Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

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

Hussein Hazimeh, Rahul Mazumder

To cite this article:
Hussein Hazimeh, Rahul Mazumder (2020) Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. Operations Research 68(5):1517-1537. https://doi.org/10.1287/opre.2019.1919

informs
http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 68, No. 5, September–October 2020, pp. 1517–1537
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

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

Hussein Hazimeh,[a] **Rahul Mazumder**[a,b]

[a] Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139; [b] MIT Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02142

Contact: hazimeh@mit.edu, https://orcid.org/0000-0003-4501-0678 (HH); rahulmaz@mit.edu, https://orcid.org/0000-0003-1384-9743 (RM)

Received: April 10, 2018
Revised: March 7, 2019
Accepted: June 30, 2019
Published Online in Articles in Advance: August 13, 2020

Subject Classification: statistics; programming: integer: algorithms; programming: nondifferentiable
Area of Review: Optimization

https://doi.org/10.1287/opre.2019.1919

Copyright: © 2020 INFORMS

Abstract. The L_0-regularized least squares problem (a.k.a. *best subsets*) is central to sparse statistical learning and has attracted significant attention across the wider statistics, machine learning, and optimization communities. Recent work has shown that modern mixed integer optimization (MIO) solvers can be used to address small to moderate instances of this problem. In spite of the usefulness of L_0-based estimators and generic MIO solvers, there is a steep computational price to pay when compared with popular sparse learning algorithms (e.g., based on L_1 regularization). In this paper, we aim to push the frontiers of computation for a family of L_0-regularized problems with additional convex penalties. We propose a new hierarchy of necessary optimality conditions for these problems. We develop fast algorithms, based on coordinate descent and local combinatorial optimization, that are guaranteed to converge to solutions satisfying these optimality conditions. From a statistical viewpoint, an interesting story emerges. When the signal strength is high, our combinatorial optimization algorithms have an edge in challenging statistical settings. When the signal is lower, pure L_0 benefits from additional convex regularization. We empirically demonstrate that our family of L_0-based estimators can outperform the state-of-the-art sparse learning algorithms in terms of a combination of prediction, estimation, and variable selection metrics under various regimes (e.g., different signal strengths, feature correlations, number of samples and features). Our new open-source sparse learning toolkit L0Learn (available on CRAN and GitHub) reaches up to a threefold speedup (with p up to 10^6) when compared with competing toolkits such as glmnet and ncvreg.

Funding: The authors acknowledge research funding from the Office of Naval Research [Grants ONR-N000141512342 and ONR-N000141812298 (Young Investigator Award)]; the National Science Foundation [Grant NSF-IIS-1718258]; and the Massachusetts Institute of Technology.
Supplemental Material: The supplementary material is available at https://doi.org/10.1287/opre.2019.1919.

Keywords: interpretable machine learning • sparsity • Lasso • high-dimensional statistics • mixed integer programming • coordinate descent • large-scale computation

1. Introduction

The ongoing surge in high-dimensional data has drawn a lot of attention to sparse learning across several scientific communities. Indeed, sparsity can be very effective in high-dimensional settings as it leads to compact models that can be easier to interpret (Bühlmann and van de Geer 2011, Hastie et al. 2015). We consider the usual linear regression setup with $y = X\beta + \epsilon$, where $y \in \mathbb{R}^n$ is the response, $X \in \mathbb{R}^{n\times p}$ is the model matrix, $\beta \in \mathbb{R}^p$ is the vector of regression coefficients, and $\epsilon \in \mathbb{R}^n$ is a noise vector. We will assume that the columns of X are standardized to have a unit L_2-norm, and we ignore the intercept term to simplify the presentation. Our goal is to estimate β under the assumption that it is sparse (i.e., has few nonzeros)—a common desiderata in the high-dimensional learning framework with $p \gg n$ (Bühlmann and van de Geer 2011, Hastie et al. 2015). A natural and direct way to obtain such a sparse estimator is by minimizing the least squares loss with an L_0-norm[1] penalty on β (Miller 2002). Statistical (optimality) properties of this estimator have been extensively studied (Greenshtein 2006, Raskutti et al. 2011, Zhang and Zhang 2012, Zhang et al. 2014). Many appealing alternative sparsity-inducing estimators have been proposed in the literature based on Lasso (Tibshirani 1996), stepwise regression, continuous nonconvex regularization (Hastie et al. 2015), etc.—each with different operating characteristics. Our focus in this paper is on the algorithmic aspects of L_0-based estimators. Recent work (Hastie et al. 2017, Mazumder et al. 2017) has brought to light an intriguing phenomenon: in low signal-to-noise-ratio (SNR) regimes, the vanilla version of L_0 penalization suffers from overfitting. One way to mitigate this problem is by considering a larger family of estimators that includes (in addition to the L_0 penalty) an L_1- or L_2-norm regularization (Mazumder et al. 2017). In this paper, we consider the following extended family of L_0-based estimators—that is, L_0L_q-regularized regression problems of the form:

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

where $q \in \{1, 2\}$ determines the type of the additional regularization (i.e., L_1 or L_2). The regularization parameter λ_0 controls the number of nonzeros (i.e., selected variables) in $\hat{\beta}$, and λ_q controls the amount of shrinkage induced by L_q regularization. In many regimes (and under suitable choices of λ_0, λ_q), estimators from problem (1) exhibit superior statistical properties (variable selection, prediction, and estimation) compared with computationally friendlier alternatives (e.g., based on Lasso or stepwise regression)—see, for example, Raskutti et al. 2011, Zhang and Zhang 2012, Zhang et al. 2014, Bertsimas et al. 2016, Bertsimas and Van Parys 2017, and Mazumder et al. 2017. In spite of its potential usefulness, problem (1) is NP-hard (Natarajan 1995) and poses computational challenges. Recent work by Bertsimas et al. (2016) has shown that high-quality solutions can be obtained for the cardinality-constrained least squares problem via mixed integer optimization (MIO), in the order of minutes when $p \sim 1000$. However, efficient solvers for the Lasso (e.g., glmnet; Friedman et al. 2010) can address much larger problems within a second. Our goal is to bridge this gap in computation time by developing fast solvers that can obtain high-quality (approximate) solutions to problem (1) for large and challenging instances (e.g., $p \sim 10^6$ and small n). This will allow performing systematic large-scale experiments to gain a deeper understanding of the statistical properties of L_0-based estimators and their differences with the state of the art. Such an understanding is currently limited as a result of computational considerations.

Our approach is based on two complementary algorithms: (i) cyclic coordinate descent (CD) for quickly finding solutions to problem (1) and (ii) novel combinatorial search algorithms, which help improve solutions from (i). In particular, the solutions obtained by (ii) cannot be improved by making small changes to their support. We establish novel convergence guarantees for our algorithms. We also address delicate implementation aspects of our algorithms and provide L0Learn: an open-source and efficient R/C++ toolkit available on CRAN at https://CRAN.R-project.org/package=L0Learn and on GitHub at https://github.com/hazimehh/L0Learn.

1.1. Current Landscape and Related Work

Our main focus is on the computational aspects of problem (1). We contextualize our contribution within the rather large and impressive literature on algorithms for sparse regression—see, for example, Beck and Eldar (2013) and Bertsimas et al. (2016) for an overview. We broadly categorize the main existing algorithms into two categories:

- *Proxy algorithms and heuristics:* Proxy algorithms use a proxy/surrogate to the L_0 norm, for example, L_1-norm or nonconvex penalties such as the minimax concave penalty (MCP) and smoothly clipped absolute deviation (SCAD) (Tibshirani 1996, Fan and Li 2001, Zhang 2010). Fast solvers have been devised for these proxies (e.g., Friedman et al. 2010, Breheny and Huang 2011, Mazumder et al. 2011)—they typically result in good solutions (though not optimal for nonconvex problems). Another approach is to use heuristics to find approximate solutions to problem (1) with $\lambda_q = 0$. Popular methods include (greedy) stepwise regression (Hastie et al. 2015), iterative hard thresholding (IHT) (Blumensath and Davies 2009, Bertsimas et al. 2016), greedy CD (Beck and Eldar 2013), and randomized CD (Patrascu and Necoara 2015).
- *Exact algorithms:* These approaches *exactly* solve an optimization problem involving the L_0 norm. Bertsimas et al. (2016) use MIO to compute near-optimal solutions for least squares with a cardinality constraint for $p \approx 1000$. Bertsimas and Van Parys (2017) propose a cutting plane method for a similar problem, which works well with mild sample correlations and a sufficiently large n. Mazumder and Radchenko (2017) use mixed integer linear optimization for solving an L_0-variant of the Dantzig selector.

In spite of their usefulness, exact algorithms are usually accompanied by a steep increase in computational cost, placing them at a disadvantage compared with faster alternatives (Hastie et al. 2017). To this end, our approach borrows the computational strengths of the proxy algorithms while maintaining a notion of "local combinatorial exactness"—that is, making small perturbations to the support of the solution cannot improve its objective. Similar to the proxy algorithms, we employ cyclic CD as one of our main workhorses. We note that standard results on the convergence of cyclic CD (Tseng 2001) do not apply for our problem, and one of our contributions is rigorously establishing its convergence. A novelty of our work is the use of local combinatorial search to obtain high-quality solutions. Our attention to the delicate computational aspects make our proposed algorithms comparable (and at times faster) in speed to the fastest proxy algorithms (e.g., glmnet and ncvreg).

1.2. Contributions

We summarize our key contributions below:

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

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

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

4. Experiments on real and synthetic data sets suggest that our algorithms do a good job in optimizing problem (1), with solutions often found to be similar to that of exact MIO methods but with significantly shorter run times. In terms of statistical performance, our algorithms are found to be superior in terms of a combination of metrics (estimation, prediction, and variable selection) compared with state-of-the-art methods for sparse learning.

1.3. Notation

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

Proofs of lemmas and theorems are included in the supplementary material.

2. Necessary Optimality Conditions

We present a family of necessary optimality conditions for problem (1), leading to different classes of local minima.[2] Our methodology is centered on the following problem:

$$\min_{\beta \in \mathbb{R}^p} \quad F(\beta) \overset{\text{def}}{=} f(\beta) + \lambda_0 \|\beta\|_0, \tag{2}$$

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

$$f(\beta) \overset{\text{def}}{=} \frac{1}{2}\|y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2. \tag{3}$$

We will use the following shorthand notations: (i) (L_0L_2) to denote problem (2) with $\lambda_1 = 0$ and $\lambda_2 > 0$, (ii) (L_0L_1) to denote problem (2) with $\lambda_1 > 0$ and $\lambda_2 = 0$, and (iii) (L_0) to denote problem (2) with $\lambda_1 = \lambda_2 = 0$. Unless otherwise specified, we will assume that $\lambda_0 > 0$.

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

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

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

$$\text{Hierarchy:} \quad \begin{array}{c}\text{FSI}(k)\\ \text{Minima}\end{array} \subseteq \begin{array}{c}\text{PSI}(k)\\ \text{Minima}\end{array}$$

$$\subseteq \begin{array}{c}\text{CW}\\ \text{Minima}\end{array} \subseteq \begin{array}{c}\text{IHT}\\ \text{Minima}\end{array} \subseteq \begin{array}{c}\text{Stationary}\\ \text{Solutions}\end{array} \tag{4}$$

In the above hierarchy, stationary solutions are the weakest. As we move from the right to left, the classes become smaller (i.e., satisfy more restrictive necessary optimality conditions) until reaching the most restrictive class: FSI(k) minima. Moreover, for sufficiently large k, FSI(k) and PSI(k) minima coincide with the class of global minimizers of problem (2). We now present a formal treatment of the classes of minima introduced above.

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 (Bertsekas 2016) of g at β in the direction d by $g'(\beta; d) \overset{\text{def}}{=} \liminf_{\alpha \downarrow 0} (g(\beta +$

$\alpha d) - g(\beta))/\alpha$. Directional derivatives play an important role in describing necessary optimality conditions for continuous optimization problems (Bertsekas 2016). Although $F(\beta)$ is not continuous, it is insightful to use the notion of a directional derivative to arrive at a basic definition of stationarity for problem (2).

