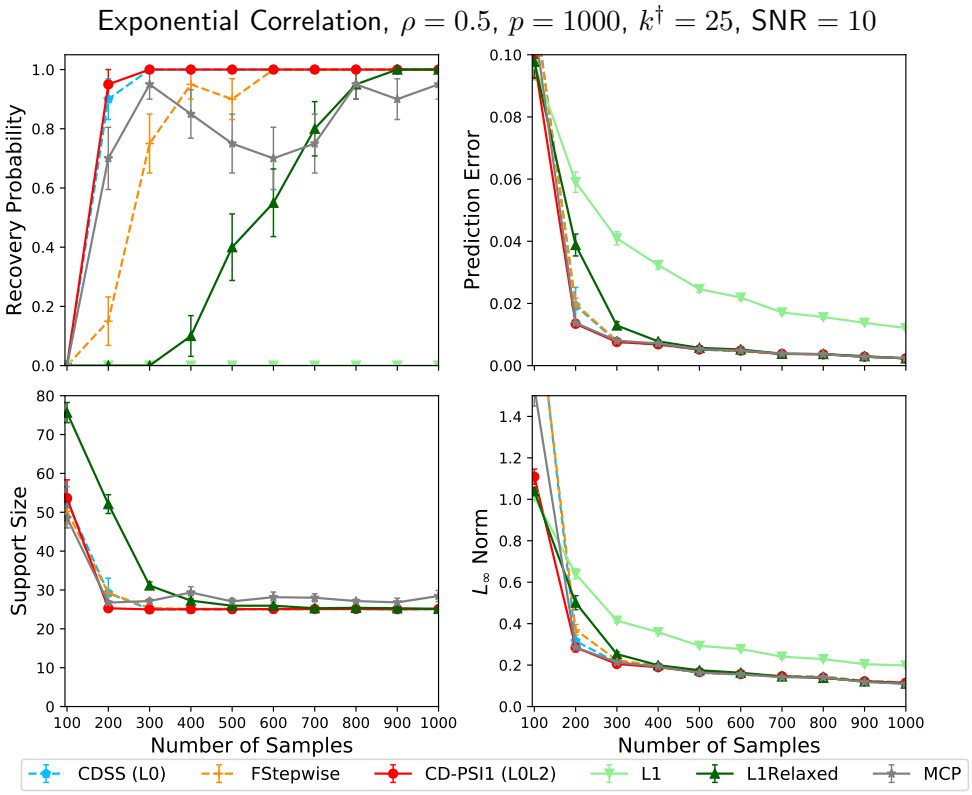

Figure 3: Performance measures as the number of samples n varies between 100 and 1000. The figure compares two of our methods (a) CDSS(L0) (i.e., Algorithm 1 for the (L_0) problem), (b) CD-PSI(1) for the (L_0L_2) problem, and other state-of-the-art algorithms.

6.4 Statistical Performance for Varying SNR

We performed two experiments to study the effect of varying the SNR on the different performance measures. In every experiment, we swept the SNR between 0.1 and 100 for 10 values equi-spaced on the logarithmic scale. For every SNR value, we generated 20 random datasets over which we averaged the results. We observed that for low SNR values, ridge regression seems to work extremely well in terms of prediction performance. Thus, we include ridge regression (L2) in our results.

6.4.1 Experiment 1: Constant Correlation

We generated datasets with constant correlation, $\rho = 0.4$, $n = 1000$, $p = 2000$, and $k = 50$. We report the results for Algorithm 1 (L_0), CD-PSI(1) (L_0L_2), and all the other state-of-the-art algorithms in Figure 4.

Figure 5 suggests that: CD-PSI(1) (L_0L_2) clearly dominates all the other methods in terms of full recovery, prediction error, and L_∞ norm for the whole range of SNR. For low SNR, its prediction error is close to that of L2 which is observed to be very effective in low SNR settings. At SNR $= 100$, CD-PSI(1) (L_0L_2) fully recovers the support while Algorithm 1 (L_0) has a recovery probability $\sim$ 0.4. However, none of the other considered methods does full recovery even for high SNR. By

