GROUPED VARIABLE SELECTION WITH DISCRETE OPTIMIZATION: COMPUTATIONAL AND STATISTICAL PERSPECTIVES

BY HUSSEIN HAZIMEH¹, RAHUL MAZUMDER¹,⋆ AND PETER RADCHENKO²,†

¹Massachusetts Institute of Technology hazimeh@google.com; ⋆rahulmaz@mit.edu
²University of Sydney †peter.radchenko@sydney.edu.au

We present a new algorithmic framework for grouped variable selection that is based on discrete mathematical optimization. While there exist several appealing approaches based on convex relaxations and nonconvex heuristics, we focus on optimal solutions for the \( \ell_0 \)-regularized formulation, a problem that is relatively unexplored due to computational challenges. Our methodology covers both high-dimensional linear regression and nonparametric sparse additive modeling with smooth components. Our algorithmic framework consists of approximate and exact algorithms. The approximate algorithms are based on coordinate descent and local search, with runtimes comparable to popular sparse learning algorithms. Our exact algorithm is based on a standalone branch-and-bound (BnB) framework, which can solve the associated mixed integer programming (MIP) problem to certified optimality. By exploiting the problem structure, our custom BnB algorithm can solve to optimality problem instances with \( 5 \times 10^6 \) features and \( 10^3 \) observations in minutes to hours – over 1000 times larger than what is currently possible using state-of-the-art commercial MIP solvers. We also explore statistical properties of the \( \ell_0 \)-based estimators. We demonstrate, theoretically and empirically, that our proposed estimators have an edge over popular group-sparse estimators in terms of statistical performance in various regimes. We provide an open source implementation of our proposed framework.

1. Introduction. Sparsity plays a ubiquitous role in modern statistical regression, especially when the number of predictors is large relative to the number of observations. In this paper, we focus on the case where predictors have a natural group structure. Typical examples where such a structure appears are models with multilevel categorical predictors and models that represent nonlinear effects of continuous variables using basis functions [16, 71, 28]. Grouping may also arise from scientifically meaningful prior knowledge about the collection of the predictor variables. More specifically, we consider the usual linear regression framework with response \( y_{n \times 1} \) and model matrix \( X_{n \times p} = [x_1, \ldots, x_p] \). We suppose that the \( p \) predictors are divided into \( q \) pre-specified, non-overlapping groups. For a given \( \beta \in \mathbb{R}^p \) and each \( g \in \{1, \ldots, q\} \), we denote by \( \beta_g \) the sub-vector of \( \beta \) whose coefficients correspond to the predictors in group \( g \). Following the traditional approach in high-dimensional regression, we assume that few of the regression coefficients are nonzero, i.e., the model is sparse. This leads to a natural generalization of the classical best subset selection problem in linear regression [49, 10] to the group setting:

\[
\min_{\beta} \| y - X\beta \|_2^2 + \lambda_0 \sum_{g=1}^q \mathbf{1}(\beta_g \neq 0),
\]

MSC2020 subject classifications: 62Jxx,62G05,90C11,90C06.

Keywords and phrases: Group Variable Selection, Sparsity, Nonparametric Additive Models, \( \ell_0 \) regularization, Mixed Integer Programming, Branch and Bound Algorithms.
where \( I(\cdot) \) is the indicator function, and \( \lambda_0 \) is a non-negative regularization parameter that controls the number of nonzero groups selected. We will refer to Problem (1) as the Group \( \ell_0 \) problem.

Problem (1) is NP-Hard [52] and poses computational challenges. A rich body of prior work explores sparsity-inducing methods to obtain approximate solutions to (1). Popular methods include: convex optimization based procedures, such as Group Lasso [71], which is a generalization of the Lasso approach [62] to the grouped setting, and local solutions to non-convex optimization problems arising from group-nonconvex regularizers, such as SCAD, MCP and others [73, 32]. Despite the appeal of these approaches, the statistical and computational aspects of optimal solutions to (1) remain to be understood at a deeper level. To this end, we aim to advance the computational frontiers of Problem (1) using novel tools from discrete optimization. Our proposed combinatorial optimization-based algorithms are scalable. In particular, they can deliver optimal solutions to (1) for instances that are much larger than state-of-the-art approaches. We also develop a better understanding of the statistical properties of Problem (1) both theoretically and empirically.

**Computation.** We propose new algorithms based on combinatorial optimization for solving Problem (1) and its variants. First we present approximate algorithms: they deliver high-quality solutions using a combination of cyclic coordinate descent and local combinatorial optimization [29]. These algorithms have runtimes comparable to popular approaches for grouped variable selection (for example, Group Lasso or MCP), but deliver solutions with considerably improved statistical performance (for example, in terms of prediction and variable selection), as we demonstrate in our experiments. Our approximate algorithms deliver good-quality feasible solutions to (1) but are unable to certify (global) optimality of solutions via matching lower bounds on the optimal objective value of (1). Certifying optimality is not only important from a methodological perspective but can also be beneficial in practice for mission-critical applications. For example, having certifiably optimal solutions can engender trust and provide transparency in consequential applications such as healthcare. Thus, we propose a new tailored branch-and-bound based optimization framework for solving (1) to certifiable optimality.

In our exact (global optimization) framework, we formulate the Group \( \ell_0 \) problem as a Mixed Integer Program (MIP). However, in a departure from earlier work [10, 11], we propose a custom branch-and-bound (BnB) algorithm to solve the MIP. Indeed, MIP-based techniques have gained considerable traction recently to solve to (near) optimality the best subset selection problem, where all groups are of size one [10, 11, 46, 47, 29, 69, 31]. All these works, with the exception of [31], leverage capabilities of powerful commercial MIP solvers such as Gurobi and CPLEX. These solvers have gained wide adoption in the past two decades due to major advances in algorithms and software development [12, 35]. However, these general-purpose solvers may take several hours to certify optimality on small instances (for example, with \( p = 1000 \)). In contrast, our custom BnB algorithm exploits problem-specific structure to scale to much larger instances. For example, it can solve to optimality instances with \( p = 5 \times 10^6 \) – this is 1000 times larger than what can be handled using Gurobi’s MIP-solver. Our BnB algorithm generalizes to the grouped setting the approach of [31] developed for the best subset selection problem.

**Statistical properties.** Statistical properties of Group Lasso have been extensively studied, and it has been shown, both empirically and theoretically, that it performs well in sparse high-dimensional settings [18, 2, 51, 34, 68, 41, 53], under certain assumptions on the data. However, Group Lasso also has its shortcomings, similar to those of Lasso in high dimensional linear regression [10, 16, 29]. More specifically, depending on the penalty weight, the resulting model may either be very dense or, alternatively, comes with overly shrunk nonzero
coefficients. This problem is aggravated when the groups are correlated with each other, as Group Lasso tends to bring in all of the correlated groups in lieu of searching for a more parsimonious model. For further discussions of these issues in the special case of Lasso see, for example, [74, 45, 16, 10], and the references therein. In this paper, we demonstrate, both empirically and theoretically, that the Group \( \ell_0 \) methodology has advantages over its Group Lasso counterpart in a variety of regimes. In particular, as a consequence of directly controlling the sparsity level in the optimization problem, our framework leads to substantially sparser models under similar data fidelity. Moreover, in many scenarios where the predictors are highly correlated, our approach performs better in terms of both estimation and prediction.