Definition 1 (Stationary Solution). A vector $\beta^* \in \mathbb{R}^p$ is a stationary solution for problem (2) if for every direction vector $d \in \mathbb{R}^p$, the lower directional derivative satisfies $F'(\beta^*; d) \geq 0$.

Let $\nabla f(\beta) \in \mathbb{R}^p$ denote a subgradient of $f(\beta)$. If β has a support S, the notation $\nabla_S f(\beta)$ refers to the components of $\nabla f(\beta)$ restricted to S. Lemma 1 gives an alternative characterization of Definition 1.

Lemma 1. *Let $\beta^* \in \mathbb{R}^p$ with support S; β^* is a stationary solution for problem (2) iff $\nabla_S f(\beta^*) = 0$.*

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

$$\beta_i^* = \text{sign}\left(\widetilde{\beta}_i^*\right) \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1 + 2\lambda_2} \quad \text{and} \quad |\widetilde{\beta}_i^*| > \lambda_1 \quad \text{for all } i \in \text{Supp}(\beta^*), \tag{5}$$

where we recall that $\widetilde{\beta}_i^* \overset{\text{def}}{=} \langle y - \sum_{j \neq i} X_j \beta_j^*, X_i \rangle$. Characterization (5) suggests that a stationary solution β^* does not depend on λ_0 and does not impose any restriction on the coordinates outside the support. Moreover, it can be readily verified that a stationary solution to problem (2) satisfies the traditional definition of a local minimum in nonlinear optimization; that is, if β^* is a stationary solution, then there exists a $\delta > 0$ such that $F(\beta^*) \leq F(\beta)$ for any β satisfying $\|\beta - \beta^*\| < \delta$.

2.2. CW Minima

We consider a class of stationary solutions inspired by coordinate-wise algorithms (Tseng 2001, Beck and Eldar 2013, Bertsekas 2016).

Definition 2 (CW Minimum). A vector $\beta^* \in \mathbb{R}^p$ is a CW minimum for problem (2) if for every $i \in [p]$, β_i^* is a minimizer of $F(\beta^*)$ with respect to the ith coordinate (with others held fixed); that is,

$$\beta_i^* \in \underset{\beta_i \in \mathbb{R}}{\arg\min}\, F\left(\beta_1^*, \dots, \beta_{i-1}^*, \beta_i, \beta_{i+1}^*, \dots, \beta_p^*\right). \tag{6}$$

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

$$\widetilde{T}\left(\widetilde{\beta}_i^*, \lambda_0, \lambda_1, \lambda_2\right) \overset{\text{def}}{=} \underset{\beta_i \in \mathbb{R}}{\arg\min} \left\{ \frac{1 + 2\lambda_2}{2} \left(\beta_i - \frac{\widetilde{\beta}_i^*}{1 + 2\lambda_2} \right)^2 + \lambda_1 |\beta_i| + \lambda_0 \mathbb{1}[\beta_i \neq 0] \right\}, \tag{7}$$

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

Lemma 2. *Let $\widetilde{T}$ be the thresholding operator defined in (7). Then,*

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

Lemma 3 presents an alternative characterization of CW minima.

Lemma 3. *A vector $\beta^* \in \mathbb{R}^p$ is a CW minimum if and only if*

$$\begin{aligned} &\beta_i^* = \text{sign}(\widetilde{\beta}_i^*) \frac{|\widetilde{\beta}_i^*| - \lambda_1}{1 + 2\lambda_2} \text{ and } |\beta_i^*| \geq \sqrt{\frac{2\lambda_0}{1 + 2\lambda_2}}, \quad \text{for every } i \in \text{Supp}(\beta^*) \\ \text{and} \quad &\frac{|\widetilde{\beta}_i^*| - \lambda_1}{1 + 2\lambda_2} \leq \sqrt{\frac{2\lambda_0}{1 + 2\lambda_2}} \quad \text{for every } i \notin \text{Supp}(\beta^*). \end{aligned} \tag{8}$$

Comparing (8) with (5), we see that the class of stationary solutions contains the class of CW minima, and the containment is strict (in general).

2.3. Swap-Inescapable Minima

We now introduce stationary solutions that further refine the class of CW minima, using notions from local combinatorial optimization. Given a CW minimum β^*, one might obtain a better solution by the following "swapping" operation: we set some nonzeros in β^* to 0 and allow some entries from outside the support of β^* to be nonzero. Then, we optimize over the new support using one of the following rules: (a) *partial optimization*, where we optimize only with respect to the coordinates added from outside the support, or (b) *full optimization*, where we optimize with respect to all the coordinates in the new support. This may lead to a solution with a smaller objective value. If the current solution cannot be improved using the swapping operation, we call β^* a *swap-inescapable* minimum. Our proposal is inspired by the work of Beck and Eldar (2013) for the cardinality-constrained problem, where the authors suggest a special case of partial swap optimization involving *one* coordinate. However, the problem studied here is different: we consider L_0 penalization (versus an L_0 constraint) and a nonsmooth $f(\beta)$. Furthermore, we allow multiple coordinates to be swapped at once via partial or full optimization.

2.3.1. PSI Minima. We formally define PSI minima, arising from the partial optimization step outlined above. Recall that for any $L \subseteq [p]$, the ith coordinate of the vector $(U^L\beta)$ is β_i if $i \in L$ and 0 otherwise.

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

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

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

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

$$\beta_i^* = \operatorname{sign}\left(\widetilde{\beta}_i^*\right)\frac{|\widetilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \quad \textit{and}$$

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

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

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

Lemmas 3 and 4 suggest that PSI(1) minima impose additional restrictions on the magnitude of nonzero coefficients when compared with CW minima. The class of CW minima contains PSI(k) minima for any k. Furthermore, as k increases, the class of PSI(k) minima becomes smaller till it coincides with the class of global minimizers of problem (2).

2.3.2. FSI Minima. We formally define FSI minima, arising from the full optimization step outlined above.

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

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

We note that for a fixed k, the class of PSI(k) minima contains FSI(k) minima, justifying a part of the hierarchy displayed in (4). As k increases, the class of FSI(k) minima becomes smaller till it coincides with the set of global minimizers of problem (2). Sections 3.2.1 and 3.2.2 introduce algorithms to obtain PSI(k) minima and FSI(k) minima, respectively.

2.4. Stationarity Motivated by IHT

Proximal gradient algorithms such as IHT are popularly used for L_0-penalized least squares problems (Blumensath and Davies 2009). It is insightful to consider the class of stationary solutions associated with IHT and study how they compare with CW minima. Let $f_d(\beta) := \frac{1}{2}\|y - X\beta\|^2 + \lambda_2\|\beta\|^2$. The gradient of $f_d(\beta)$ is Lipschitz continuous with parameter L; that is, $\|\nabla f_d(\beta) - \nabla f_d(\alpha)\| \le L\|\beta - \alpha\|$ for all $\beta, \alpha \in \mathbb{R}^p$. IHT applied to problem (2) performs the following updates:

$$\beta^{k+1} \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p}\left\{\frac{1}{2\tau}\left\|\beta - \left(\beta^k - \tau\nabla f_d(\beta^k)\right)\right\|^2 + \lambda_1\|\beta\|_1 + \lambda_0\|\beta\|_0\right\}, \tag{9}$$

where $\tau > 0$ is a constant step size. We say that $\alpha \in \mathbb{R}^p$ is a fixed point of update (9) if $\beta^k = \alpha$ leads to $\beta^{k+1} = \alpha$. This suggests another notion of stationarity (see Definition 5) for problem (2). To this end, consider Theorem 1 establishing the convergence of β^k to a fixed point of update (9).

Theorem 1. *Let L be defined as above. The sequence $\{\beta^k\}$ defined in (9) converges to a fixed point β^* of update (9) for any $\tau < \frac{1}{L}$. Note that β^* is a fixed point if and only if*

$$\beta_i^* = \operatorname{sign}(\tilde{\beta}_i^*)\frac{|\tilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \quad \textit{and} \quad |\beta_i^*| \ge \sqrt{2\lambda_0\tau} \qquad \textit{for } i \in \operatorname{Supp}(\beta^*)$$
$$\textit{and} \qquad \frac{|\tilde{\beta}_i^*| - \lambda_1}{1+2\lambda_2} \le \sqrt{\frac{2\lambda_0}{(1+2\lambda_2)^2\tau}} \qquad \textit{for } i \notin \operatorname{Supp}(\beta^*) \tag{10}$$

Definition 5. A vector β^* is an IHT minimum for problem (2) if it satisfies (10) for $\tau < \frac{1}{L}$.

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

Remark 1. 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 (9). Because the columns of X are normalized, we have $M \ge 1$. Comparing (10) with (8), we see that the class of IHT minima includes CW minima. Usually, for high-dimensional problems, $M \gg 1$—making the class of IHT minima much larger than CW minima (see Section 5.2 for numerical examples).

We have now explained the full hierarchy among the classes of local minima presented in (4). Section 3 discusses algorithms to obtain solutions that belong to these classes.

3. Algorithms

Section 3.1 presents a cyclic CD algorithm that converges to CW minima, and Section 3.2 discusses local combinatorial optimization to obtain PSI(k)/FSI(k) minima.

3.1. Cyclic Coordinate Descent

Our main workhorse is cyclic CD (Bertsekas 2016) with full minimization in every coordinate—we also include some additional tweaks for reasons we discuss shortly. With initialization β^0, we update the first coordinate (with others fixed) to get β^1, and we continue the updates according to a cyclic rule. Let β^k denote the solution obtained after performing k coordinate updates. Then, β^{k+1} is obtained by updating the ith coordinate (with others held fixed) via

$$\beta_i^{k+1} \in \arg\min_{\beta_i \in \mathbb{R}} F\left(\beta_1^k, \ldots, \beta_{i-1}^k, \beta_i, \beta_{i+1}^k, \ldots, \beta_p^k\right), \tag{11}$$

where $i = (k+1) \bmod (p+1)$. Recall that the operator $\widetilde{T}(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2)$ (defined in (7)) describes solutions of problem (11). Specifically, it returns two solutions when $\frac{|\widetilde{\beta}_i| - \lambda_1}{1+2\lambda_2} = \sqrt{\frac{2\lambda_0}{1+2\lambda_2}}$. In such a case, we consistently choose one of these solutions[3]—namely, the nonzero solution. Thus, we use the new operator (note the use of T instead of $\widetilde{T}$):

$$T\left(\widetilde{\beta}_i, \lambda_0, \lambda_1, \lambda_2\right) \overset{\text{def}}{=} \begin{cases} \operatorname{sign}\left(\widetilde{\beta}_i\right) \frac{|\widetilde{\beta}_i| - \lambda_1}{1+2\lambda_2} & \text{if } \frac{|\widetilde{\beta}_i| - \lambda_1}{1+2\lambda_2} \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ 0 & \text{if } \frac{|\widetilde{\beta}_i| - \lambda_1}{1+2\lambda_2} < \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \end{cases} \tag{12}$$

for update (11). In addition to the above modification, we introduce "spacer steps" that are occasionally performed during the course of the algorithm to *stabilize* its behavior[4]—spacer steps are commonly used in the context of continuous nonlinear optimization problems (e.g., see Bertsekas 2016). We perform a spacer step as follows. Let C be an a priori fixed positive integer. We keep track of the supports encountered so far, and when a certain support—say, S—appears for Cp-many times, we perform one pass of cyclic CD to minimize the continuous function $\beta_S \mapsto f(\beta_S)$. This entails updating every coordinate in S via the operator $T(\widetilde{\beta}_i, 0, \lambda_1, \lambda_2)$ (see Subroutine 1).

Algorithm 1 summarizes the above procedure. Count[S] is an associative array that stores the number of times a support S appears—it takes S as a key and returns the number of times S has appeared so far. Count[S] is initialized to 0 for any S that appears for the first time during the course of the algorithm. Note that in the worst case, storing Count[] may require an exponential (in p) amount of memory. However, in practice, only one or a few supports appear for Cp many times and need to be maintained in Count[].