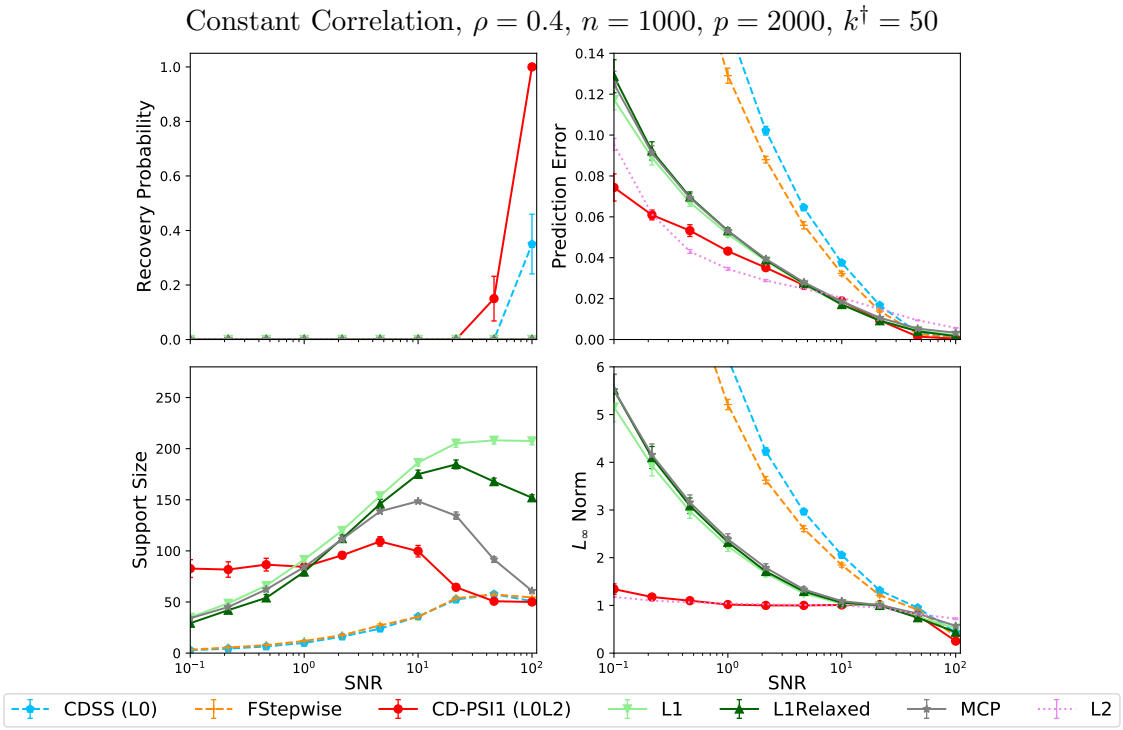

Figure 4: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.1 and 100. The figure compares two of our methods (a) CDSS(L0) (i.e., Algorithm 1 for the (L_0) problem), (b) CD-PSI(1) for the (L_0L_2) problem, and other state-of-the-art algorithms.

inspecting the L_∞ norm, we can see that Lasso, Relaxed Lasso, and MCP exhibit very similar behavior, although they are optimizing for different objectives.

6.4.2 Experiment 2: Exponential Correlation

We generated datasets having exponential correlation with $\rho = 0.5$, $n = 1000$, $p = 5000$, and $k^\dagger = 50$. We report the results for Algorithm 1 (L_0), CD-PSI(1) (L_0L_2), and all the other competing algorithms in Figure 5.

It seems that the optimization tasks in this experiment are relatively easier than the experiment in Section 6.4.1, as the correlation between the variables decays exponentially. Thus, we observe less significant differences among the algorithms, when compared to the first experiment – see Figure 5. CD-PSI(1) (L_0L_2) again dominates with respect to all measures and for all SNR values. Also Algorithm 1 (L_0) performs better than FStepwise, especially in terms of selection probability. We note that even in this relatively easy case, Lasso never fully recovers the support. MCP also seems to suffer in terms of full recovery.

6.5 Comparisons among PSI(k)/FSI(k)

Here, we examine the differences among the various classes of minima introduced in the paper: CW, PSI(k) and FSI(k) minima. To understand the differences, we consider a relatively difficult setting with Constant Correlation where $\rho = 0.9$, $n = 250$, $p = 1000$, and the support size is $k^\dagger = 25$. We