Additive models with \( \ell_0 \)-sparsity. In addition to linear models, we also study an important example of regression with group structure that arises in high-dimensional sparse additive modeling [28, 27]. Here, we estimate a nonparametric multivariate regression function in \( q \) covariates, \((x_1, \ldots, x_q)\), which we model as a sparse additive sum of the form \( \sum_{j \in S} f_j(x_j) \), where \( S \subset \{1, \ldots, q\} \). In this setting, each group generally corresponds to the basis representation of a given additive component, one for each of the \( q \) predictors. Because the groups are allowed to be large, additional regularization needs to be imposed, typically in the form of a roughness type penalty on the regression functions. A number of successful Group Lasso-based approaches have been proposed and analyzed in this setting — see, for example, [48, 58, 33, 37, 56, 72] and the references therein. To our knowledge, this is the first paper to explore statistical and computational aspects of Group \( \ell_0 \)-based formulations in the context of sparse additive modeling. We show theoretically and empirically that Group \( \ell_0 \) based methods enjoy certain statistical advantages when compared to the Group Lasso-based counterparts.

Contributions. The focus of this paper is on Problem (1) and the sparse additive modeling problem (which can be formulated as a variant of Problem (1), as we discuss in Section 2). Our main contributions for these two problems can be summarized as follows:

- We develop fast approximate algorithms, based on first-order and local combinatorial optimization. We establish convergence guarantees for these algorithms and provide useful characterizations of the corresponding local minima. Our experiments indicate that these algorithms can have an edge in terms of statistical performance over popular alternatives for grouped variable selection.
- We present mixed integer second order cone program (MISOCP) formulations for the Group \( \ell_0 \)-based estimators; and design a novel specialized, nonlinear branch-and-bound (BnB) framework for solving the MISOCP to global optimality. Our custom BnB solver can handle instances with \( 5 \times 10^6 \) features and \( 10^3 \) observations — more than a 1000 times larger than what can be handled by state-of-the-art commercial MISOCP solvers.
- We establish non-asymptotic prediction and estimation error bounds for our proposed estimators, for both the high-dimensional linear regression and sparse additive modeling problems. We show that under the assumption of sparsity, these error bounds compare favorably with the ones for Group Lasso.
- We demonstrate empirically that our approach appears to outperform the state of the art (for example, Group Lasso and available algorithms for nonconvex penalized estimators) in a variety of high-dimensional regimes and under different statistical metrics (for example, prediction, estimation, and variable selection). We provide open-source implementations of both our approximate and exact optimization algorithms\(^1\).

\(^1\)Our open-source code is available on github at [https://github.com/hazimehh/L0Group](https://github.com/hazimehh/L0Group).
Organization. In Section 2, we present formulations for the Group $\ell_0$ and sparse additive modeling problems. Section 3 presents approximate algorithms based on first-order and local combinatorial optimization algorithms. Then, in Section 4, we present our exact MIP algorithm. Statistical properties of our approach are investigated in Section 5. Section 6 presents computational experiments. Technical proofs and additional computational details are provided in the supplement.

Notation. For any non-negative integer $k$, we denote the set $\{1, \ldots, k\}$ by $[k]$. The complement of a set $A$ is denoted by $A^c$. We denote the index sets corresponding to the $q$ groups of predictors by $G_g$, for $g \in [q]$ so that $\cup_{g=1}^{q} G_g = [p]$ and $G_g \cap G_{\ell} = \emptyset$ for all $g \neq \ell$. For a vector $\theta$, we use the notation $\text{Supp}(\theta)$ to denote the group support, i.e., $\text{Supp}(\theta) = \{g \mid \theta_g \neq 0\}$. We also define a measure of $\ell_0$-group sparsity (i.e., number of nonzero groups): $G(\theta) := \sum_{g=1}^{q} 1(\theta_g \neq 0)$. We denote the gradient of a scalar-valued function, say $J(\theta)$, by $\nabla J(\theta)$. Moreover, we use the notation $\nabla_{\theta_g} J(\theta)$ to refer to the subvector of $\nabla J(\theta)$ corresponding to the variables in $\theta_g$. Vectors and matrices are denoted in boldface.

2. Optimization problems considered. In this section, we present optimization formulations for the Group $\ell_0$ approach (and its variants), as well as the $\ell_0$-sparse additive function estimation approach.

2.1. Group $\ell_0$ with ridge regularization. The algorithms discussed in this paper apply to the Group $\ell_0$ estimator (1) with an optional ridge regularization term:

\[
\min_{\beta} \|y - X\beta\|_2^2 + \lambda_0 \sum_{g=1}^{q} 1(\beta_g \neq 0) + \lambda_2 \|\beta\|_2^2, \tag{2}
\]

where $\lambda_0 > 0$ controls the number of selected groups, and $\lambda_2 \geq 0$ controls the strength of the ridge regularization. Our proposed algorithms apply to both settings: $\lambda_2 = 0$ and $\lambda_2 > 0$ in Problem (2). The choice of the ridge term in (2) is motivated by earlier work in the context of best-subset selection [47, 29], which suggest that when the signal-to-noise ratio (SNR) is low, additional ridge regularization can improve the prediction performance of best-subset selection (both theoretically and empirically). Additionally, as discussed in Section 4.2, the choice $\lambda_2 > 0$, allows for deriving stronger MIP formulations by appealing to perspective formulations [24, 26].

2.2. Nonparametric additive models with $\ell_0$-sparsity. In the multivariate setting, estimating the conditional mean function $\mathbb{E}(y|x) = f(x_1, \ldots, x_q)$ becomes notoriously difficult, due to curse of dimensionality. To overcome this problem, additive approximation schemes [27] are commonly used as an effective methodology: $f(x) = \sum_{j=1}^{q} f_j(x_j)$. A popular approach [see, for example, 66] is to choose $f_j$ from some smooth functional class $\mathcal{C}_j$, such as the class of twice continuously differentiable functions. Given the observations $(y_i, x_i), \ i \in [n]$, the additive model $f(x)$ can be estimated by solving the following optimization problem:

\[
\min_{f} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{q} f_j(x_{ij}))^2 + \lambda \sum_{j=1}^{q} \text{Pen}(f_j), \tag{3}
\]

where $\text{Pen}(f_j)$ is a roughness penalty that controls the amount of smoothness in function $f_j$.