Algorithm 1 (Coordinate Descent with Spacer Steps)

```
Input: Initial Solution $\beta^0$, 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 (12)
                                                        // Nonspacer Step
        $k \leftarrow k+1$
        Count[Supp($\beta^k$)] $\leftarrow$ Count[Supp($\beta^k$)] + 1
        if Count[Supp($\beta^k$)] = $Cp$ then
            $\beta^{k+1} \leftarrow$ SpacerStep($\beta^k$)          // Spacer Step
            Count[Supp($\beta^k$)] = 0
            $k \leftarrow k+1$
```

Subroutine 1 (SpacerStep(β))

```
Input: $\beta$
for $i$ in Supp($\beta$) do
    $\beta_i \leftarrow \arg\min_{\beta_i \in \mathbb{R}} f(\beta_1, \ldots, \beta_i, \ldots, \beta_p)$ using (12)
    with $\lambda_0 = 0$
return ($\beta$)
```

3.1.1. Why Cyclic CD? Cyclic CD has been practically shown to be among the fastest algorithms for Lasso (Friedman et al. 2010) and continuous nonconvex regularizers (e.g., MCP, SCAD) (Breheny and Huang 2011, Mazumder et al. 2011). Coordinate updates in cyclic CD have low cost and exploit sparsity via sparse residual updates and active set convergence (Friedman et al. 2010). This makes it well suited for high-dimensional problems with $p \gg n$ and p of the order of tens of thousands to millions. On the other hand, methods requiring evaluation of the full gradient (proximal gradient descent, greedy coordinate descent, etc.) can have difficulty in scaling with p (Nesterov 2012). For example, proximal gradient descent methods do not exploit sparsity-based structure as well as CD-based methods (Friedman et al. 2010, Nesterov 2012). We also note that, based on our experiments, random CD (proposed by Patrascu and Necoara 2015 for a problem similar to ours) exhibits slower convergence in practice (see Section 5.2)—see also related discussions in Beck and Tetruashvili (2013) for convex problems. We have also observed empirically that cyclic CD has an edge over competing algorithms, in terms of both optimization objective (see Section 5.2) and statistical performance (see Sections 5.3, 5.4, and 5.6).

3.1.2. Convergence Analysis. We analyze the convergence behavior of Algorithm 1—in particular, we prove a new result establishing convergence to a CW minimum (the limit point depends on the initialization). Moreover, we show that the linear rate of convergence of CD that holds for minimization of a smooth convex function (Beck and Tetruashvili 2013) extends to our nonconvex problem (in an asymptotic sense). We note that if we avoid full minimization and use a conservative step size, the proofs of convergence become straightforward by virtue of a sufficient decrease of the objective value after every coordinate update.[5] However, using CD with a conservative step size for problem (2) can have a detrimental effect on the solution quality. By examining the fixed points, it can be shown that a conservative step size leads to a class of stationary solutions that contains CW minima. Cyclic CD has been studied in earlier work with nonconvex but continuous regularizers (Tseng 2001, Breheny and Huang 2011, Mazumder et al. 2011) with a least squares data fidelity term. However, to our knowledge, a convergence analysis of cyclic CD for problem (2) is novel.

We present a few lemmas describing the behavior of Algorithm 1 (some additional technical lemmas are in the supplementary material). Theorem 2 establishes convergence, and Theorem 3 presents an asymptotic linear rate of convergence of Algorithm 1.

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

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

For the remainder of the section, we will make the following minor assumptions to establish the convergence of Algorithm 1 for the (L_0) and (L_0L_1) problems. These assumptions are not needed for problem (L_0L_2).

Assumption 1. *Let* $m = \min\{n, p\}$. *Every set of m columns of X is linearly independent.*

Assumption 2 (Initialization). *If* $p > n$, *we assume that the initial estimate* β^0 *satisfies*

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

The following remark demonstrates that Assumption 2 is rather minor.

Remark 2. 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 $F(\beta^0) = \lambda_0 n$ (because the least squares loss is 0). This satisfies Assumption 2. For the (L_0L_1) problem, we note that there always exists an optimal Lasso solution $\hat{\beta}$ such that $\|\hat{\beta}\|_0 \leq n$ (e.g., see Tibshirani 2013). Therefore, $\hat{\beta}$ satisfies Assumption 2.

In what follows, we assume that Assumptions 1 and 2 hold for the (L_0) and (L_0L_1) problems. Lemma 6 shows that in the (L_0) and (L_0L_1) problems, the support size of any β^k obtained by Algorithm 1 cannot exceed $\min\{n, p\}$.

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

The following theorem establishes the convergence of Algorithm 1.

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

(1) *The support of* $\{\beta^k\}$ *stabilizes after a finite number of iterations; that is, 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* $\text{Supp}(B) = S$.

Theorem 3 presents an asymptotic linear rate of convergence for Algorithm 1: we make use of part (1) of Theorem 2 and the convergence rate of cyclic CD (for smooth and strongly convex functions) (Beck and Tetruashvili 2013). Note that in Theorem 3, *full cycle* refers to a single pass of vanilla CD over all the coordinates in S, and β^K refers to the iterate generated after performing K full cycles of CD.

Theorem 3 (Adaptation of Beck and Tetruashvili 2013, theorem 3.9). *Let* $\{\beta^K\}$ *be the full-cycle iterates generated by Algorithm 1, and let B be the limit with support S. Let* m_S *and* M_S *denote the smallest and largest eigenvalues of* $X_S^T X_S$, *respectively. Then, there is an integer N such that for all* $K \geq N$, *the following holds:*

$$F\left(\beta^{K+1}\right) - F(B) \leq \left(1 - \frac{m_S + 2\lambda_2}{2(1+2\lambda_2)\left(1+|S|\left(\frac{M_S+2\lambda_2}{1+2\lambda_2}\right)^2\right)}\right) \cdot \left(F\left(\beta^K\right) - F(B)\right). \quad (13)$$

3.2. Local Combinatorial Optimization Algorithms

We present algorithms to obtain solutions belonging to the classes of swap-inescapable minima introduced in Section 2.3.

3.2.1. Algorithms for PSI Minima.

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

$$\min_{\beta, S_1, S_2} \; F\left(\beta^\ell - U^{S_1}\beta^\ell + U^{S_2}\beta\right) \quad \text{s.t.} \; S_1 \subseteq S,\; S_2 \subseteq S^c, |S_1| \leq k,\; |S_2| \leq k, \quad (14)$$

where $S = \text{Supp}(\beta^\ell)$. Note that if there is a feasible solution $\hat{\beta}$ to problem (14) satisfying $F(\hat{\beta}) < F(\beta^\ell)$, then $\hat{\beta}$ may not be a CW minimum. In this case, Algorithm 1 can be initialized with $\hat{\beta}$ to obtain a better solution for problem (2). Otherwise, if such a $\hat{\beta}$ does not exist, then β^ℓ is a PSI(k) minimum (by Definition 3). Algorithm 2 (a.k.a. CD-PSI(k)) summarizes the algorithm.

Algorithm 2 (CD-PSI(k))

```
$\hat{\beta}^0 \leftarrow \beta^0$
for $\ell = 0, 1, \ldots$ do
    $\beta^{\ell+1} \leftarrow$ Output of Algorithm 1 initialized with $\hat{\beta}^\ell$
    if problem (14) has a feasible solution $\hat{\beta}$ satisfying $F(\hat{\beta}) < F(\beta^{\ell+1})$ then
        $\hat{\beta}^{\ell+1} \leftarrow \hat{\beta}$
    else
        Terminate
```

Theorem 4. *Let $\{\beta^\ell\}$ be the sequence of iterates generated by Algorithm 2. For the (L_0) and (L_0L_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.*

As indicated in Theorem 4, Algorithm 2 terminates in a finite number of iterations (which depends on the number of swaps that improve the objective value). In our experiments, Algorithm 2 typically terminates in fewer than 20 iterations. Each iteration involves a call to Algorithm 1 (which is quite fast) followed by solving a feasibility problem (i.e., finding $\hat{\beta}$ as described in Algorithm 2). As we discuss next, for moderate p (e.g., 10^3–10^4), this feasibility problem can be handled efficiently using MIO solvers. When $k = 1$, we propose a specialized algorithm for solving this feasibility problem that can easily scale to problems with $p = 10^6$.

3.2.1.1. MIO Formulation for Problem (14). Problem (14) admits an MIO formulation given by

$$\min_{\theta,\beta,z} \quad f(\theta) + \lambda_0 \sum_{i\in[p]} z_i \tag{15a}$$

$$\text{s.t.} \quad \theta = \beta^\ell - \sum_{i\in S} e_i \beta_i^\ell (1 - z_i) + \sum_{i\in S^C} e_i \beta_i \tag{15b}$$

$$-\mathcal{M} z_i \le \beta_i \le \mathcal{M} z_i, \quad \forall i \in S^c \tag{15c}$$

$$\sum_{i\in S^c} z_i \le k \tag{15d}$$

$$\sum_{i\in S} z_i \ge |S| - k \tag{15e}$$

$$\beta_i \in \mathbb{R}, \quad \forall i \in S^c \tag{15f}$$

$$z_i \in \{0,1\}, \quad \forall i \in [p], \tag{15g}$$

where the optimization variables are $\theta \in \mathbb{R}^p$, $\beta_i, i \in S^c$, and $z \in \{0,1\}^p$. In formulation (15), $S = \text{Supp}(\beta^\ell)$, where β^ℓ is fixed, and $\mathcal{M}$ is a Big-M parameter (a priori specified) controlling the L_∞ norm of β_{S^c}. Any sufficiently large value of $\mathcal{M}$ will lead to a solution for problem (14); however, a tight choice for $\mathcal{M}$ affects the run time of the MIO solver—see Bertsimas et al. (2016) 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 (15) is a mixed integer quadratic optimization (MIQO) problem.

We now explain the constraints in problem (15) and how they relate to problem (14). To this end, let S_1 and S_2 be subsets defined in (14). Let $\theta = \beta^\ell - U^{S_1}\beta^\ell + U^{S_2}\beta$, and this relation is expressed in (15b). Let us consider any binary variable z_i where $i \in S$. If $z_i = 0$, then β_i^ℓ is removed from S, and we have $\theta_i = 0$ (see (15b)). If $z_i = 1$, then β_i^ℓ is not removed from θ, and we have $\theta_i = \beta_i^\ell \neq 0$ (see (15b)). Note that $|S_1| = \sum_{i\in S}(1 - z_i) = |S| - \sum_{i\in S} z_i$. The condition $|S_1| \le k$ is thus encoded in the constraint $\sum_{i\in S} z_i \ge |S| - k$ in (15e). Thus we have that $\|\theta_S\|_0 = \sum_{i\in S} z_i$.

Now consider any binary variable z_i where $i \in S^c$. If $z_i = 1$, then by (15c) we observe that β_i is free to vary in $[-\mathcal{M}, \mathcal{M}]$. This implies that $\theta_i = \beta_i$. If $z_i = 0$, then $\theta_i = \beta_i = 0$. Note $\sum_{i\in S^c} z_i = |S_2|$, and the constraint $|S_2| \le k$ is expressed via $\sum_{i\in S^c} z_i \le k$ in (15d). It also follows that $\|\theta_{S^c}\|_0 = \sum_{i\in S^c} z_i$. Finally, we note that the function appearing in the objective (15a) is $F(\theta)$, because $\lambda_0 \sum_{i\in[p]} z_i = \lambda_0 \|\theta\|_0$.

Remark 3. Problem (15) has a smaller (combinatorial) search space compared with an MIO formulation for the full problem (2)—solving (15) for small values of k is usually much faster than problem (2). Furthermore, an MIO framework can quickly deliver a feasible solution to problem (15) with a smaller objective than the current solution—this is usually much faster than establishing optimality via dual bounds. Note that the MIO framework can also certify (via dual bounds) if there is no feasible solution with a strictly smaller objective value.

Section 5 presents examples where problem (15) leads to higher-quality solutions—from both the optimization and statistical performance viewpoints. We now present an efficient algorithm for solving the special case of problem (14) with $k = 1$.