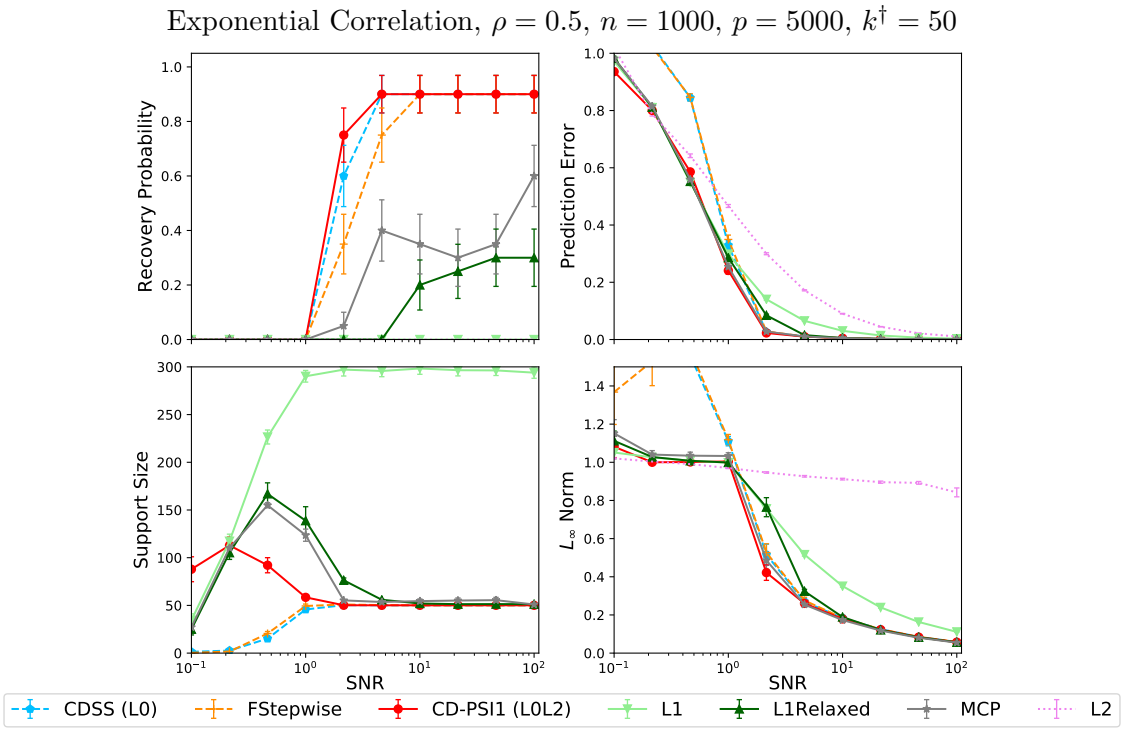

Figure 5: Performance measures as the signal-to-noise ratio (SNR) is varied between 0.1 and 100. The figure compares two of our methods (a) CDSS(L0) (i.e., Algorithm 1 for the (L_0) problem), (b) CD-PSI(1) for the (L_0L_2) problem, and other state-of-the-art algorithms.

set SNR= 300 to allow for full support recovery. We generated 10 random training datasets under this setting and ran Algorithm 1 and the PSI and FSI variants of Algorithm 2 for $k \in \{1, 2, 5\}$ using the (L_0) penalty. All algorithms were initialized with a vector of zeros. For Algorithm 2 we used Gurobi (v7.5) to solve the MIQO subproblems (20) and (30) when $k > 1$.

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

In Figure 7, we show the evolution of solutions when running the FSI(5) variant of Algorithm 2 for the same dataset. The algorithm starts from a CW minimum and iterates between running CD-PSI(1) and finding a swap that improves the objective by solving optimization problem (30) using MIO. The PSI(1) minima obtained from running CD-PSI(1) are marked by red circles and the results obtained by solving problem (30) using MIO are denoted by blue squares.

Figure 7 shows that running CD-PSI(1) on top of the solutions obtained by MIO leads to important gains in terms of better objective values, for most of the cases. This also confirms our intuition that

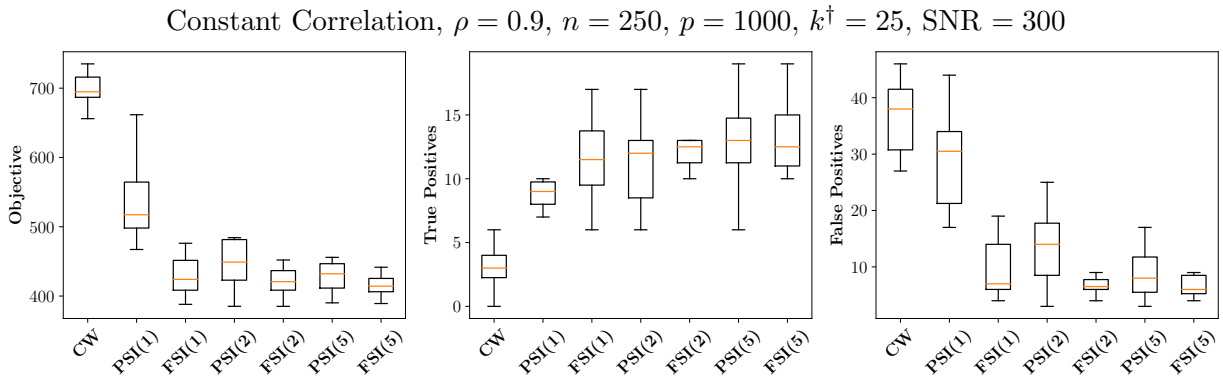

Figure 6: Box plots showing the distribution of objective values, number of true positives, and number of false positives for the different classes of local minima.

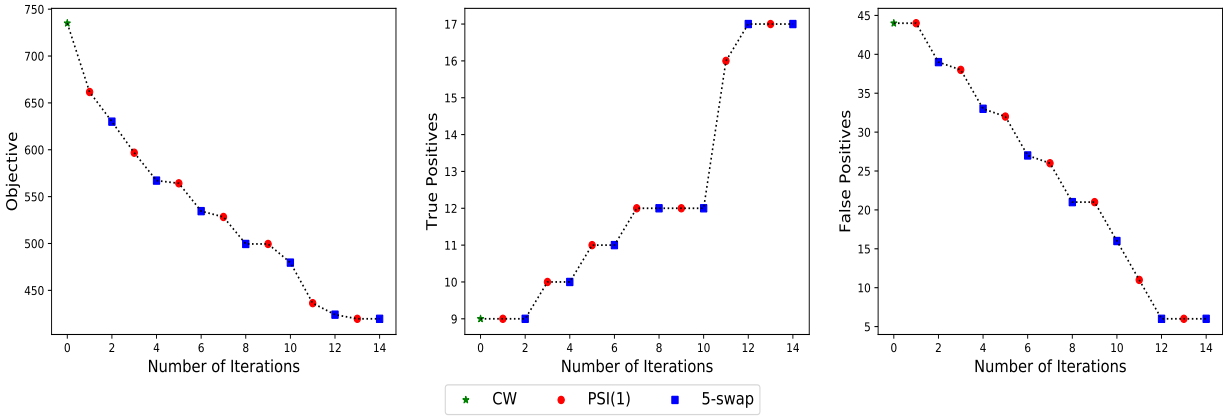

Figure 7: Evolution of solutions during the course of a variant of Algorithm 2 where we successively run CD-PSI(1) and solve combinatorial problem (30) to generate an FSI(5) minimum.