A key ingredient in the additive function fitting framework is the estimation of a univariate smooth regression function based on observations $(y_i, u_i), \ i \in [n]$. Suppose, for simplicity, that the $u_i$s are distinct and $u_i \in [0,1]$ for all $i$. For illustration, let us take
Pen\((g) = \int_0^1 (g''(u))^2 du\). Then, the solution to the corresponding (infinite dimensional) univariate problem is of the form: \(g(u) = \alpha_0 + \alpha_1 u + \sum_{j=1}^n \gamma_j N_j(u)\), where \(N_j(u)\) are some cubic spline basis functions, such as truncated power series functions, natural cubic splines or the B-spline basis functions, with knots chosen at the distinct data points \(u_i, i \in [n]\). Note that \(\int_0^1 (g''(u))^2 du = \gamma'\Omega\gamma\), where \(\Omega\) is an \(n \times n\) positive definite matrix with the elements \(\omega_{ij} = \int_0^1 N''_j(u)N''_j(u)du\). If we refer to the corresponding functional class as \(C\), define the elements of \(g\) as \(g_i := g(u_i)\), for \(i = 1, \ldots, n\), and let \(\|g\|_C^2 := \gamma'\Omega\gamma\), then the univariate optimization problem is equivalent to

\[
(4) \quad (\hat{g}_1, \ldots, \hat{g}_m) = \hat{g} \in \arg \min \|y - g\|_2^2 + \lambda\|g\|_C^2.
\]

Problem (4) is a generalized least squares problem in \((\alpha_0, \alpha_1, \gamma)\). A direct extension to the additive model setting is given by the following formulation:

\[
(5) \quad \min \|y - \sum_{j=1}^q f_j\|_2^2 + \lambda \sum_{j=1}^q \|f_j\|_{C_j}^2,
\]

where we minimize over \(f_j \in C_j\) for all \(j\), and \(f_j = (f_j(x_{ij}), \ldots, f_j(x_{nj}))\).

We wish to impose sparsity on the additive components \(f_j, j \in [q]\), which naturally leads to the following optimization problem:

\[
(6) \quad \min \|y - \sum_{j=1}^q f_j\|_2^2 + \lambda_0 \sum_{j=1}^q 1(f_j \neq 0) + \lambda \sum_{j=1}^q \|f_j\|_{C_j}^2.
\]

We note that the choice \(\operatorname{Pen}(f_j) = \sqrt{f_j''(u)^2 du}\) leads to the optimization problem

\[
(7) \quad \min \|y - \sum_{j=1}^q f_j\|_2^2 + \lambda_0 \sum_{j=1}^q 1(f_j \neq 0) + \lambda \sum_{j=1}^q \|f_j\|_{C_j}^2.
\]

Problems (6) and (7) are close cousins and result in similar estimators. The terms \(\sum_j \|f_j\|_{C_j}\) encourage smoothness in each of the additive components, while the sum of indicators directly controls the number of included predictors. In Section 5, we establish theoretical error bounds for the estimator that corresponds to Problem (7).

**Connections with Group Lasso-type penalization schemes.** For Grouped Lasso-type penalization schemes, the choice of the penalty becomes rather subtle. Problem (3) with \(\operatorname{Pen}(f_j) = \|f_j\|_C^2\) does not induce sparsity in \(\|f_j\|_{C_j}\)'s for finite \(\lambda\). Alternatively, the choice \(\operatorname{Pen}(f_j) = \|f_j\|_{C_j}\) does result in several components \(\|f_j\|_{C_j}\) being set to zero when \(\lambda\) is large. Note, however, that \(\|f_j\|_{C_j} = 0\) does not imply \(f_j = 0\). This is because \(\|f_j\|_{C_j}\) is a seminorm that is not affected by the linear components of \(f_j\). To set \(f_j = 0\) one needs to include the linear components into the penalty. To overcome these limitations, alternatives have been proposed – here we mention some penalization schemes that are used to encourage selection and smoothness. One possible choice \([48]\) is \(\operatorname{Pen}(f_j) = \sqrt{\|f_j\|_2^2 + \lambda I\|f_j\|_{C_j}^2}\), where \(\|f_j\|_2\) denotes the usual \(\ell_2\) norm of the vector \(f_j\). The corresponding penalization term is \(\lambda \sum_j \operatorname{Pen}(f_j)\), and, hence, the parameters \(\lambda\) and \(\lambda'\) jointly control smoothness and sparsity. The sum of \(\|f_j\|_2^2\) and \(\lambda'\|f_j\|_{C_j}^2\) leads to double penalization, thereby potentially resulting in unwanted shrinkage that may interfere with variable selection. Similar issues arise with the choices \(\operatorname{Pen}(f_j) = \|f_j\|_2 + \lambda'\|f_j\|_{C_j}\), considered in \([16]\), and \(\operatorname{Pen}(f_j) = \sqrt{\|f_j\|_2^2 + \lambda'\|f_j\|_{C_j}^2 + \lambda\|f_j\|_{C_j}^2}\), which appears in \([48]\).
Thus, the choice of Pen\((f_j)\) plays an important role in obtaining sparsity for Lasso-type regularization methods. In contrast, the levels of smoothness and sparsity are controlled separately in the \(\ell_0\)-formulations: Problems (6) and (7). Group Lasso-type penalization schemes may be interpreted as convex relaxations of the \(\ell_0\)-penalty appearing in Problem (7), as discussed in the Supplement A.

**Other choices of smooth function classes.** We note that the above framework, where each additive component is taken to be a cubic spline, can be generalized to more flexible smooth nonparametric models, depending upon the choice of Pen\((\cdot)\) and the functional classes \(C_j\)'s. For example, one may consider the class of functions that are \(\tau\) times continuously differentiable, together with the choice Pen\((f_j) = \int f_j^{(\tau)}(u)du\), where \(f_j^{(\tau)}\) denotes the \(\tau\)th derivative of \(f_j\) - solutions to these problems are given by splines of order \(\tau\) [66].

Another popular paradigm pursued in several works [37, 40, 57] is the Reproducing Kernel Hilbert Space (RKHS) framework, wherein every \(C_j\) is taken to be a Hilbert space encouraging some form of smoothness on \(f_j\). Here, Pen\((f_j) = \|f_j\|_{\mathcal{K}_j}\) is an appropriate Hilbert space norm.