Subroutine 2 (Vanilla Implementation of Problem (14) with $k = 1$)

```
$S \leftarrow \text{Supp}(\beta^\ell)$
for $i \in S$ do
    for $j \in S^c$ do
        $v_j^* \leftarrow \arg\min_{v_j\in\mathbb{R}} F(\beta^\ell - e_i\beta_i^\ell + e_j v_j)$    (16)
        $F_j^* \leftarrow F\left(\beta^\ell - e_i\beta_i^\ell + e_j v_j^*\right)$    (17)
    $\vartheta \leftarrow \arg\min_{j\in S^c} F_j^*$    (18)
    if $F_\vartheta^* < F(\beta^\ell)$ then
        $\hat{\beta} \leftarrow \beta^\ell - e_i\beta_i^\ell + e_\vartheta v_\vartheta^*$    (19)
        Terminate
```

3.2.1.2. An Efficient Algorithm for Computing PSI(1) Minima. Subroutine 2 presents an algorithm for problem (14) with $k = 1$. That is, we search for a feasible solution $\hat{\beta}$ of problem (14) satisfying $F(\hat{\beta}) < F(\beta^\ell)$.

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 (16) and evaluate the new objective in (17). Therefore, Subroutine 2 entails an overall cost of $O(n(p- \|\beta^\ell\|_0)\|\beta^\ell\|_0)$. However, we show below that a careful implementation can reduce the cost by a factor of n, leading to a cost of $O((p - \|\beta^\ell\|_0)\|\beta^\ell\|_0)$-many operations.

A solution v_j^* of problem (16) is given by

$$v_j^* = \begin{cases} \text{sign}(\bar\beta_j)\frac{|\bar\beta_j|-\lambda_1}{1+2\lambda_2} & \text{if } \frac{|\bar\beta_j|-\lambda_1}{1+2\lambda_2} \geq \sqrt{\frac{2\lambda_0}{1+2\lambda_2}} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where

$$\bar\beta_j = \langle r + X_i\beta_i^\ell, X_j\rangle = \langle r, X_j\rangle + \langle X_i, X_j\rangle\beta_i^\ell, \tag{21}$$

and $r = y - X\beta^\ell$. We note that in Algorithm 2, solving (14) is directly preceded by a call to Algorithm 1. The quantities $\langle r, X_j\rangle$ and $\langle X_i, X_j\rangle$ appearing on the right-hand side of (21) can be stored during the call to Algorithm 1 (these two quantities are computed by CD as part of the "covariance updates"—see Section 4 for details). By reusing these two stored quantities, we can compute every $\bar\beta_j$ (and consequently, v_j^*) in $O(1)$ arithmetic operations.

Furthermore, the following equivalent representations hold:

$$\arg\min_{j\in S^c} F_j^* \iff \arg\max_{j\in S^c} |v_j^*| \tag{22}$$

$$F_\vartheta^* < F\left(\beta^l\right) \iff |v_\vartheta^*| > |\beta_i^l|. \tag{23}$$

Thus, we can avoid the computation of the objective F_j^* in (17) and replace (18) with $\vartheta \leftarrow \arg\max_{j\in S^c}|v_j^*|$. Furthermore, we can replace $F_\vartheta^* < F(\beta^\ell)$ (before Equation (19)) with $|v_\vartheta^*| > |\beta_i^l|$. We summarize these changes in Subroutine 3, which is the efficient counterpart of Subroutine 2. Note that Subroutine 3 has a cost of $O((p - \|\beta^l\|_0)\|\beta^l\|_0)$ operations.

Subroutine 3 (Efficient Implementation of Problem (14) with $k = 1$)

```
$S \leftarrow \text{Supp}(\beta^\ell)$
for $i \in S$ do
    for $j \in S^c$ do
        Compute $v_j^*$ in $O(1)$ using (20)
            $\vartheta \leftarrow \arg\max_{j\in S^c} |v_j^*|$
    if $|v_\vartheta^*| > |\beta_i^\ell|$ then
            $\hat\beta \leftarrow \beta^\ell - e_i\beta_i^\ell + e_\vartheta v_\vartheta^*$

            Terminate
```

Remark 4. Because 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).

3.2.2. Algorithm for FSI Minima. To obtain an FSI(k) minimum, problem (14) needs to be modified—we replace optimization with respect to the variable $U^{S_2}\beta$ by that of $U^{(S\setminus S_1)\cup S_2}\beta$. This leads to the following problem:

$$\min_{\beta,S_1,S_2} F\left(\beta^\ell - U^{S_1}\beta^\ell + U^{(S\setminus S_1)\cup S_2}\beta\right)$$
$$\text{s.t. } S_1 \subseteq S,\ S_2 \subseteq S^c, |S_1| \leq k,\ |S_2| \leq k, \tag{24}$$

where $S = \text{Supp}(\beta^l)$. Similarly, Algorithm 2 gets modified by considering problem (24) instead of problem (14). 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 an FSI(k) minimum.

We present an MIO formulation for problem (24). We write $\theta = \beta^\ell - U^{S_1}\beta^\ell + U^{(S\setminus S_1)\cup S_2}\beta$ and use a binary variable $w_i, i \in S$ to indicate whether $i \in S_1$ or not: we set $w_i = 0$ iff $i \in S_1$. We use another binary variable $z_i, i \in [p]$ to indicate the number of nonzeros in θ (i.e., $z_i = 0 \Rightarrow \theta_i = 0$). This leads to the following MIO problem:

$$\min_{\theta,w,z} \quad f(\theta) + \lambda_0 \sum_{i\in[p]} z_i \tag{25a}$$
$$-\mathcal{M}z_i \leq \theta_i \leq \mathcal{M}z_i, \quad \forall i \in [p] \tag{25b}$$
$$z_i \leq w_i, \quad \forall i \in S \tag{25c}$$
$$\sum_{i\in S^c} z_i \leq k \tag{25d}$$
$$\sum_{i\in S} w_i \geq |S| - k \tag{25e}$$
$$\theta_i \in \mathbb{R}, \quad \forall i \in [p] \tag{25f}$$
$$z_i \in \{0,1\}, \quad \forall i \in [p] \tag{25g}$$
$$w_i \in \{0,1\} \quad \forall i \in S. \tag{25h}$$

In (25b), for every $i \in [p]$, the binary variable $z_i = 1$ if $i \in \text{Supp}(\beta)$, and $\mathcal{M}$ is a sufficiently large constant (similar to that in (15)). The second term in the objective (25a) stands for $\lambda_0 \sum_i z_i = \lambda_0\|\theta\|_0$. In (25c) we enforce the condition that if $w_i = 0$, then $z_i = 0$, implying that $\theta_i = 0$. If $w_i = 1$, then $i \notin S_1$. Coordinates in $S \setminus S_1$ (i.e., those with $w_i = 1$) are free to be inside or outside of Supp(θ). We enforce $|S_1| \leq k$ and $|S_2| \leq k$ via (25e) and (25d), respectively.

Remark 5. Formulation (25) has a larger search space compared with formulation (15) of PSI minima, as a result of the additional number of continuous and binary variables. Although this leads to increased run times compared with problem (15), it can still be solved faster than an MIO formulation for the full problem (2) (for the same reasons as in Remark 3).

In Section 5.5, we present experiments where we compare the quality of FSI(k) minima, for different values of k, to the other classes of minima.

4. Efficient Computation of the Regularization Path

We designed L0Learn:[6] an extensible C++ toolkit with an R interface that implements most of the algorithms discussed in this paper. Our toolkit achieves lower running times[7] compared with other popular sparse learning toolkits (e.g., glmnet and ncvreg) by utilizing a series of computational tricks and heuristics. These include an adaptive grid of tuning parameters, continuation, active set updates, greedy cyclic ordering of coordinates, correlation screening, and a careful accounting of floating point operations—some of these heuristics (as specified below) appear in prior work for deriving highly efficient algorithms for the Lasso (e.g., glmnet). Below, we provide a detailed account of the aforementioned strategies that are also found to result in high-quality solutions.

4.1. Adaptive Selection of Tuning Parameters

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

Lemma 7. *Suppose $\beta^{(i)}$ is the output of Algorithm 1 with $\lambda_0 = \lambda_0^i$. Let $S = \text{Supp}(\beta^{(i)})$, let $r = y - X\beta^{(i)}$ denote the residual, and let*

$$M^i = \frac{1}{2(1+2\lambda_2)} \max_{j \in S^c} (|\langle r, X_j \rangle| - \lambda_1)^2. \tag{26}$$

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 $\lambda_0^{i+1} \in (M^i, \lambda_0^i]$.

Lemma 7 suggests a simple scheme to compute a sequence $\{\lambda_0^i\}$ that avoids duplicate solutions. Suppose we have computed the regularization path up to $\lambda_0 = \lambda_0^i$; then λ_0^{i+1} can be computed as $\lambda_0^{i+1} = \alpha M^i$, where α is a fixed scalar in $(0, 1)$. Moreover, we note that M^i (defined in (26)) can be computed without explicitly calculating $\langle r, X_i \rangle$ for every $i \in S^c$, as these dot products can be maintained in memory while running Algorithm 1 with $\lambda_0 = \lambda_0^i$. Therefore, computing M^i, and consequently λ_0^{i+1}, requires only $O(|S^c|)$ operations.

4.2. (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 reorder the coordinates based on sorting the quantities $|\langle r^0, X_i \rangle|$ for $i \in [p]$ in descending order.[8] In practice, we perform *partial sorting*, in which only the top t (e.g., $t = 5000$ and p is much larger) coordinates are sorted, whereas the rest maintain their initial order. This is typically faster and equally effective compared with sorting all coordinates. Note that this ordering is performed *once* before the start of Algorithm 1—this is different from the greedy CD that finds the maximal correlation at every coordinate update. Our experiments in Section 5.2 indicate that (partially) greedy cyclic order performs significantly better than a vanilla cyclic order or random order.

4.3. Correlation Screening

When using continuation, we perform a screening method inspired by Tibshirani et al. (2012).[9] We restrict the updates of Algorithm 1 to the support of the warm start in addition to a small portion (e.g., top 10^3) of other coordinates that are highly correlated with the current residuals—these coordinates are readily available as a byproduct of the (partially) greedy cyclic ordering rule, described above. After convergence on the screened support, we check whether any coordinate from outside the support violates the conditions of a CW minimum and rerun the algorithm if needed. Typically, the solution obtained from the screened set turns out to be a CW minimum, and only one pass is done over all the coordinates.

4.4. Active Set Updates

As in prior work (Friedman et al. 2010), active set methods are found to be very useful in our context as well. From an empirical standpoint, the iterates generated by Algorithm 1 can typically achieve support stabilization in fewer than 10 full cycles.[10] This is further supported by Theorem 2, which establishes finite-time stabilization of the support. If the support does not change across multiple consecutive full cycles, we restrict the updates of Algorithm 1 to the current support. After convergence on this support, we check whether any coordinate from outside the support violates the conditions of a CW minimum and rerun the algorithm if needed.

4.5. Fast Coordinate Updates

Following Friedman et al. (2010), we present techniques for efficiently computing the coordinate updates;

these are also found to be useful for our implementation of PSI(1).

Let β^k be the current iterate in Algorithm 1, and let r^k be the residuals. To compute $\widetilde{\beta}_i = \langle r^k, X_i\rangle + \beta_i^k$, one can use one of the following rules that exploit sparsity (Friedman et al. 2010):

i. *Residual updates:* We maintain the residuals r^k throughout the algorithm and compute $\widetilde{\beta}_i$ using $O(n)$ operations. Once β_i^{k+1} is computed, we update r^{k+1} with cost $O(n)$ operations. Because β^k is sparse, many of the coordinates remain at 0 during the algorithm implying that $r^{k+1} = r^k$.

ii. *Covariance updates:* This appears in Friedman et al. (2010) for updating $\widetilde{\beta}_i$ without using the precomputed r^k. Note that Algorithm 1 precomputes $\langle y, X_i\rangle$ for all $i \in [p]$. If a coordinate ℓ enters the support for the first time, we compute and store the covariance terms: $\langle X_\ell, X_j\rangle$ for all $j \in [p]$ with cost $O(np)$. In iteration $k+1$, we compute $\widetilde{\beta}_i$ using these covariances and exploiting the sparsity of β^k—this costs $O(\|\beta^k\|_0)$ operations. This costs less than computing $\widetilde{\beta}_i$ using rule i if $\|\beta^k\|_0 < n$.

Scheme ii is useful when the supports encountered by Algorithm 1 are small with respect to n. It is also useful for an efficient implementation of the PSI(1) algorithm as it stores the dot products required in (21). However, when the supports encountered by CD are relatively large compared with n (e.g., 10% of n), then scheme i can become significantly faster because the dot product computations can be accelerated using calls to the Basic Linear Algebra Subprograms (BLAS).