MIO can lead to solutions that are not available via PSI(1). From the plot of true positives and false positives, we can see that CD-PSI(1) improves the solution by increasing the true positives whereas MIO improves the solution by removing false positives. This observation confirms the behavior we noticed in Figure 6, where PSI(1) minima were successful in obtaining a good number of true positives, but suffered in terms of false positives.

6.6 Large High-dimensional Experiments

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

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

Every experiment is performed with 10 replications, and the results are averaged. We report the results for Settings 1 and 2 in Table (1). In Table 1, Algorithm 1 with the (L_0L_1) and (L_0L_2) penalties fully recovers the true support and attains the lowest prediction error. None of the other methods were able to do full support recovery; Lasso and Relaxed Lasso capture most of

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

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

the true positives but include a very large number of false positives. MCP comes in the middle between $(L_0L_1)/(L_0L_2)$ and Lasso — it captures all the true positives and adds few more false positives. We also note that in such high SNR settings, we do not expect shrinkage (arising from the L_1/L_2 penalties) to lead to major statistical improvements. Thus, the difference in performance between (L_0) and $(L_0L_1)/(L_0L_2)$ might be explained by the fact that, the continuous regularizers aid Algorithm 1 in getting better solutions for Problem (3).

Timings and out-of-sample performance: We ran Algorithm 1 using our toolkit `L0Learn` and compared the running time and predictive performance versus `glmnet` and `ncvreg`, on a variety of real and synthetic datasets. We note that `L0Learn`, `glmnet` and `ncvreg` are solving different optimization problems—the run times provided herein are meant to demonstrate that a main workhorse for our proposed algorithm is competitive when compared to efficient state-of-the-art implementations for sparse learning. Below we provide some details about the datasets:

- **House Prices**: $p = 104,000$ and $n = 200$. We used a second order polynomial expansion on the popular Boston House Prices dataset [15] to get 104 features. Then, we added random "probes" (aka noisy features) by appending to the data matrix 1000 random permutations of every column. The validation and testing sets have 100 and 206 samples, respectively.
- **Amazon Reviews**: $p = 17,580$ and $n = 2500$. We used the Amazon Grocery and Gourmet Food dataset [19] to predict the helpfulness of every review (based on its text). Specifically, we calculated the helpfulness of every review as the ratio of the number of up votes to that of down votes, and we obtained X by using Scikit-learn's TF-IDF transformer (while removing stopwords). The validation and testing sets have 500 and 1868 samples, respectively. We also created an augmented version of this dataset where we added random probes by appending to the data matrix 9 random permutations of every column to get $p = 174,755$.
- **US Census**: $p = 55,537$ and $n = 5000$. We used 37 features extracted from the 2016 US Census Planning Database to predict the mail-return rate[12] [9]. We appended the data matrix with 1500 random permutations of every column, and we randomly sampled 15,000 rows, evenly distributed between the training, testing, and validation sets.
- **Gaussian 1M**: $p = 10^6$ and $n = 200$. We generated a synthetic dataset with independent standard normal entries. We set $k^\dagger = 20$, SNR=10, and performed validation and testing as

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

described in Section 6.1.

For all real datasets, we tuned and tested on separate validation and testing sets. The timings were performed on a machine with an i7-4800MQ CPU and 16GB RAM running Ubuntu 16.04 and OpenBLAS 0.2.20. For all methods, we report the training time required to obtain a grid of 100 solutions. For (L_0L_2), (L_0L_1), and MCP, we provide the time for a fixed λ_2, λ_1, and γ, respectively (these parameters have been set to the optimal values obtained via validation set tuning over 10 values of the tuning parameter). Table 2 presents run times for all the four methods — each method computes solutions on a grid of 100-tuning parameters.

The results presented in Table 2 show the following: `L0Learn` is faster than `glmnet` and `ncvreg` on all the considered datasets, e.g., more than twice as fast on the Amazon Reviews dataset. The speed-ups can be attributed to the careful design of `L0Learn` (as described in Section 5) and due to the nature of L_0 regularization which generally selects sparser supports than those obtained by L_1 or MCP regularization. Moreover, `L0Learn`, for both the (L_0L_2) and (L_0L_1) problems, provides much sparser supports and competitive testing MSE compared to the other toolkits. Finally, we note that prediction errors for our methods can be potentially improved by using Algorithm 2, at the cost of slightly increased computation times.

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

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

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

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

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

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

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

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

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

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

Table 2: Training time (in seconds), out-of-sample MSE, and the support sizes for a variety of high-dimensional datasets. The training time is for obtaining a regularization path with 100 solutions.