2.3. **General problem formulation considered in this paper.** Our focus in this paper is on Problem (2) and the sparse additive modeling problems defined in (6) and (7). These three problems can all be formulated as follows:

\[
\min_{\beta} \beta'P\beta + \langle a, \beta \rangle + \lambda_0 G(\beta) + \lambda_1 \sum_{g=1}^{q} \|P_g \beta_g\|_2,
\]

for suitable choices of \(a, P \succ 0, P_g \succ 0, g \in [q]\), where we recall that \(G(\beta) := \sum_{g=1}^{q} 1(\beta_g \neq 0)\). The term \(\sum_{g=1}^{q} \|P_g \beta_g\|_2\) is only used for the sparse additive modeling problem in (7). Problems (1) and (6) can be obtained by setting \(\lambda_1 = 0\) and choosing \(P\) and \(a\) appropriately.

To simplify the presentation, we apply a change of variable in Problem (8): \(\theta_g = P_g^{\frac{1}{2}} \beta_g\) for \(g \in [q]\). This leads to the following equivalent problem:

\[
\min_{\theta} h(\theta) := \theta' W \theta + \langle b, \theta \rangle + \lambda_0 G(\theta) + \lambda_1 \sum_{g=1}^{q} \|\theta_g\|_2 , \tag{9}
\]

for appropriately defined\(^2\) \(W\) and \(b\). For notational convenience, we define:

\[\ell(\theta) := \theta' W \theta + \langle b, \theta \rangle\] \text{ and } \[\Omega(\theta) := \lambda_0 G(\theta) + \lambda_1 \sum_{g=1}^{q} \|\theta_g\|_2.\]

Our algorithmic development will focus on Problem (9).

**Overview of our algorithms:** Problem (9) is nonconvex due to the discontinuity in \(G(\theta)\). In Section 3, we design fast algorithms that can obtain high-quality approximate solutions for this problem. In Section 4, we develop an exact algorithmic framework, based on a custom MIP solver, which obtains certifiably optimal solutions to (9). Our algorithm constructs: (i) a sequence of feasible solutions, whose objective values are valid upper bounds, and (ii) a sequence of lower bounds (a.k.a. dual bounds). As our BnB algorithm progresses, these

\(^2\)Let \(D_1 = \text{diag}(P_{1}^{\frac{1}{2}}, \ldots, P_{q}^{\frac{1}{2}})\) be a block diagonal matrix. Then \(W = D_1^{-1}PD_1^{-1}\), and \(b = D_1^{-1}a\).
upper and lower bounds converge towards the optimal objective of Problem (9). The solver terminates and certifies optimality when the upper and lower bounds match. Our experiments indicate that high-quality initial solutions, as available from the algorithms presented in Section 3, can significantly speed up convergence and reduce memory requirements in our BnB algorithm.

3. Approximate Algorithms. In this section, we develop fast approximate algorithms to obtain high quality local minimizers for Problem (9). While these algorithms do not deliver certificates of optimality (via dual bounds), they attain nearly-optimal (and at times optimal) solutions to many statistically challenging instances, in running times comparable to group Lasso-based algorithms.

A main workhorse of our approximate algorithms is a nonstandard application of cyclic block coordinate descent (BCD) to the discontinuous objective function (9). We draw inspiration from the appealing scalability properties of coordinate descent in sparse learning problems [see, for example, 25, 4, 29]. Our second algorithm is based on local combinatorial search and is used to improve the quality of solutions obtained by BCD. We establish convergence guarantees for these two algorithms.

Our algorithms arise from studying necessary optimality conditions for Problem (9). To this end, we show that the quality of solutions obtained by BCD are of higher quality than local solutions corresponding to the popular proximal gradient descent (PGD) [55] algorithm. The local minimizers corresponding to local combinatorial search form a smaller subset of those available from BCD. In this section, we establish the following hierarchy among the classes of local minima:

\[(10) \quad \text{Global Minima} \subseteq \text{Local Search Minima} \subseteq \text{BCD Minima} \subseteq \text{PGD Minima}.\]

Above, PGD minima correspond to the fixed points of the PGD algorithm; they include all the fixed points of our proposed BCD algorithm. As we move from right to left in the above hierarchy, the classes become smaller, i.e., they impose stricter necessary optimality conditions. At the top of the hierarchy we have the global minimizers of the problem, which can be obtained using our exact MIP-based framework (we discuss this in Section 4). Our approximate algorithms are inspired by recent work [29] on the sparse regression problem, but the approach presented here has notable differences. In particular, the coordinate descent algorithm in [29] performs exact minimization per coordinate, which can be computationally expensive when extended to the group setting. Thus, our proposed BCD algorithm performs inexact minimization per group. In addition, the presence of $\ell_2$ norms in our objective function makes the analysis for the rate of convergence for our algorithm different.

3.1. Block Coordinate Descent. We present a cyclic BCD algorithm to obtain good feasible solutions to Problem (9) and establish convergence guarantees. We first introduce a useful upper bound for $\ell(\theta)$. For every $g \in [q]$, we define $S_g = \{(\theta, \tilde{\theta}) \mid \theta_i = \tilde{\theta}_i, \forall i \in [q] \text{ s.t. } i \neq g\}$. By the Block Descent Lemma [8], the following upper bound holds for every $g \in [q]$:

\[(11) \quad \ell(\theta) \leq \ell(\tilde{\theta}) + \langle \nabla_{\theta_g} \ell(\tilde{\theta}), \theta_g - \tilde{\theta}_g \rangle + \frac{L_g}{2} \|\theta_g - \tilde{\theta}_g\|^2_2, \quad \forall (\theta, \tilde{\theta}) \in S_g,\]

In practice, MIP solvers terminate when the difference between the upper and lower bounds are below a small, user-defined threshold.

Though PGD is popularly used in the context of convex optimization problems, it also leads to useful algorithms for nonconvex sparse learning problems. In particular, PGD for our problem can be viewed as a generalization of the iterative hard thresholding (IHT) algorithm [13] to the group setting.
where $L_g$ is the “group-wise” Lipschitz constant of $\nabla \ell(\theta)$, i.e., $L_g$ is a constant which satis-
ﬁes: $\|\nabla_{\theta_s} \ell(\theta) - \nabla_{\theta_s} \ell(\tilde{\theta})\|_2 \leq L_g \|\theta_g - \tilde{\theta}_g\|_2$, for all $(\theta, \tilde{\theta}) \in S_g$. Since $\ell(\theta)$ is a quadratic function, $L_g = 2\sigma_{\max}(W_g)$, where $W_g$ is the submatrix of $W$ with columns and rows restricted to group $g$, and $\sigma_{\max}(\cdot)$ denotes the largest eigenvalue.