5. Computational Experiments

In this section, we investigate both the optimization and statistical performances of our proposed algorithms and compare them to other popular sparse learning algorithms. For convenience, we provide a road map of this section. Section 5.2 compares the optimization performance of our proposed algorithms and other variants of CD and IHT. Section 5.3 empirically studies the statistical properties of estimators available from our proposed algorithms versus others for varying sample sizes. Section 5.4 provides a similar study for varying SNR. Section 5.5 performs an in-depth investigation among the PSI(k)/FSI(k) algorithms for different values of k. Section 5.6 presents timing and statistical performance comparisons on some large-scale instances, including real data sets. Additional experiments are placed in the supplementary material.

5.1. Experimental Setup

5.1.1. Data Generation. We consider a series of experiments on synthetic data sets 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 use a sparse coefficient vector $\beta^\dagger$ with $k^\dagger$ equispaced nonzero entries, each set to 1. We then generate the response vector $y = X\beta^\dagger + \epsilon$, where $\epsilon_i \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$ is independent of X. We define the SNR by $\text{SNR} = \frac{\text{Var}(X\beta^\dagger)}{\text{Var}(\epsilon)} = \frac{\beta^{\dagger T}\Sigma\beta^\dagger}{\sigma^2}$. An alternative to the SNR is the "proportion of variance explained" or R^2 for the *true model*: $R^2 = 1 - \frac{\text{Var}(y - X\beta^\dagger)}{\text{Var}(y)} = \frac{\text{SNR}}{\text{SNR}+1}$.

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

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

We select the tuning parameters by minimizing the prediction error on a separate validation set, which is generated under the fixed design setting as $y' = X\beta^\dagger + \epsilon'$, where $\epsilon_i' \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$. In the supplementary material, we also include alternative validation strategies. In particular, we include the results of the experiments in Sections 5.3 and 5.4 based on both oracle and random design tuning (following Hastie et al. 2017). The results are found to be quite similar.

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

- *Lasso:* We use our own implementation, and in the figures we denote it by "L_1."
- *Relaxed Lasso:* We use the Relaxed Lasso version suggested in Hastie et al. (2017), defined as $\beta^{\text{relaxed}} = \gamma\beta^{\text{lasso}} + (1-\gamma)\beta^{\text{LS}}$, where β^{lasso} is the Lasso estimate, the nonzero components of β^{LS} are given by the least squares solution on the support of β^{lasso}, and $\gamma \in [0,1]$ is a second tuning parameter (in addition to the tuning parameter for the Lasso). We use our own implementation for relaxed lasso and denote it by "L1Relaxed."
- *Elastic net:* This uses a combination of the L_1 and L_2 regularization (Zou and Hastie 2005). We use the implementation of glmnet and refer to it as "L_1L_2."
- *MCP:* This is the MCP penalty of Zhang (2010). We use the implementation provided in ncvreg (Breheny and Huang 2011).
- *Forward stepwise:* We use the implementation of Hastie et al. (2017) and denote it by "FStepwise."
- *IHT:* We use our implementation for IHT for the (L_0) problem with a step size of M^{-1}, where M is defined in Remark 1.

For all the methods involving one tuning parameter, we tune over a grid of 100 parameter values, except for forward stepwise selection, which we allow to run for up to 250 steps. For the methods with two parameters, we tune over a two-dimensional grid of 100×100 values. For our algorithms, the tuning parameter λ_0 is generated according to Section 4. For the (L_0L_2) problem, we sweep λ_2 between $0.1\lambda_2^*$ and $10\lambda_2^*$, where λ_2^* is the optimal regularization parameter for ridge regression (based on validation over 100 grid points between 0.0001 and 1,000). For (L_0L_1), Lasso, relaxed Lasso, and elastic net, we sweep λ_1 from $\|X^Ty\|_\infty$ down to $0.0001 \times \|X^Ty\|_\infty$. For relaxed Lasso and elastic net, we sweep their second parameter between 0 and 1. For MCP, the range of the first parameter λ is chosen by ncvreg, and we sweep the second parameter γ between 1.5 and 25.

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

- *Prediction error:* This is the same measure used in Bertsimas et al. (2016) and is defined by $\|X\hat{\beta} - X\beta^\dagger\|^2/\|X\beta^\dagger\|^2$. The prediction error of a perfect model is 0 and that of the null model ($\hat{\beta} = 0$) is 1.
- *L_∞ norm:* This is the estimation error measured in terms of the L_∞ norm: $\|\hat{\beta} - \beta^\dagger\|_\infty$.
- *Full support recovery:* We study whether the support of $\beta^\dagger$ is completely recovered by $\hat{\beta}$ (i.e., $1[\text{Supp}(\beta^\dagger) = \text{Supp}(\hat{\beta})]$); we look at the average of this quantity across multiple replications, leading to an estimate for the probability of full support recovery.

We would like to point out that SNR alone does not dictate how difficult the underlying statistical problem is (e.g., in terms of variable selection, estimation, or prediction error). The situation is rather subtle: in addition to SNR, the matrix X and choices of n, p, and $k^\dagger$ influence the statistical performance. For example, consider two instances with $p = 100$ and $p = 10^5$, where we set $k^\dagger = 10, \Sigma = I, n = 100$, and SNR = 1. For $p = 100$, it is possible to obtain a model with estimation error smaller than the null model (with high probability), but this may not be possible for $p = 10^5$. Similarly, for the case of constant correlation, a problem with $\rho = 0$ and SNR = 1 may be statistically easier than one with $\rho = 0.5$ and SNR = 100 (with suitable choices of n, p, and $k^\dagger$). The experiments that follow are intended to provide a solid understanding of how support recovery, sparsity, estimation error, and prediction error vary under different problem settings. We also seek to understand (i) when our algorithms can achieve full support recovery—a topic of considerable importance in high-dimensional statistics (Wainwright 2009, Gamarnik and Zadik 2017)—and (ii) when pure L_0 starts to overfit (a deeper statistical understanding of this seems to be in a nascent stage).

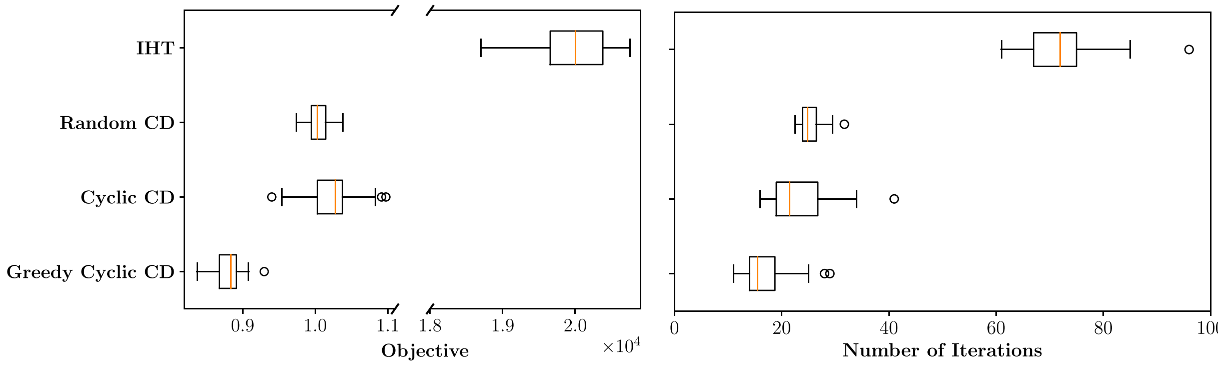
5.2. Comparison Among CD Variants and IHT: Optimization Performance

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

- *Cyclic CD:* This is Algorithm 1 with default cyclic order.
- *Random CD:* This is a randomized version of CD, where the coordinates to be updated are chosen uniformly at random from $[p]$. This has been considered in Patrascu and Necoara (2015).
- *Greedy cyclic CD:* This is our proposed Algorithm 1 with a partially greedy cyclic ordering of coordinates, described in Section 4.

We generated a data set with exponential correlation, $\rho = 0.5$, $n = 500$, $p = 2000$, SNR = 10, and a support size $k^\dagger = 100$. We generated 50 random initializations each with a support size of 100, where the nonzero indices are selected uniformly at random in 1 to p and assigned values that are drawn from Uniform(0, 1). For every initialization, we ran cyclic CD, greedy cyclic CD, and IHT and recorded the value of the objective function of the solution along with the number of iterations (here, one full pass over all p coordinates is defined as one iteration) till convergence. For random CD, we ran the algorithm 10 times

Figure 1. (Color online) 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



Notes. For each algorithm, we used 50 random initializations (as described in Section 5.2). The ticks of the box plots represent 1.5 times the interquartile range.

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 less than $< 10^{-7}$. Figure 1 presents the results: the objective values resulting from greedy cyclic CD are significantly lower than the other methods; on average, we have roughly a 12% improvement in the objective from random CD and 55% improvement over IHT. This finding can be partially explained in the light of our discussion in Section 2.4, 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% fewer iterations than random CD and 75% fewer iterations than IHT.

5.3. Statistical Performance for Varying Sample Sizes

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

We consider Algorithms 1 and 2 (with $k = 1$) for the (L_0), (L_0L_1), and (L_0L_2) problems, in addition to the competing penalties discussed in Section 5.1. In Experiments 1 and 2, we swept n between 100 and 1,000, and for every value of n, we generated 20 random training and validation data sets having exponential correlation, $p = 1000$, $k^{\dagger} = 20$, and SNR = 5. All the results we report here are based on validation set tuning. In the supplementary material, we include the results for oracle and random design tuning.

5.3.1. Experiment 1: High Correlation. Here, we choose $\rho = 0.9$ (exponential correlation)—this is a difficult problem because of the high correlations among features in the sample. Figure 2 summarizes the results. In the top panel of Figure 2, we show the results of Algorithm 2 applied to the (L_0) and (L_0L_2) problems versus the other competing algorithms. In the bottom panel, we present a detailed comparison among Algorithms 1 and 2 for all the three problems: (L_0), (L_0L_1), and (L_0L_2).