A Appendix: Proofs and Technical Details

A.1 Proof of Lemma 8

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

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

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

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

Since the support S occurs infinitely often, we consider the infinite sequence $\{\beta^{l,1}\}_{l \in L}$ (i.e., the sequence of intermediate spacer steps where coordinate 1 is updated). By Lemma 7, $\{\beta^{l,1}\}_{l \in L}$ is bounded and therefore, there exists a further subsequence $\{\beta^{l',1}\}_{l' \in L'}$ that converges to a limit point β^* (say). We assume that for every $l' \in L'$ we have $l' \geq N$, which implies that $\beta^{l'-1}$, $\beta^{l',1}, \beta^{l',2}, \ldots, \beta^{l',J}$ all have the support S (this follows from Part 1 of this lemma). Next, we will show that $\beta^{l,2}$ also converges to β^*.

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

$$F(\beta^{l',1}) - F(\beta^{l',2}) = f(\beta_S^{l',1}) - f(\beta_S^{l',2}) \geq \frac{1 + 2\lambda_2}{2} (\beta_2^{l',1} - \beta_2^{l',2})^2 \tag{36}$$

where the last inequality follows by replacing $\beta_2^{l',2}$ with the expression given by the thresholding map in (14) and simplifying. By Lemma 5, $\{F(\beta^k)\}$ converges; so taking the limit as $l' \to \infty$ in (36) we get, $0 \geq \lim_{l' \to \infty} (\beta_2^{l',1} - \beta_2^{l',2})^2$; i.e., $\beta_2^{l',1} - \beta_2^{l',2} \to 0$ as $l' \to \infty$. Since $\beta^{l',1} \to \beta^*$, we conclude that $\beta_2^{l',2} \to \beta_2^*$. Therefore, $\beta^{l',2} \to \beta^*$. The same argument applies to $\beta^{l',i}$ and $\beta^{l',i+1}$ for every $i \in \{2, 3, \ldots, J-1\}$. Therefore, we conclude that for every $i \in \{1, 2, \ldots, J\}$ we have $\beta^{l',i} \to \beta^*$.

Now, by the definition of the spacer step, for every $i \in [J]$ and every $l' \in L'$; we have:

$$\beta_i^{l',i} = T(\widetilde{\beta}_i, 0, \lambda_1, \lambda_2) = \text{sign}(\widetilde{\beta}_i^{l'}) \frac{|\widetilde{\beta}_i^{l'}| - \lambda_1}{1 + 2\lambda_2} \tag{37}$$

where, $\widetilde{\beta}_i^{l'} = \langle y - \sum_{j \neq i} X_j \beta_j^{l',i}, X_i \rangle$ and $|\widetilde{\beta}_i^{l'}| \geq \lambda_1$. Taking $l' \to \infty$, we get $\widetilde{\beta}_i^* = \lim_{l' \to \infty} \widetilde{\beta}_i^{l'} = \langle y - \sum_{j \neq i} X_j \beta_j^*, X_i \rangle$ where $|\widetilde{\beta}_i^*| \geq \lambda_1$. Therefore, for every $i \in [J]$, we have

$$\beta_i^* = \lim_{l' \to \infty} \beta_i^{l',i} = \begin{cases} \text{sign}(\widetilde{\beta}_i^*) \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1 + 2\lambda_2} & \text{if } \lambda_1 > 0 \\ \frac{\widetilde{\beta}_i^*}{1 + 2\lambda_2} & \text{if } \lambda_1 = 0 \end{cases} \tag{38}$$

where we used the continuity of the sign function at $\widetilde{\beta}_i^*$ (note, $|\widetilde{\beta}_i^*| \geq \lambda_1$) in the case of $\lambda_1 > 0$. Therefore, using characterization (6), we observe that β^* is a stationary solution. Finally, Lemma 6 and Assumption 1 for the (L_0) and (L_0L_1) problems imply that $\beta_S \mapsto f(\beta_S)$ is strongly convex and has a unique minimizer — hence β^* is unique.

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

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

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

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

From Lemma 5, we have $\lim_{l' \to \infty} F(\beta^{l'}) = \lim_{k'' \to \infty} F(\beta^{k''})$. This implies $f(\beta^*) = f(\widehat{\beta})$ and in particular, $f(\beta_S^*) = f(\widehat{\beta}_S)$ (since the supports of both limits points are a subset of S). However, by Part 2, we know that β_S^* is the unique minimizer of $\min_{\beta_S} f(\beta_S)$—which leads to a contradiction. Hence, we conclude that $\beta^{k'} \to \beta^*$ as $k' \to \infty$.