Cyclic BCD sequentially minimizes the objective of (9) with respect to one group of vari-
ables while the other groups are held ﬁxed. Let $\theta^l$ be the iterate obtained by the algorithm after the $l$-th iteration. Then, in iteration $l + 1$, the variables in a group $g$ (say), are updated while the other groups are held ﬁxed. Speciﬁcally, we have $(\theta^l, \theta^{l+1}) \in S_g$. Using (11) with $\theta = \theta^l$, $L_g > L_g$ and adding $\Omega(\theta)$ to both sides we get:

\[
(12) \quad h(\theta) \leq \tilde{g}(\theta; \theta^l) := \ell(\theta^l) + \langle \nabla_{\theta_g} \ell(\theta^l), \theta_g - \theta^l \rangle + \frac{L_g}{2} \|\theta_g - \theta^l\|_2^2 + \Omega(\theta).
\]

Note that the left hand side of (12) is the objective function of Problem (9). We obtain $\theta^{l+1}_g$ by minimizing the upper bound on our objective, $\tilde{g}(\theta; \theta^l)$, with respect to $\theta_g$:

\[
(13) \quad \theta^{l+1}_g \in \arg\min_{\theta_g} \tilde{g}(\theta; \theta^l) = \arg\min_{\theta_g} \frac{L_g}{2} \|\theta_g - \left(\theta^l_g - \frac{1}{L_g} \nabla_{\theta_g} \ell(\theta^l)\right)\|_2^2 + \Omega(\theta_g).
\]

Although nonconvex, the minimization problem in (13) admits a closed-form solution, which can be obtained via the operator $H : \mathbb{R}^u \to \mathbb{R}^u$ deﬁned as follows:

\[
(14) \quad H(z; \lambda; \tilde{L}_g) = \begin{cases} \frac{z}{\|z\|_2} \left[\|z\|_2 - \frac{\lambda_1}{L_g}\right] & \text{if } \|z\|_2 > \frac{2\lambda_0}{L_g} + \frac{\lambda_1}{L_g} \\ 0 & \text{otherwise} \end{cases}
\]

where $\lambda = (\lambda_0, \lambda_1)$. It can be readily seen that an optimal solution of (13) is given by $H(z; \lambda; \tilde{L}_g)$, where $z = \theta^l_g - \frac{1}{L_g} \nabla_{\theta_g} \ell(\theta^l)$. Below we summarize our proposed cyclic BCD algorithm.

\textbf{Algorithm 1: Cyclic Block Coordinate Descent (BCD)}

- \textbf{Input:} Initialization $\theta^0$ and $\tilde{L}_g$ for every $g \in [q]$.
- \textbf{Repeat} Steps 1, 2 for $l = 0, 1, 2, \ldots$ until convergence:
  1. $g \leftarrow 1 + (l \mod q)$ and $\theta^{l+1}_j \leftarrow \theta^l_j$ for all $j \neq g$
  2. $\theta^{l+1}_g \leftarrow H(z; \lambda; \tilde{L}_g)$, where $z = \theta^l_g - (1/\tilde{L}_g) \nabla_{\theta_g} \ell(\theta^l)$.

\textbf{Convergence Analysis.} To establish convergence of the sequence $\{\theta^l\}$ in Algorithm 1, we make use of the following assumption.

\textbf{Assumption 1.} At least one of the following conditions holds:

(a) Strong Convexity: $W > 0$.
(b) Restricted Strong Convexity: Let $\hat{\theta}$ be a (Group Lasso) solution deﬁned as $\hat{\theta} \in \arg\min_{\theta} \ell(\theta) + \lambda_1 \sum_{g=1}^q \|\theta_g\|_2$. Let $k = \max_{\theta} \{\|\theta\|_0 \mid G(\theta) \leq G(\hat{\theta})\}$. Every collection of $k$ columns in $W$ are linearly independent, and the initial solution $\theta^0$ (in Algorithm 1) satisﬁes $h(\theta^0) \leq h(\hat{\theta})$.

Assumption 1(a) holds if a ridge regularization term is used, i.e., it holds for Problem (2) with $\lambda_2 > 0$. Assumption 1(b) is less restrictive because we can have $W \succeq 0$. Suppose that for some non-negative integer $u$, every set of $u$ columns in $W$ are linearly independent. Then, in the Group Lasso problem (deﬁned in Assumption 1(b)), $\lambda_1$ can be chosen sufﬁciently large so that some Group Lasso solution $\hat{\theta}$ satisﬁes $k \leq u$. If $\hat{\theta}$ is used to initialize Algorithm 1, then Assumption 1(b) is satisﬁed.
The following theorem establishes a linear convergence guarantee for the sequence generated by Algorithm 1.

**Theorem 1.** Let \( \{ \theta^l \} \) be the sequence generated by Algorithm 1 and suppose that Assumption 1 holds. Then,

1. The group support stabilizes after a finite number of iterations, i.e., there exists an integer \( K \) and a support \( S \subseteq [q] \) such that \( \text{Supp}(\theta^K) = S \) for all \( l \geq K \).
2. The sequence \( \{ \theta^l \} \) converges to a solution \( \theta^* \), with \( \text{Supp}(\theta^*) = S \) (as defined in Part 1), satisfying:
   \[
   \theta^*_S \in \arg\min_{\theta_S} \ell(\theta_S) + \lambda_1 \sum_{g \in S} \| \theta_g \|_2 
   \]
   for all \( l \geq K \).
3. The function \( \theta^*_S \mapsto \ell(\theta^*_S) \) is strongly convex with a strong convexity parameter \( \sigma_S > 0 \). Let \( \hat{L}_g \) be the Lipschitz constant of \( \nabla_{\theta^*_g} \ell(\theta^*_S) \). Define \( \hat{L}_{\max} = \max_{g \in S} \hat{L}_g + 2\lambda_1 \) and \( \hat{L}_{\min} = \min_{g \in S} \hat{L}_g + 2\lambda_1 \). Then, for \( l \geq K \), the following holds:
   \[
   h(\theta^{l+1}g) - h(\theta^*) \leq \left( 1 - \frac{\sigma_S}{\eta} \right) \left( h(\theta^g) - h(\theta^*) \right),
   \]
   where \( \eta = 2\hat{L}_{\max} (1 + |S| (L_S + 2\lambda_1 |S|)^2 / \hat{L}_{\min}^2) \).

The proof of Theorem 1 is in the supplement. We present here a high-level sketch of the proof. We establish part 1 by proving a sufficient decrease condition. For part 2, we show that the objective function restricted to the group support \( S \) is strongly convex, and thus convergence follows from standard results on cyclic BCD, e.g., [8]. To establish the linear rate of convergence in part 3 of the theorem, we extend the result of [5] who show that cyclic BCD can achieve a linear rate of convergence on smooth and strongly convex functions: note that our objective function after support stabilization is not smooth due to the presence of the term \( \sum_{g \in S} \| \theta_g \|_2 \).