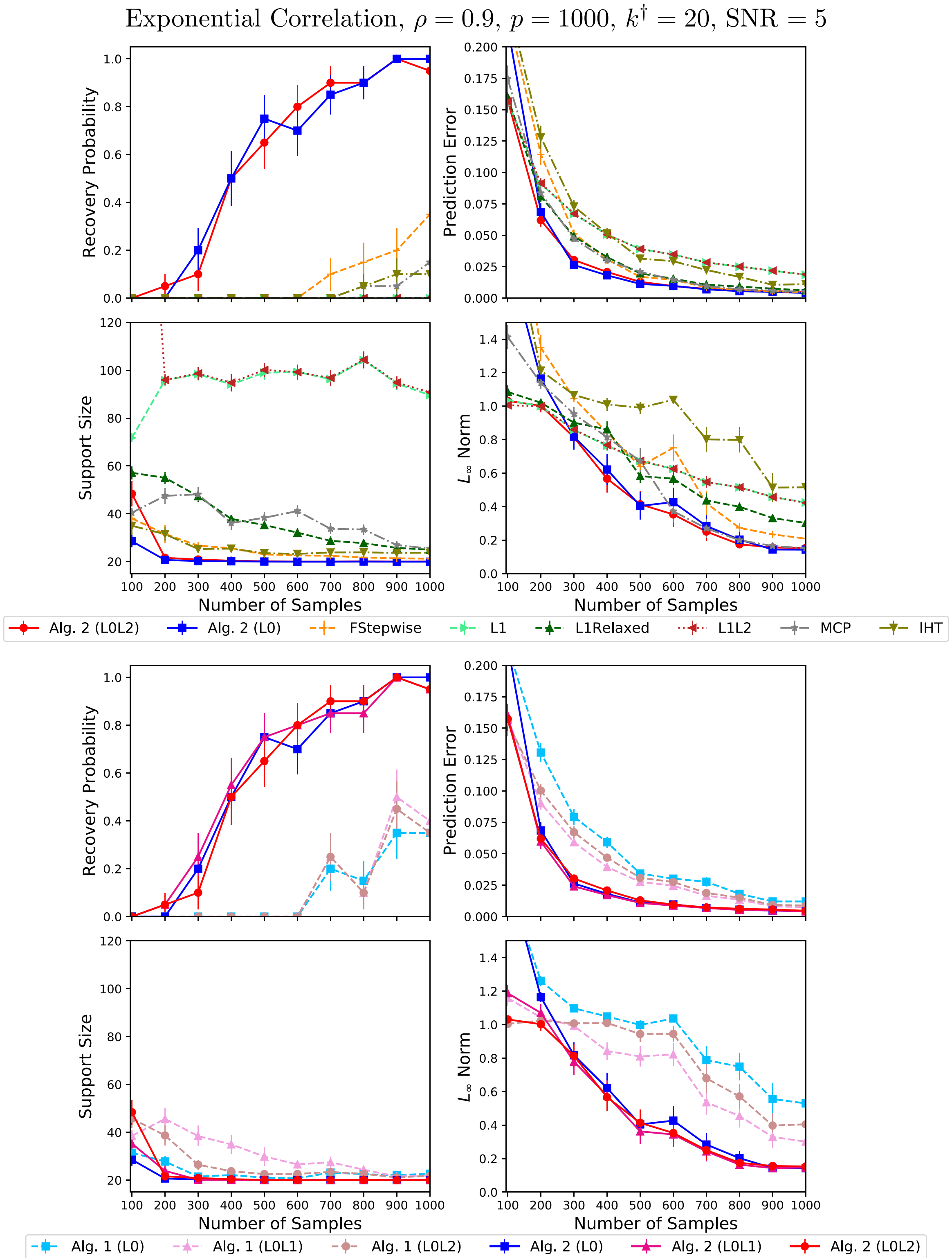
From the top panel (Figure 2), we can see that Algorithm 2 applied to the (L_0L_2) problem overall achieves the best performance in terms of different measures across all values of n. Algorithm 2 (L_0) and Algorithm 2 (L_0L_2) perform similarly for $n \geq 300$. The probability of full recovery for Algorithm 2 increases with n and becomes 1 at about $n = 900$. Note that the slight variation in the recovery probability values for our methods are solely due to the validation procedure (the oracle tuning presented in the supplementary material eliminates this wiggly behavior). Lasso, relaxed Lasso, and elastic net do not achieve full support recovery for any n; the corresponding lines are aligned with the horizontal axis. The L_1-based methods also lead to large support sizes—as expected, L_1L_2 leads to supports that are denser than Lasso (Zou and Hastie 2005). Because of shrinkage, tuning parameter selection based on prediction error makes the Lasso select models with many nonzero coefficients—this leads to suboptimal variable selection. The relaxed Lasso attempts to undo the shrinkage effect of the Lasso leading to models with fewer nonzeros. In addition, we note that shrinkage of Lasso also interferes with variable selection—a shortcoming that is inherited by the relaxed Lasso—as seen in the panel displaying recovery probability.

Moreover, MCP, FStepwise, and IHT have a probability of recovery about 0.3 even when $n = p = 1000$—suggesting that they fail to do correct support recovery in this regime. A similar phenomenon occurs for the prediction error and the L_∞ norm, where Algorithm 2 is seen to dominate.

The bottom figure shows that Algorithm 2 can significantly outperform Algorithm 1 (which performs no swaps). It seems that in this highly correlated setting, our local combinatorial optimization procedures have an edge in performance.

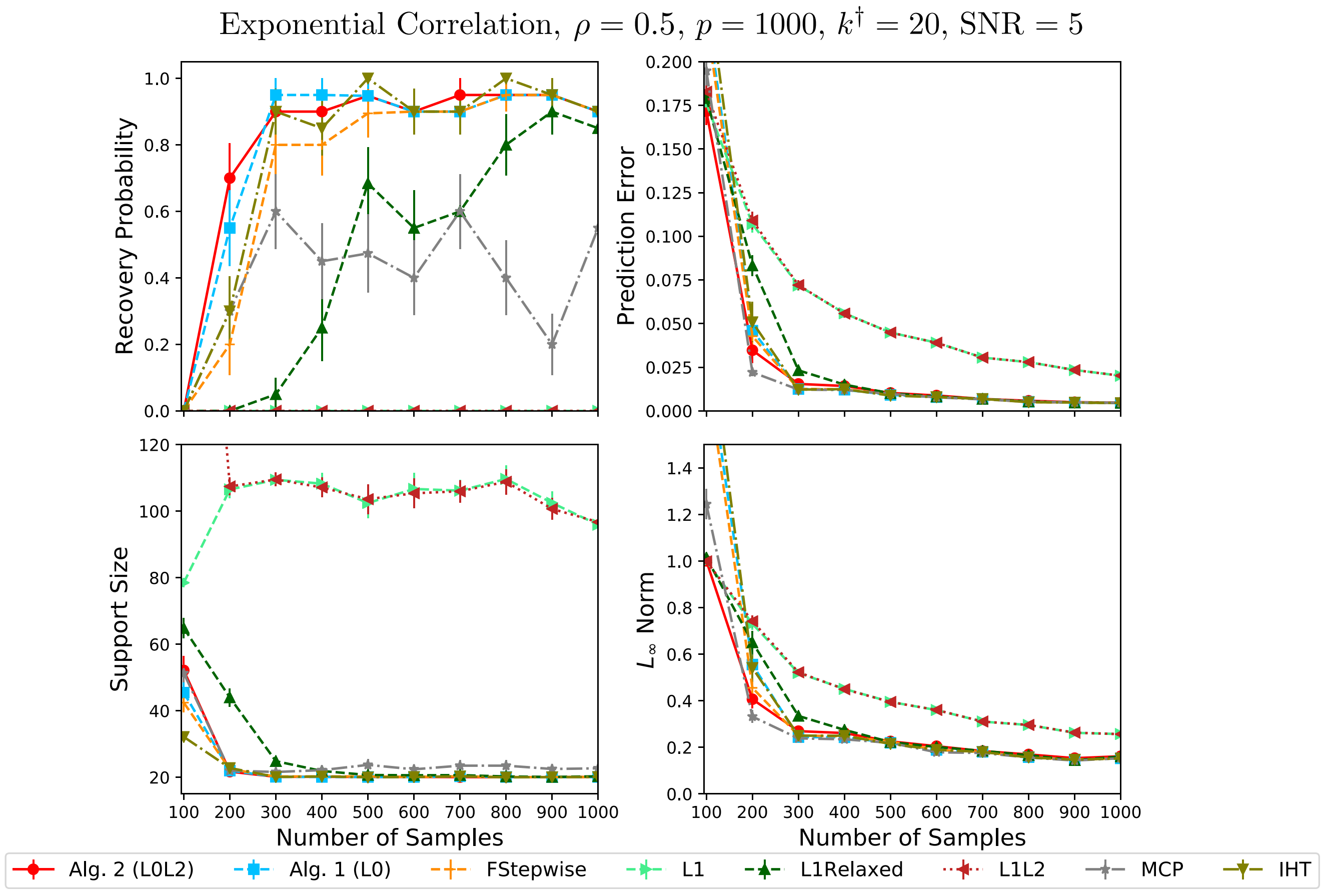
5.3.2. Experiment 2: Mild Correlation. In this experiment, we keep the same setup as the previous experiment, but we reduce the correlation parameter ρ to 0.5. In Figure 3, we show the results for Algorithm 1 applied to (L_0) and Algorithm 2 applied to (L_0L_2) versus the other competing methods. We note that our other algorithms have a similar profile (we do not include their plots for space constraints). This setup is easier from a statistical perspective, when compared with Experiment 1, where $\rho = 0.9$. Thus, we expect all the methods to perform better (overall) and display a phase transition (for recovery probability) at a smaller sample size (compared with Experiment 1). Indeed, as shown in Figure 3, Algorithm 1 (L_0) and Algorithm 2 (L_0L_2) have roughly the same profiles, and they outperform the other methods; they fully recover the true support using roughly 300 samples. The swap variants of our methods in this case do not seem to lead to significant improvements over the nonswap variants, and this further supports our hypothesis: when the statistical problem is relatively easy, Algorithm 1 works quite well—more advanced optimization (e.g., using swaps) do not seem necessary. MCP and

Figure 2. (Color online) Performance Measures as the Number of Samples n Varies Between 100 and 1,000



Notes. The top panel compares two of our methods (Algorithm 2 for the (L_0) and (L_0L_2) problems) with other state-of-the-art algorithms. The bottom panel compares Algorithms 1 and 2 for all three problems. Algorithm 2 performs significantly better than Algorithm 1 and the other methods because it does a better job at optimization.

Figure 3. (Color online) Performance Measures as the Number of Samples n Varies Between 100 and 1,000



Notes. The figure compares two of our methods (Algorithm 2 (L_0L_2) and Algorithm 1 for (L_0)) with other state-of-the-art algorithms. Algorithms 1 and 2 perform similarly in this case (in contrast to the highly correlated setting in Figure 2). Adding L_1 or L_2 regularization to (L_0) does not help in this case.

FStepwise also exhibit good performance, but they start doing full support recovery for much larger values of n; MCP does not seem to be robust. Lasso in this case never recovers the true support, and this property is inherited by relaxed Lasso, which requires at least 900 samples to match our methods in terms of support recovery.

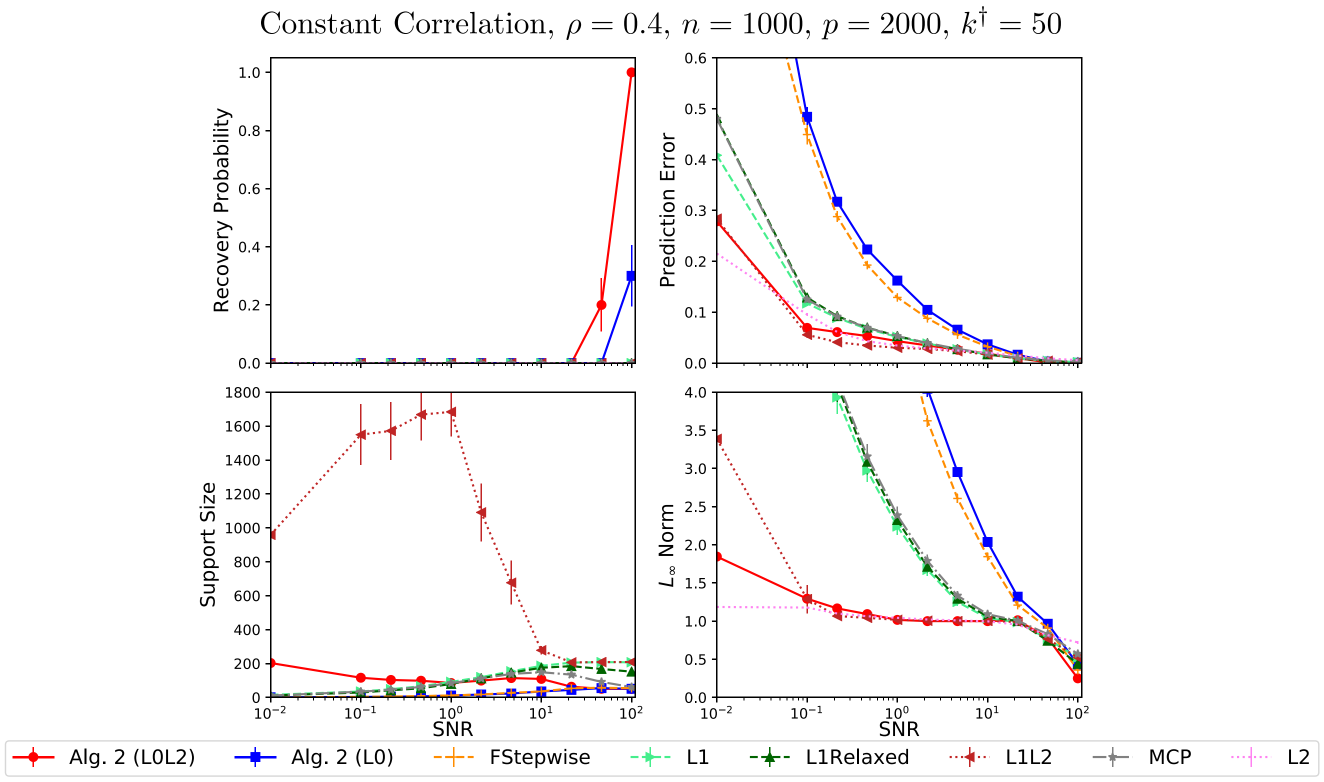
5.4. Statistical Performance for Varying SNR

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

5.4.1. Experiment 1: Constant Correlation. We generated data sets with constant correlation, $\rho = 0.4$, $n = 1000$, $p = 2000$, and $k^\dagger = 50$. We report the results for Algorithm 2 applied to the (L_0) and (L_0L_2) problems along with all the other state-of-the-art algorithms in Figure 4.