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

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

A.2 Proof of Lemma 10

Proof. We first derive an useful expression for $B_j^{(2)}$, which will help us establish parts 1 and 2 of this lemma. By Lemma 5, $F(\beta^k)$ converges to a finite nonnegative limit F^*. Lemmas 8 and 9 imply

that there is a subsequence $\{\beta^{k'}\}_{k' \in K'}$ that converges to $B^{(1)}$ and satisfies $\text{Supp}(\beta^{k'}) = S_1$ for every k'. As $k \to \infty$, we get: $F^* = f(B^{(1)}) + |S_1| = F(B^{(1)})$. Similarly, for $B^{(2)}$ we have $F^* = F(B^{(2)})$. Therefore, $F(B^{(1)}) = F(B^{(2)})$, which is equivalent to

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

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

$$f(B^{(1)}_{S_1}) - f(B^{(2)}_{S_2}) = \lambda_0. \tag{39}$$

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

$$\begin{aligned}
f(B^{(2)}_{S_2}) &= \frac{1}{2}\|y - X_{S_2} B^{(2)}_{S_2}\|^2 + \lambda_1 \|B^{(2)}_{S_2}\|_1 + \lambda_2 \|B^{(2)}_{S_2}\|_2^2 \\
&= \frac{1}{2}\|y - X_{S_1} B^{(2)}_{S_1} - X_j B^{(2)}_j\|^2 + \lambda_1 \|B^{(2)}_{S_1}\|_1 + \lambda_1 |B^{(2)}_j| + \lambda_2 \|B^{(2)}_{S_1}\|_2^2 + \lambda_2 (B^{(2)}_j)^2 \\
&= \Big(\frac{1}{2}\|y - X_{S_1} B^{(2)}_{S_1}\|^2 + \lambda_1 \|B^{(2)}_{S_1}\|_1 + \lambda_2 \|B^{(2)}_{S_1}\|_2^2\Big) \\
&- \langle y - X_{S_1} B^{(2)}_{S_1}, X_j \rangle B^{(2)}_j + \frac{1}{2}\|X_j\|^2 {B^{(2)}_j}^2 + \lambda_1 |B^{(2)}_j| + \lambda_2 (B^{(2)}_j)^2 \\
&= f(B^{(2)}_{S_1}) - \langle y - X_{S_1} B^{(2)}_{S_1}, X_j \rangle B^{(2)}_j + \frac{1}{2}\|X_j\|^2 (B^{(2)}_j)^2 + \lambda_1 |B^{(2)}_j| + \lambda_2 (B^{(2)}_j)^2.
\end{aligned} \tag{40}$$

From Lemma 8 we know that $B^{(2)}$ is a stationary solution. Using the characterization of stationary solutions in (6) and rearranging the terms, we get:

$$\begin{aligned}
\langle y - X_{S_1} B^{(2)}_{S_1}, X_j \rangle &= (1 + 2\lambda_2) B^{(2)}_j + \lambda_1 \text{sign}(\langle y - X_{S_1} B^{(2)}_{S_1}, X_j \rangle) \\
|\langle y - X_{S_1} B^{(2)}_{S_1}, X_j \rangle| &> \lambda_1.
\end{aligned} \tag{41}$$

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

$$\langle y - X_{S_1} B^{(2)}_{S_1}, X_j \rangle B^{(2)}_j = (1 + 2\lambda_2)(B^{(2)}_j)^2 + \lambda_1 |B^{(2)}_j|. \tag{42}$$

Plugging in the above expression in the second term on the r.h.s of (40) and using the fact that $\|X_j\|^2 = 1$ we get

$$f(B^{(2)}_{S_2}) = f(B^{(2)}_{S_1}) - \frac{1 + 2\lambda_2}{2}(B^{(2)}_j)^2. \tag{43}$$

Substituting (43) into equation (39) and rearranging terms, we arrive at

$$|B^{(2)}_j| = \sqrt{\frac{2\lambda_0}{1 + 2\lambda_2} + \frac{2}{1 + 2\lambda_2}\Big(f(B^{(2)}_{S_1}) - f(B^{(1)}_{S_1})\Big)}. \tag{44}$$

Part 1.) We consider Part 1, where there exists an $i \in S_1$ such that $\langle X_i, X_j \rangle \neq 0$. By Lemma 8 we have that $B^{(1)}$ is a stationary solution. Thus, $B^{(1)}_{S_1} \in \arg\min_{\beta_{S_1}} f(\beta_{S_1})$, and the following holds

$$f(B^{(1)}_{S_1}) \leq f(B^{(2)}_{S_1}). \tag{45}$$

We will show the inequality above is strict. To this end, suppose that (45) holds with equality. Lemma 9 implies that S_1 appears in the sequence of iterates. But the function $f(\beta_{S_1})$ is strongly convex (this is trivial for (L_0L_2) and holds due to Assumption 1 and Lemma 6 for the (L_0) and (L_0L_1) problems). Thus, $B^{(1)}_{S_1}$ is the unique minimizer of $f(\beta_{S_1})$. Therefore, it must be the case that $B^{(1)}_{S_1} = B^{(2)}_{S_1}$, and in particular $B^{(1)}_i = B^{(2)}_i$. By the characterization of stationary solutions in (6) we have:

$$\begin{aligned} &\operatorname{sign}(\langle y - X_{S_1\setminus\{i\}}B^{(1)}_{S_1\setminus\{i\}}, X_i\rangle)\ \overbrace{\frac{|\langle y - X_{S_1\setminus\{i\}}B^{(1)}_{S_1\setminus\{i\}}, X_i\rangle| - \lambda_1}{1+2\lambda_2}}^{\geq 0} \\ &= \operatorname{sign}(\langle y - X_{S_2\setminus\{i\}}B^{(2)}_{S_2\setminus\{i\}}, X_i\rangle)\ \underbrace{\frac{|\langle y - X_{S_2\setminus\{i\}}B^{(2)}_{S_2\setminus\{i\}}, X_i\rangle| - \lambda_1}{1+2\lambda_2}}_{\geq 0} \end{aligned} \tag{46}$$