**Optimality conditions of BCD and PGD.** The conditions in Theorem 1 (part 2) characterize a fixed point of Algorithm 1. These are necessary optimality conditions for Problem (9) since any global minimizer must be a fixed point for Algorithm 1. In what follows, we will show that the necessary optimality conditions imposed by PGD (which is a generalization of [13] to the group setting) are generally less restrictive compared to those imposed by Algorithm 1. Note that PGD is an iterative algorithm whose updates for Problem (9) are given by:

\[
\theta^{l+1} \in \arg\min_{\theta} \left\{ \frac{1}{2\tau} \| \theta - (\theta^l - \tau \nabla \ell(\theta^l)) \|_2^2 + \Omega(\theta) \right\},
\]
where \( \tau > 0 \) is a step size. Let \( L \) be the Lipschitz constant of \( \nabla \ell(\theta) \). For a constant step size, the update in (19) converges if \( \tau = 1/L \) where \( L \) is a constant chosen such that \( \hat{L} > L \) [see, for example, 29, 42]. For the choice \( \tau = 1/\hat{L} \), it can be readily checked that any fixed point of PGD satisfies the three optimality conditions in Theorem 1 (part 2), but with \( \hat{L}_g \) replaced by \( \hat{L} \). The group-wise Lipschitz constant \( \hat{L}_g \) satisfies \( \hat{L}_g \leq L \) (for any \( g \)). In many high-dimensional problems, we can have \( \hat{L}_g \ll L \) [see 4, 29]. Thus, Algorithm 1 generally imposes more restrictive necessary optimality conditions compared to PGD, which can lead to higher quality local minima in practice. This establishes a part of the hierarchy in (10).
3.2. Local Combinatorial Search. In this section, we introduce a local combinatorial search algorithm to improve the quality of solutions obtained by cyclic BCD (Algorithm 1). The algorithm performs the following two steps in the \( t \)-th iteration:

1. **Block Coordinate Descent:** We run Algorithm 1 initialized at the current solution \( \theta^{t-1} \) to obtain a solution \( \theta^t \). We denote the indices of the nonzero groups in \( \theta^t \) by \( \text{Supp}(\theta^t) = S \).

2. **Group Combinatorial Search:** We attempt to improve the solution \( \theta^t \) by swapping groups of variables from inside and outside the support \( S \). In particular, we search for two subsets \( S_1 \subseteq S \) and \( S_2 \subseteq S^c \) such that removing \( S_1 \) from the support, adding \( S_2 \) to the support, and then optimizing over the groups in \( S_2 \), improves the current objective. To ensure that the local search problem is computationally feasible, we restrict our search to subsets satisfying \(|S_1| \leq m \) and \(|S_2| \leq m \), where \( m \) is a pre-specified integer that takes relatively small values (for example, in the range 1 to 10).

We present a formal description of the optimization problem in step 2 (above). We denote the standard basis of \( \mathbb{R}^p \) by \( \{e_1, \ldots, e_p\} \). Given a set \( J \subseteq [q] \), we define the \( p \times p \) matrix \( U^J \) as follows: the \( i \)-th column of \( U^J \) is \( e_i \) if \( i \in \cup_{g \in J} G_g \) and 0 otherwise. In other words, for any \( \theta \in \mathbb{R}^p \), we have \((U^J)_{ij} = \theta_i \) if \( i \in \cup_{g \in J} G_g \) and 0 otherwise. The optimization problem in Step 2 is given by:

\[
(20) \quad \min_{S_1, S_2, \theta} h(\theta^t - U^{S_1} \theta^t + U^{S_2} \theta) \quad \text{s.t.} \quad S_1 \subseteq S, S_2 \subseteq S^c, |S_1| \leq m, |S_2| \leq m,
\]

where we recall that \( S = \text{Supp}(\theta^t) \). If there is a feasible solution \( \hat{\theta} \) to (20) satisfying \( h(\hat{\theta}) < h(\theta^t) \), then we move to the improved solution \( \hat{\theta} \); otherwise, we terminate the algorithm. We summarize the algorithm below:

**Algorithm 2: Local Combinatorial Search**

- **Input:** Initial solution \( \theta^0 \) and swap subset size \( m \).
- **Repeat** Steps 1–3 for \( t = 1, 2, \ldots \) until convergence:
  1. Run Algorithm 1 initialized from \( \theta^{t-1} \) to obtain a solution \( \theta^t \).
  2. Search for a feasible solution \( \hat{\theta} \) to (20) satisfying \( h(\hat{\theta}) < h(\theta^t) \).
  3. If step 2 succeeds, \( \theta^t \leftarrow \hat{\theta} \). Otherwise, terminate.

Theorem 2 establishes that Algorithm 2 converges in a finite number of iterations and characterizes the corresponding solution.

**Theorem 2.** Let \( \{\theta^t\} \) be the sequence of iterates generated by Algorithm 2 and suppose Assumption 1 holds. Then, \( \theta^t \) converges in a finite number of iterations to a solution that we denote by \( \theta^\dagger \). Let \( S = \text{Supp}(\theta^\dagger) \). Then, \( \theta^\dagger \) satisfies the necessary optimality conditions in part 2 of Theorem 1. In addition, \( \theta^\dagger \) satisfies:

\[
(21) \quad h(\theta^t) \leq \min_{S_1, S_2, \theta} h(\theta^t - U^{S_1} \theta^t + U^{S_2} \theta) \quad \text{s.t.} \quad S_1 \subseteq S, S_2 \subseteq S^c, |S_1| \leq m, |S_2| \leq m.
\]

Theorem 2 shows that the solutions obtained by Algorithm 2 impose more restrictive necessary optimality conditions (in particular, condition (21)) compared to Algorithm 1, which justifies part of the hierarchy in (10). This is expected, as every iteration of Algorithm 2 improves over a solution obtained by Algorithm 1. The quality of solutions returned by Algorithm 2 depends on the swap subset size \( m \). For a sufficiently large choice of \( m \), the algorithm will return a global minimizer. Intuitively, the computational cost of the local search in step 2 of Algorithm 2 increases with \( m \). In our experiments, we observe that small choices such as \( m = 1 \) can lead to significant improvements in solution quality compared to algorithms that do not incorporate combinatorial optimization. These improvements are most pronounced in settings where \( n \ll p \) or the predictors across groups are highly correlated. In Section 4.1.2,
we present a MIP formulation for the local search problem in Algorithm 2 for \( m > 1 \). For the special case of \( m = 1 \), we use our own custom implementation that is more efficient than using a MIP-based approach.