Figure 4 suggests that full support recovery is difficult for all methods (suggesting that constant correlation leads to a difficult problem). At SNR = 100, (L_0L_2) fully recovers the support, whereas (L_0) has a recovery probability of about 0.3; this suggests that the additional L_2 regularization aids the optimization performance of Algorithm 2. However, none of the other considered methods does full recovery, even for high SNR. Algorithm 2 (L_0L_2) generally exhibits excellent performance in terms of all measures across the whole SNR range. Pure (L_0) tends to overfit quickly (as SNR becomes smaller) because of the lack of shrinkage (Mazumder et al. 2017), and it selects small support sizes (such as "FStepwise"). Additional shrinkage (L_0L_2) seems to help alleviate this problem. The elastic net (L_1L_2) performs similarly to (L_0L_2) in

Figure 4. (Color online) Performance Measures as the SNR Is Varied Between 0.01 and 100



Notes. The figure compares two of our methods (Algorithm 2 applied to the (L_0) and (L_0L_2) problems) with other state-of-the-art algorithms. For low SNR levels, (L_0L_2) performs much better than (L_0) (as the latter overfits in these settings).

terms of prediction error but at the cost of very dense supports—in fact, its support size can reach up to 90% of p for SNR ~ 1, which is undesirable from the viewpoint of having a parsimonious model. We note that for low SNR values, (L_0L_2)'s prediction error is comparable with that of L_2 (for SNR = 0.01, L_2 has the best predictive performance); however, (L_0L_2) leads to much sparser models and hence has an advantage.

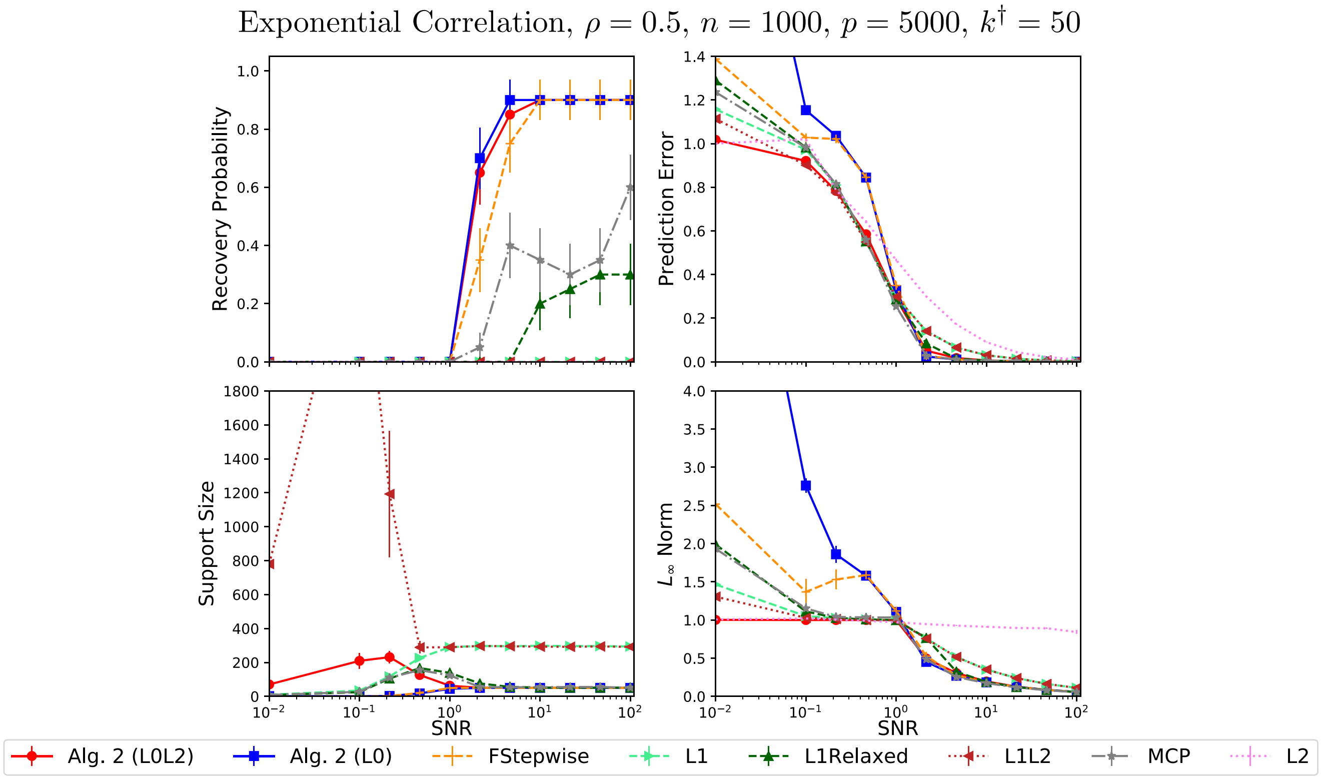
5.4.2. Experiment 2: Exponential Correlation. We generated data sets having exponential correlation with $\rho = 0.5$, $n = 1000$, $p = 5000$, and $k^{\dagger} = 50$. We report the results in Figure 5. We observe that this setup is relatively easier (from a statistical viewpoint) than the constant correlation experiment in Section 5.4.1. Thus, we observe less significant differences among the algorithms when compared with the first experiment (see Figures 4 and 5). Algorithm 2 (L_0L_2) again seems to dominate across different measures and for all SNR values. Algorithm 2 (L_0) and Algorithm 2 (L_0L_2) exhibit similar performance for high SNR. For low SNR, Algorithm 2 (L_0L_2), L_2, and L_1L_2 have the best predictive performance, though (L_0L_2) leads to the most compact models. We note that even in this relatively easy case, Lasso and elastic net never fully recover the support—MCP and relaxed Lasso also suffer in terms of full support recovery.

5.5. Comparing PSI(k) vs. FSI(k)

Here, we examine the differences among the various classes of minima introduced in the paper (i.e., CW, PSI(k), and FSI(k) minima) for the (L_0) problem. To understand the differences, we consider a relatively difficult setting with constant correlation where $\rho = 0.9$, $n = 250$, $p = 1000$, and $k^{\dagger} = 25$. We set SNR = 300 to allow for full support recovery. We generated 10 random training data sets under this setting and ran Algorithm 1 and the PSI and FSI variants of Algorithm 2 for $k \in \{1, 2, 5\}$. All algorithms were initialized with a vector of zeros. For Algorithm 2, we used Gurobi (v7.5) to solve the MIQO subproblems (15) and (25) 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 data sets. PSI(1) and FSI(1) minima lead to a significant reduction in the objective when compared with 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 contain

Figure 5. (Color online) Performance Measures as the SNR Is Varied Between 0.01 and 100



Notes. The figure compares two of our methods (Algorithm 2 applied to the (L_0) and (L_0L_2) problems) with other state-of-the-art algorithms. For SNR ≤ 1, (L_0L_2) performs significantly better than (L_0); this performance improvement vanishes for larger SNR values.

on average a large number of false positives (> 35) and few true positives; this is perhaps due to high correlations among all features, which makes the optimization task arguably very challenging. Both PSI and FSI minima increase the number of true positives significantly. A closer inspection shows that FSI minima do a better job in having fewer false positives when compared with PSI minima. This comes at the cost of solving relatively more difficult optimization problems but within reasonable computation times.

In the supplementary material, we present an experiment studying the evolution of the intermediate solutions before Algorithm 2 reaches an FSI(k) minimum. We observe that CD is effective at increasing the true positives, whereas local combinatorial search significantly reduces the false positives.

Figure 6. (Color online) Box Plots Showing the Distribution of Objective Values, Number of True Positives, and Number of False Positives for the Different Classes of Local Minima

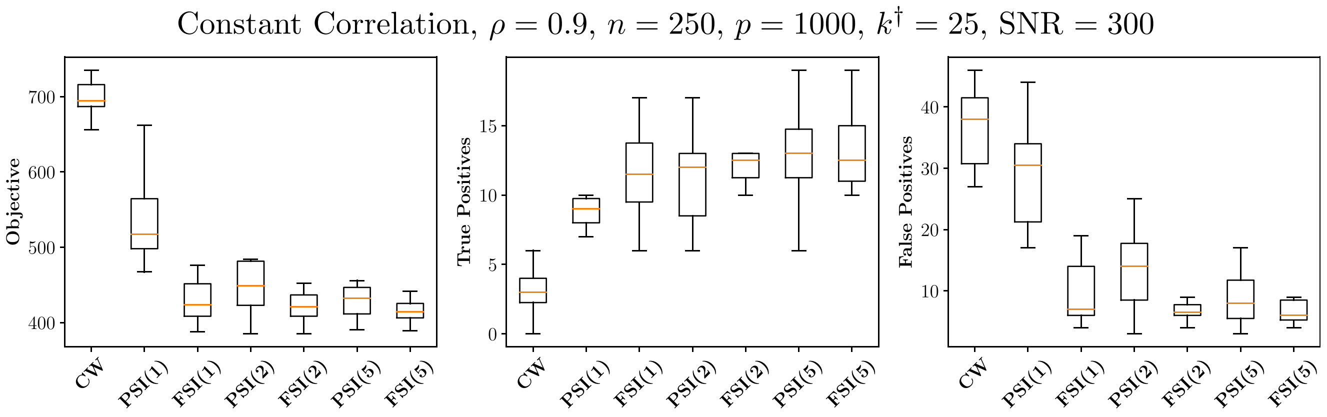


Table 1. Performance Measures for the Different Algorithms Under Settings 1 and 2

Setting 1 ($n = 1000, p = 10^5/2, \rho = 0.5$)				
Method =	$\|\beta\|_0$	TP	FP	PE $\times 10^2$
Alg. 2 (L_0)	160 ± 24	79 ± 9	81 ± 33	5 ± 1.6
Alg. 1 (L_0L_2)	**100 ± 0**	**100 ± 0**	**0 ± 0**	**0.97 ± 0.05**
Alg. 1 (L_0L_1)	**100 ± 0**	**100 ± 0**	**0 ± 0**	1 ± 0.05
L_1	808 ± 7	95 ± 1	712 ± 7	7.9 ± 0.17
L1Relaxed	602 ± 40	95 ± 1	508 ± 41	7.9 ± 0.19
MCP	102 ± 1	**100 ± 0**	2.3 ± 1	**0.97 ± 0.05**
FStepwise	216 ± 17	64 ± 7	152 ± 23	8.9 ± 1.3
Setting 2 ($n = 1000, p = 10^5, \rho = 0.3$)				
Method =	$\|\beta\|_0$	TP	FP	PE$\times 10^3$
Alg. 2 (L_0)	69 ± 18	47 ± 3	22 ± 22	1.6 ± 1
Alg. 1 (L_0L_2)	**50 ± 0**	**50 ± 0**	**0 ± 0**	**0.5 ± 0.02**
Alg. 1 (L_0L_1)	**50 ± 0**	**50 ± 0**	**0 ± 0**	**0.5 ± 0.02**
L_1	478 ± 11	**50 ± 0**	428 ± 11	4.7 ± 0.1
L1Relaxed	385 ± 12	50 ± 0.2	335 ± 13	4.4 ± 0.2
MCP	65 ± 3	**50 ± 0**	15 ± 3	3.5 ± 0.13
FStepwise	75 ± 2	**50 ± 0**	25 ± 2	1.1 ± 0.07

Notes. 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 best metrics are in bold.

5.6. Large High-Dimensional Experiments

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

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

Every experiment is performed with 10 replications, and the results are averaged. We report the results for settings 1 and 2 in Table 1.