Observing that the two sign terms in (46) are equal, we can simplify the above to:

$$\begin{aligned} \langle y - X_{S_1\setminus\{i\}}B^{(1)}_{S_1\setminus\{i\}}, X_i\rangle &= \langle y - X_{S_2\setminus\{i\}}B^{(2)}_{S_2\setminus\{i\}}, X_i\rangle \\ &= \langle y - X_jB^{(2)}_j - X_{S_1\setminus\{i\}}B^{(2)}_{S_1\setminus\{i\}}, X_i\rangle \end{aligned} \tag{47}$$

where, the second line in (47) follows by noting $X_{S_2\setminus\{i\}}B^{(2)}_{S_2\setminus\{i\}} = X_jB^{(2)}_j + X_{S_1\setminus\{i\}}B^{(2)}_{S_1\setminus\{i\}}$. Substituting $B^{(2)}_{S_1} = B^{(1)}_{S_1}$ in (47) and simplifying, we get $\langle X_j, X_i\rangle = 0$, which contradicts the assumption in Part 1. Thus, we have established that inequality (45) is strict. Using this result in (44), we conclude that: $|B^{(2)}_j| > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$.

Part 2.) We now consider the case where $\langle X_i, X_j\rangle = 0$ for all $i \in S_1$. In this case, the optimization problem $\min_{\beta_{S_2}} f(\beta_{S_2})$ separates into optimization w.r.t the variables β_{S_1} and β_j. Note that $B^{(2)}_{S_1}$ and $B^{(1)}_{S_1}$ are both minimizers of $\min_{\beta_{S_1}} f(\beta_{S_1})$; and hence $f(B^{(2)}_{S_1}) = f(B^{(1)}_{S_1})$. Thus, from (44) we get $|B^{(2)}_j| = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. Finally, we note that for any $\beta \in \mathbb{R}^p$ such that $\operatorname{Supp}(\beta) = S_2$, we have that $T(\widetilde{\beta}_j, \lambda_0, \lambda_1, \lambda_2) = B^{(2)}_j$. This completes the proof. □

A.3 Proof of Theorem 2

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

Part 1.) By Lemma 9, there is a subsequence $\{\beta^r\}_{r\in R}$ of $\{\beta^k\}$ which satisfies: $\operatorname{Supp}(\beta^r) = S$ for all r and $\beta^r \to B$ (as $r \to \infty$). By Lemma 8, there exists an integer N such that for every $r \geq N$, if $r+1$ is a spacer step then $\operatorname{Supp}(\beta^{r+1}) = \operatorname{Supp}(\beta^r)$. In what follows, we assume that $r \geq N$. Let j be any element in S. We will show that there exists an integer N_j such that for every $r \geq N_j$, we have $j \in \operatorname{Supp}(\beta^{r+1})$. We show this by contradiction. To this end, let $j \notin \operatorname{Supp}(\beta^{r+1})$ for infinitely many values of r. Hence, there is a further subsequence $\{\beta^{r'}\}_{r'\in R'}$ of $\{\beta^r\}_{r\in R}$ (with $R' \subseteq R$) such that $\operatorname{Supp}(\beta^{r'+1}) = S \setminus \{j\}$. For every $r' \in R'$, note that $r'+1$ is a non-spacer step (since $r' \geq N$). Therefore, applying Lemma 11 with $k = r'$, we get:

$$F(\beta^{r'}) - F(\beta^{r'+1}) \geq \frac{1+2\lambda_2}{2}\left(|\beta^{r'}_j| - \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}\right)^2. \tag{48}$$

Taking $r' \to \infty$ in (48) and using the convergence of $\{F(\beta^k)\}$ (by Lemma 5), we conclude

$$|B_j| = \lim_{r' \to \infty} |\beta_j^{r'}| = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}. \tag{49}$$

Since $\text{Supp}(\beta^{r'+1}) = S \setminus \{j\}$ for every r', Lemma 8 implies that $\{\beta^{r'+1}\}$ converges to a limit point, which we denote by $\widehat{B}$. If $\langle X_i, X_j \rangle = 0$ for all $i \in S \setminus \{j\}$, then Lemma 10 (part 2) implies $j \in \text{Supp}(\beta^{r'+1})$, which contradicts the definition of $\{\beta^{r'}\}_{r' \in R'}$. Thus, it must be the case that there exists an index $i \in S \setminus \{j\}$ such that $\langle X_i, X_j \rangle \neq 0$. Applying Lemma 10 (part 1) to B and $\widehat{B}$ we have that $|B_j| > \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$ – this contradicts (49). Therefore, there exists an integer N_j such that for every $r \geq N_j$, we have $j \in \text{Supp}(\beta^{r+1})$.