3.3. Algorithms for the cardinality constrained formulation. Algorithms 1 and 2 provide solutions for the (penalized) formulation in (9). While this leads to a family of high-quality estimators across a range of model sizes, it does not allow for explicit control over the number of nonzero groups \( G(\theta) \). To this end, we consider the cardinality constrained variant of problem (9):

\[
\min_{\theta} E(\theta) := \ell(\theta) + \lambda \sum_{g \in [q]} \|\theta_g\|_2 \quad \text{s.t.} \quad G(\theta) \leq k.
\] (22)

In order to obtain a solution to (22) with a desired support size, we propose the following procedure. First, we run Algorithm 2 (say) over a grid of \( \lambda_0 \)-values to obtain a sequence of solutions. Then, if a desired support size, say \( k \), is missing, we obtain it by applying proximal gradient descent (PGD) to Problem (22):

\[
\theta_{l+1} \in \arg \min_{\theta: \ G(\theta) \leq k} \left\{ \frac{1}{2\tau} \|\theta - (\theta_l - \tau \nabla \ell(\theta_l))\|_2^2 + \lambda \sum_{g \in [q]} \|\theta_g\|_2 \right\},
\] (23)

where \( \tau > 0 \) is a step size and the initial solution \( \theta^0 \) can be obtained from Algorithm 2 (for example, we take a solution with group support size closest to \( k \)).

The next proposition establishes the convergence of update (23) and describes its fixed points.

**Proposition 1.** Let \( \{\theta^l\} \) be the sequence of iterates generated the PGD updates (23). Let \( L \) be the Lipschitz constant of \( \nabla \ell(\theta) \) and a scalar \( \hat{L} \) such that \( \hat{L} > L \). Then, \( \{\theta^l\} \) converges for a step size \( \tau = 1/\hat{L} \). Moreover, a solution \( \theta^* \) with group support \( S \) is a fixed point of (23) iff \( G(\theta^*) \leq k \), and

\[
\theta^*_S \in \arg \min_{\theta_S} E(\theta_S) \quad \text{and} \quad \|\nabla_{\theta_g} \ell(\theta^*)\|_2 \leq \gamma(k) \quad \text{for} \ g \in S^c,
\]

where \( \gamma_g = \|\hat{L}\theta^*_g - \nabla_{\theta_g} \ell(\theta^*)\|_2 \), and \( \gamma(k) \) denotes the \( k \)th largest value in the sequence \( \{\gamma_g\}_{g=1}^q \).

We omit the proof of Proposition 1 as it can be established by a simple extension to the standard results on the convergence of IHT [for example, those in 13, 4].

4. Mixed Integer Programming. In this section, we propose MIP formulations and algorithms to solve (9) and the combinatorial search problem in Algorithm 2. Section 4.1 introduces MIP formulations, and Section 4.2 presents a new BnB algorithm for solving the corresponding problems to optimality.

4.1. MIP Formulations.

4.1.1. Formulations for Problem (9). Below we present two MIP-formulations for (9).
**Big-M Formulation:** We first present a Big-M based MIP formulation for Problem (9):

\[(24a) \quad \min_{\theta, z} \quad \ell(\theta) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\theta_g\|_2\]
\[(24b) \quad \text{s.t. } \|\theta_g\|_2 \leq M_U z_g, \ g \in [q]\]
\[(24c) \quad z_g \in \{0, 1\}, \ g \in [q]\]

where, the optimization variables are $\theta$ (continuous) and $z$ (binary). Above, $M_U$ is an a-priori specified constant (leading to the name “Big-M”) such that some optimal solution, say $\theta^*$, to (9) satisfies $\max_{g \in [q]} \|\theta_g^*\|_2 \leq M_U$. In (24), the binary variable $z_g$ controls whether all the regression coefficients in group $g$ are zero or not: $z_g = 0$ implies that $\theta_g = 0$, and $z_g = 1$ implies that $\|\theta_g\|_2 \leq M_U$. Such Big-M formulations are commonly used in mixed integer programming to model relations between discrete and continuous variables, and have been recently used in $\ell_0$-regularized regression [10, 69] (for example). Various techniques have been proposed to estimate the constant $M_U$ in practice; see [10] for a discussion on estimating the Big-M in the context of linear regression. The constraints in (24b) are second order cones [14]. Moreover, the objective function in (24) can be written as a linear function, with additional second order cone constraints to express the quadratic function $\ell(\theta)$ and the terms $\|\theta_g\|_2, \ g \in [q]$. Thus, Problem (24) can be reformulated as a Mixed Integer Second Order Cone Program (MISOCP), which can be modeled and solved (for small/moderate problem instances) with commercial MIP solvers such as Gurobi, CPLEX, and MOSEK. We present an efficient, standalone BnB algorithm for (24) in Section 4.2.

**Perspective reformulation:** Recall that Problem (9) contains a ridge term in its objective. The ridge term can be used to derive stronger MIP formulations for (9) based on the perspective formulation [24, 26]. As we discuss below, the perspective-based formulation differs from the Big-M formulation (24)—when $\lambda_2 > 0$, it usually leads to tighter convex relaxations and consequently, reduced MIP runtimes. First, we rewrite (9) as

\[(25) \quad \min_{\theta, z} \quad \tilde{\ell}(\theta) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\theta_g\|_2 + \lambda_2 \sum_{g=1}^{q} \|\theta_g\|_2^2 \quad \text{s.t. } (24b), (24c)\]

where $\ell(\theta) = \tilde{\ell}(\theta) + \lambda_2 \|\theta\|_2^2$. Using the perspective reformulation [24, 26, 23] for the ridge term $\sum_{g \in [q]} \|\theta_g\|_2^2$ in the objective, we can reformulate (25) as

\[(26a) \quad \min_{\theta, z, s} \quad \tilde{\ell}(\theta) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\theta_g\|_2 + \lambda_2 \sum_{g=1}^{q} s_g, \]
\[(26b) \quad \text{s.t. } \|\theta_g\|_2 \leq M_U z_g, \ g \in [q]\]
\[(26c) \quad \|\theta_g\|_2^2 \leq s_g z_g, \ g \in [q]\]
\[(26d) \quad z_g \in \{0, 1\}, \ s_g \geq 0, \ g \in [q].\]

Compared to (25), formulation (26) uses additional auxiliary variables $s_g \in \mathbb{R}_{\geq 0}, \ g \in [q]$ and rotated second order cone constraints: $\|\theta_g\|_2^2 \leq s_g z_g$ for $g \in [q]$. Each $s_g$ takes the place of the term $\|\theta_g\|_2^2$ in the objective function in (24). Specifically, any optimal solution $(\theta^*, z^*, s^*)$ to (26) must satisfy $s_g^* = \|\theta_g^*\|_2^2$.

Although the MIP formulations (26) and (25) are equivalent, their continuous relaxations are generally different. The following proposition states that the relaxation of (26) is generally tighter (i.e., has a higher objective) than the relaxation of (25).
PROPOSITION 2. Let $v_1$ and $v_2$ be the objective values of (25) and (26) upon relaxing the binary variable $z_g$ to $[0, 1]$ for all $g \in [q]$. Let $(\theta^*, z^*, s^*)$ be an optimal solution to the relaxation corresponding to $v_2$. Then, the following holds:

$$v_2 - v_1 \geq \lambda_2 \sum_{g \in [q]} \|\theta^*_g\|_2^2 \left( (z^*_g)^{-1} - 1 \right).$$

Proposition 2 implies that using formulation (26) (over formulation (24)) can lead to tighter lower bounds for the root node relaxation; and hence tighter dual bounds for the node relaxations in the BnB tree. This can result in improved runtimes in the overall BnB solver (as we demonstrate in our experiments). Thus, in our algorithmic framework in Section 4.2, we focus on formulation (26). To be clear, our BnB procedure applies even without the presence of a ridge term (i.e., $\lambda_2 = 0$). Specifically, if $\lambda_2 = 0$ in (26), the conic constraints (26c) can be removed and formulation (26) reduces to the Big-M formulation in (24).