In Table 1, Algorithm 1 for the (L_0L_1) and (L_0L_2) problems fully recovers the true support and attains the lowest prediction error. None of the other methods were able to do full support recovery; Lasso and relaxed Lasso capture most of the true positives but include a very large number of false positives. MCP comes in the middle between $(L_0L_1)/(L_0L_2)$ and Lasso—it captures all the true positives and includes few false positives. We also note that in such high SNR settings, we do not expect shrinkage (arising from the L_1/L_2 penalties) to lead to major statistical improvements. Thus, the difference in performance between (L_0) and $(L_0L_1)/(L_0L_2)$ seems to be due to the continuous regularizers that help in optimization.

5.6.2. Timings and Out-of-Sample Performance. We ran Algorithm 1 using our toolkit L0Learn, and we compared the running time and predictive performance versus glmnet and ncvreg on a variety of real and synthetic data sets. For the real data sets, there is no ground truth—we study predictive performance vis-à-vis model sparsity. We note that L0Learn, glmnet, and ncvreg are solving different optimization problems—the run times provided herein are meant to demonstrate that a main workhorse for our proposed framework is competitive when compared with efficient state-of-the-art implementations for sparse learning. Below we provide some details about the data sets:

- *House Prices:* $p = 104{,}000$ and $n = 200$. We added pairwise interactions to the popular Boston House Prices data set (Harrison and Rubinfeld 1978) to get 104 features. Then, we added random "probes" (a.k.a. noisy features) by appending to the data matrix 1,000 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 data set (He and McAuley 2016) 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) (Pedregosa et al. 2011). The validation and testing sets have 500 and 1,868 samples, respectively. We also created an augmented version of this data set where we added random probes by appending to the data matrix nine random permutations of every column to get $p = 174{,}755$.
- *U.S. Census:* $p = 55{,}537$ and $n = 5000$. We used 37 features extracted from the 2016 U.S. Census Planning Database to predict the mail-return rate[11] (Erdman and Bates 2014). We appended the data matrix with 1,500 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 data set with independent standard normal entries. We set $k^\dagger = 20$ and SNR = 10, and we performed validation and testing as described in Section 5.1.

For all real data sets, we tuned and tested on separate validation and testing sets. The timings were performed on a machine with an i7-4800MQ CPU and 16 GB 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.

The results presented in Table 2 show the following: L0Learn is faster than glmnet and ncvreg on all the considered data sets (e.g., more than twice as fast on the Amazon Reviews data set). The speedups can be attributed to the careful design of L0Learn

Table 2. Training Time (in Seconds), Out-of-Sample MSE, and the Corresponding Support Sizes for a Variety of High-Dimensional Data Sets

Toolkit	Time	MSE	$\|\|\beta\|\|_0$
Amazon Reviews ($p = 17,580$, $n = 2500$)			
Toolkit	Time	MSE $_{\times 10^2}$	$\|\|\beta\|\|_0$
glmnet (L_1)	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	1,484
Amazon Reviews (+ Probes) ($p = 174,755$, $n = 2500$)			
Toolkit	Time	MSE $_{\times 10^2}$	$\|\|\beta\|\|_0$
glmnet (L_1)	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
U.S. Census ($p = 55,537$, $n = 5000$)			
Toolkit	Time	MSE	$\|\|\beta\|\|_0$
glmnet (L$_1$)	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 (L_1)	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	MSE	$\|\|\beta\|\|_0$
glmnet (L_1)	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

Note. The training time is for obtaining a regularization path with 100 solutions. The best metrics are in bold.

(as described in Section 4) and because of 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 mean squared error (MSE) compared with 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.

6. Conclusion

We proposed new algorithms for problem (1), based on a combination of cyclic coordinate descent and local combinatorial search, and studied their convergence properties. Our algorithms are inspired by a hierarchy of necessary optimality conditions for problem (1), with solutions higher up the hierarchy being of higher quality. In terms of optimization performance, Algorithm 1 leads to better solutions and is faster than IHT and random CD. Our local optimization algorithms (Algorithm 2) often lead to further improvements over Algorithm 1. In many difficult settings, solutions from Algorithm 2 match those of global MIO solvers for problem (1) while running much faster.

Our algorithms shed interesting insights into the statistical properties of high-dimensional regression—in terms of variable selection, estimation error, and prediction error vis-à-vis problem parameters (n, p, SNR, $\beta^\dagger$, and Σ). There is no overall winner among the vanilla versions of Lasso, stepwise, or L_0 across different settings—modifications such as problem (1) or relaxed Lasso (Hastie et al. 2017) seem necessary. In low signal settings (e.g., low SNR or small n), where recovery (in terms of a small estimation error or full support recovery) seems impossible, one can hope to get a good predictive model that is also sparse. In these regimes, (L_0L_2), elastic net, and ridge typically achieve the best predictive performance, with (L_0L_2) selecting much smaller support sizes. We observe that estimators arising from problem (1) typically outperform the state-of-the-art sparse learning algorithms in terms of a combination of metrics (prediction, variable selection, and estimation) across a wide range of settings; these estimators promise to be an appealing alternative to the relaxed Lasso (Hastie et al. 2017). Our proposed algorithms allow us to uncover regimes (previously unseen because of computational limitations) where there are important differences between L_0-based estimators and existing popular algorithms (based on L_1, stepwise selection, IHT, etc.). We provide an open-source implementation of the algorithms through our toolkit L0Learn, which achieves up to a 3× speedup when compared with competing toolkits.

Acknowledgments

The authors thank the editors and reviewers for their insightful suggestions that led to an improvement of the paper.

Endnotes

[1] The L_0 (pseudo) norm of β—that is, $\|\beta\|_0$ counts the number of nonzeros in β.

[2] As we argue in Section 2.1, these also satisfy the usual notion of a local minimizer in nonlinear optimization.

[3] This convention is used for a technical reason in the context of our proof of Theorem 2.

[4] 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.

[5] This observation also appears in establishing convergence of IHT-type algorithms—see, for example, Beck and Eldar (2013), Lu (2014), and Bertsimas et al. (2016).

[6] Available on CRAN at https://CRAN.R-project.org/package=L0Learn and on GitHub at https://github.com/hazimehh/L0Learn, accessed September 12, 2019.

[7] Problem (2) usually leads to solutions with fewer nonzeros compared with Lasso and MCP-penalized regression. This also contributes to reduced run times.

[8] Because the columns of X have a 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.

[9] Our approach differs from Tibshirani et al. (2012), who derive screening rules for convex problems.

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

[11] We thank Dr. Emanuel Ben David, U.S. Census Bureau, for help on preparing this data set.

References

Beck A, Eldar YC (2013) Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM J. Optim.* 23(3):1480–1509.

Beck A, Tetruashvili L (2013) On the convergence of block coordinate descent type methods. *SIAM J. Optim.* 23(4):2037–2060.

Bertsekas D (2016) Nonlinear Programming, 3rd ed. (Athena Scientific, Nashua, NH).

Bertsimas D, Van Parys B (2017) Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. Working paper, Massachusetts Institute of Technology, Cambridge.

Bertsimas D, King A, Mazumder R (2016) Best subset selection via a modern optimization lens. *Ann. Statist.* 44(2):813–852.

Blumensath T, Davies M (2009) Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmonic Anal.* 27(3):265–274.

Breheny P, Huang J (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Ann. Appl. Statist.* 5(1):232–253.

Bühlmann P, van de Geer S (2011) *Statistics for High-Dimensional Data* (Springer, Berlin).

Erdman C, Bates N (2014) The U.S. Census Bureau mail return rate challenge: Crowdsourcing to develop a hard-to-count score. Statistics Research Report 2014-08, U.S. Census Bureau, Washington, DC.

Fan J, Li R (2001) Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.* 96(456):1348–1360.

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. *J. Statist. Software* 33(1):1–22.

Gamarnik D, Zadik I (2017) High dimensional regression with binary coefficients. estimating squared error and a phase transition. Kale S, Shamir O, eds. *Proc. 2017 Conf. Learn. Theory* (PMLR, Amsterdam), 948–953.

Greenshtein E (2006) Best subset selection, persistence in high-dimensional statistical learning and optimization under ℓ_1 constraint. *Ann. Statist.* 34(5):2367–2386.

Harrison D, Rubinfeld DL (1978) Hedonic housing prices and the demand for clean air. *J. Environ. Econom. Management* 5(1):81–102.

Hastie T, Tibshirani R, Tibshirani RJ (2017) Extended comparisons of best subset selection, forward stepwise selection, and the Lasso. Working paper, Stanford University, Stanford, CA.

Hastie T, Tibshirani R, Wainwright M (2015) *Statistical Learning with Sparsity: The Lasso and Generalizations* (CRC Press, Boca Raton, FL).

He R, McAuley J (2016) Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *Proc. 25th Internat. Conf. World Wide Web* (International World Wide Web Conferences Steering Committee, Geneva, Switzerland), 507–517.

Lu Z (2014) Iterative hard thresholding methods for l0 regularized convex cone programming. *Math. Programming* 147(1):125–154.

Mazumder R, Radchenko P (2017) The discrete Dantzig selector: Estimating sparse linear models via mixed integer linear optimization. *IEEE Trans. Inform. Theory* 63(5):3053–3075.

Mazumder R, Friedman JH, Hastie T (2011) Sparsenet: Coordinate descent with nonconvex penalties. *J. Amer. Statist. Assoc.* 106(495):1125–1138.

Mazumder R, Radchenko P, Dedieu A (2017) Subset selection with shrinkage: Sparse linear modeling when the SNR is low. Working paper, Massachusetts Institute of Technology, Cambridge.

Miller A (2002) *Subset Selection in Regression* (CRC Press, Boca Raton, FL).

Natarajan B (1995) Sparse approximate solutions to linear systems. *SIAM J. Comput.* 24(2):227–234.

Nesterov Y (2012) Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.* 22(2):341–362.

Patrascu A, Necoara I (2015) Random coordinate descent methods for ℓ_0 regularized convex optimization. *IEEE Trans. Automatic Control* 60(7):1811–1824.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. Scikit-learn: Machine learning in Python. *J. Machine Learn. Res.* 12(2011):2825–2830.

Raskutti G, Wainwright M, Yu B (2011) Minimax rates of estimation for high-dimensional linear regression over ℓ_q-balls. *IEEE Trans. Inform. Theory* 57(10):6976–6994.

Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B* 58(1):267–288.

Tibshirani RJ (2013) The lasso problem and uniqueness. *Electronic J. Statist.* 7:1456–1490.

Tibshirani R, Bien J, Friedman J, Hastie T, Simon N, Taylor J, Tibshirani RJ (2012) Strong rules for discarding predictors in lasso-type problems. *J. Roy. Statist. Soc. Ser. B* 74(2):245–266.

Tseng P (2001) Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.* 109(3):475–494.

Wainwright MJ (2009) Information-theoretic limits on sparsity recovery in the high-dimensional and noisy setting. *IEEE Trans. Inform. Theory* 55(12):5728–5741.

Zhang C-H (2010) Nearly unbiased variable selection under minimax concave penalty. *Ann. Statist.* 38(2):894–942.

Zhang C-H, Zhang T (2012) A general theory of concave regularization for high-dimensional sparse estimation problems. *Statist. Sci.* 27(4):576–593.

Zhang Y, Wainwright M, Jordan MI (2014) Lower bounds on the performance of polynomial-time algorithms for sparse linear regression. *Proc. 2014 Conf. Learn. Theory* (PMLR, Amsterdam), 921–948.

Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. *J. Roy. Statist. Soc. Ser. B* 67(2):301–320.

Hussein Hazimeh is a third-year PhD candidate at the MIT Operations Research Center, where he is advised by Rahul Mazumder. He received his MS in computer science from the

University of Illinois at Urbana–Champaign working with ChengXiang Zhai. His main research interests lie at the intersection of optimization and machine learning. In particular, he is working on developing scalable algorithms for sparse and interpretable learning models.

Rahul Mazumder is the Robert G. James Career Development Associate Professor in the Operations Research and Statistics group at MIT Sloan School of Management. He is a recipient of the Office of Naval Research Young Investigator Award 2018. His research interests are at the intersection of statistical machine learning; large-scale optimization; analytics; and their applications in recommender systems, computational social science, and computational biology, among others.