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

Part 2.) Finally, we show that the limit of $\{\beta^k\}$ is a CW minimum. To this end, note that the results of Part 1 (above) and Lemma 8 (Parts 3 and 4) imply that β^k converges to the limit B, which satisfies $\text{Supp}(B) = S$, and for every $i \in S$, we have:

$$B_i = \text{sign}(\widetilde{B}_i)\frac{|\widetilde{B}_i| - \lambda_1}{1+2\lambda_2} \quad \text{and} \quad |B_i| \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}. \tag{50}$$

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

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

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

$$\frac{|\widetilde{B}_j| - \lambda_1}{1+2\lambda_2} \leq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}. \tag{51}$$

(50) and (51) together imply that B is a CW minimum (by definition). □

References

[1] A. Beck and Y. C. Eldar. 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.

[2] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013. doi: 10.1137/120887679. URL http://dx.doi.org/10.1137/120887679.

[3] D. Bertsekas. *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific, 2016. ISBN 9781886529052. URL https://books.google.com/books?id=TwOujgEACAAJ.

[4] D. Bertsimas and B. Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *arXiv preprint arXiv:1709.10029*, 2017.

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

[6] T. Blumensath and M. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[7] P. Breheny and J. Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The annals of applied statistics*, 5(1):232, 2011.

[8] P. Bühlmann and S. van-de-Geer. *Statistics for high-dimensional data*. Springer, 2011.

[9] C. Erdman and N. Bates. The us census bureau mail return rate challenge: Crowdsourcing to develop a hard-to-count score. *Statistics*, page 08, 2014.

[10] J. Friedman, T. Hastie, and R. Tibshirani. 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/.

[11] G. Furnival and R. Wilson. Regression by leaps and bounds. *Technometrics*, 16:499–511, 1974.

[12] D. Gamarnik and I. Zadik. High dimensional regression with binary coefficients. estimating squared error and a phase transtition. In *Conference on Learning Theory*, pages 948–953, 2017.

[13] E. Greenshtein and Y. Ritov. Persistence in high-dimensional linear predictor selection and the virtue of overparametrization. *Bernoulli*, 10:971–988, 2004.

[14] E. Greenshtein. Best subset selection, persistence in high-dimensional statistical learning and optimization under ℓ_1 constraint. *The Annals of Statistics*, 34(5):2367–2386, 2006.

[15] D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81 – 102, 1978. ISSN 0095-0696. doi: https://doi.org/10.1016/0095-0696(78)90006-2. URL http://www.sciencedirect.com/science/article/pii/0095069678900062.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Prediction, Inference and Data Mining (Second Edition)*. Springer Verlag, New York, 2009.

[17] T. Hastie, R. Tibshirani, and R. J. Tibshirani. Extended Comparisons of Best Subset Selection, Forward Stepwise Selection, and the Lasso. *ArXiv e-prints*, July 2017.

[18] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, FL, 2015.

[19] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 507–517, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4143-1. doi: 10.1145/2872427.2883037. URL https://doi.org/10.1145/2872427.2883037.

[20] P.-L. Loh and M. J. Wainwright. Support recovery without incoherence: A case for nonconvex regularization. *The Annals of Statistics*, 45(6):2455–2482, 2017.

[21] Z. Lu. Iterative hard thresholding methods for l0 regularized convex cone programming. *Mathematical Programming*, 147(1):125–154, Oct 2014. ISSN 1436-4646. doi: 10.1007/s10107-013-0714-4. URL https://doi.org/10.1007/s10107-013-0714-4.

[22] R. Mazumder and P. Radchenko. The Discrete Dantzig Selector: Estimating sparse linear models via mixed integer linear optimization. *IEEE Transactions on Information Theory*, 63 (5):3053 – 3075, 2017.

[23] R. Mazumder, J. H. Friedman, and T. Hastie. 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.

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

[25] A. Miller. *Subset selection in regression*. CRC Press Washington, 2002.

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

[27] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[28] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Norwell, 2004.

[29] A. Patrascu and I. Necoara. Random coordinate descent methods for ℓ_0 regularized convex optimization. *IEEE Transactions on Automatic Control*, 60(7):1811–1824, July 2015. ISSN 0018-9286. doi: 10.1109/TAC.2015.2390551.

[30] G. Raskutti, M. Wainwright, and B. Yu. Minimax rates of estimation for high-dimensional linear regression over-balls. *Information Theory, IEEE Transactions on*, 57(10):6976–6994, 2011.

[31] C. Sanderson and R. Curtin. Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 2016.

[32] W. Su, M. Bogdan, and E. Candes. False discoveries occur early on the lasso path. *The Annals of Statistics*, 45(5):2133–2150, 2017.

[33] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):245–266, 2012.

[34] R. J. Tibshirani. The lasso problem and uniqueness. *Electronic Journal of Statistics*, 7:1456–1490, 2013.

[35] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494, 2001.

[36] C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

[37] C.-H. Zhang and T. Zhang. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

[38] Y. Zhang, M. J. Wainwright, and M. I. Jordan. Lower bounds on the performance of polynomial-time algorithms for sparse linear regression. In *Conference on Learning Theory*, pages 921–948, 2014.