4.1.2. MIP formulation for local combinatorial search. We present a MIP formulation for the local search problem\(^5\) that arises in Algorithm 2. Problem (20) can be formulated using the following Big-M based MIP:

$$\begin{align*}
\min_{u, z, \theta} & \quad \ell(u) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|u_g\|_2 \\
\text{s.t.} & \quad u = \theta^t - \sum_{g \in S} U^g \theta^t (1 - z_g) + \sum_{g \in S^c} U^g \theta \\
& \quad \|u_g\|_2 \leq M_{u} z_g, \quad g \in S^c \\
& \quad \sum_{g \in S} z_g \geq |S| - m, \quad \sum_{g \in S^c} z_g \leq m \\
& \quad z_g \in \{0, 1\}, \quad g \in [q].
\end{align*}$$

In the formulation above, we assume that $M_{u}$ is chosen sufficiently large so that some optimal solution to (20), say $\theta^*$, satisfies $\|\theta^*_g\|_2 \leq M_{u}$, $g \in S^c$. As we discuss below, the objective in (27) represents $h(u)$ with $u = \theta^t - U^{S_1} \theta^t + U^{S_2} \theta$, where $h(u)$, $S_1$ and $S_2$ are as defined in (20). Note that the variable $u$ is an auxiliary variable introduced to simplify the presentation. The binary variables $z_g, g \in [q]$ are used to select the subsets $S_1 \subseteq S$ and $S_2 \subseteq S^c$. In particular, for $g \in S$, $z_g = 0$ iff $g \in S_1$, and this is encoded by constraint (27a). On the other hand, for $g \in S^c$, $z_g = 0$ iff $g \in S_2$, and this is encoded by constraints (27a) and (27b). Therefore, $\sum_{g=1}^{q} z_g$ is equal to $G(u)$. The constraints (27c) enforce $|S_1| \leq m$ and $|S_2| \leq m$.

The local search MIP-formulation (27) has a smaller search space compared to the full problem (24). This is due to the additional constraints appearing in (27c). Furthermore, Problem (27) effectively uses $|S^c|$-many ‘free’ continuous group-variables—this is in contrast to $|S| + |S^c|$ continuous group-variables appearing in the full problem. Thus, for small values of $m$, Problem (27) can be typically solved faster than the MIP formulation of (8). While (27) is based on a Big-M formulation, in the presence of an additional ridge regularizer, one can also derive a perspective reformulation using ideas similar to (26).

\(^5\)We recommend the use of the MIP formulations when $m \geq 2$. When $m = 1$ a solution to the local search procedure can be computed efficiently from first principles.
4.2. Exact optimization via a custom nonlinear Branch-and-Bound algorithm. High-performance commercial MIP solvers, such as Gurobi and CPLEX, often deliver state-of-the-art performance for a variety of MIP problems. These solvers are based on a BnB framework, which can solve MIP problems to global optimality, typically without having to explicitly enumerate all (exponentially many) solutions in the search space. These solvers are general-purpose and do not take into account the specific structure of the problems we consider here. Therefore, their performance can suffer: we have empirically observed that they may require several hours to solve (to certifiable optimality) instances of (26) with \( p \sim 10^3 \), and larger problems can take much longer.

To address this lack of scalability in general-purpose MIP solvers, we propose a specialized, nonlinear BnB framework for solving (26) to certifiable optimality. Our framework takes into account problem structure to achieve scalability. As we demonstrate in the experiments section, our BnB can solve instances with \( p \sim 5 \times 10^6 \) to certifiable optimality in minutes to hours, whereas Gurobi takes prohibitively long (at least a day) for \( p \sim 10^3 \). An important feature of our proposal is an open-source, standalone implementation of the BnB solver, which does not rely on sophisticated and proprietary BnB-capabilities of commercial MIP solvers (e.g., Gurobi). We first give a high-level overview of our novel nonlinear BnB framework and then dive into specific technical details.

Overview of nonlinear BnB: Nonlinear BnB is a general framework for solving mixed integer nonlinear programs [7]. This framework constructs a search tree to partition the set of feasible solutions of the given MIP (Problem (26) in our case). Instead of explicitly enumerating all the (exponentially many) feasible solutions, BnB uses intelligent enumeration and methods to prune parts of the tree by using lower bounds (dual bounds) on the optimal objective value. In what follows, we briefly describe how the tree is constructed and pruned. Starting at the root node, the algorithm solves a nonlinear convex relaxation of Problem (26), where all binary variables are relaxed to \([0, 1]\) – this is usually referred to as the root relaxation. Then, the algorithm chooses a branching variable, say \( z_g \), and creates two child nodes (optimization subproblems): one with \( z_g = 0 \) and another with \( z_g = 1 \), where all other binary variables are relaxed to \([0, 1]\). The algorithm then proceeds recursively: for every unvisited node, it solves the corresponding optimization problem and checks if there is any fractional (i.e., non-binary) variable \( z_g \). If there is any fractional \( z_g \), the branching process must continue — to this end, the algorithm branches on one fractional \( z_g \), generating two new child nodes. Thus, every node in the search tree corresponds to an optimization subproblem and every edge represents a branching decision.

While growing the search tree, BnB maintains an upper bound on the objective function (which can be obtained from any feasible solution to the problem). If the optimization subproblem at the current node leads to an objective value that exceeds the upper bound, then the node is pruned (i.e., no children are generated for this node), because none of its descendants can have a better objective value than the upper bound. Another case where BnB can safely prune a node is when the corresponding subproblem leads to an integral solution, i.e., a binary \( z \) (since there will be no variables to branch on). For further discussion on nonlinear BnB, see [7].

Specific details: There are many delicate details in BnB that can critically affect its scalability: for example, the choice of the algorithm for solving the continuous node subproblems, obtaining upper bounds, branching, and tree-search strategies. We discuss our choices below:

- Subproblem solver: The optimal solutions of the continuous optimization subproblems encountered in the course of BnB are typically sparse (see Section 4.2.1 for further discussions). To solve these subproblems, we propose an active-set algorithm, which exploits sparsity by considering a reduced problem restricted to a small subset of groups. Moreover,
we share information on the active sets across the BnB tree to speed up convergence (see Section 4.2.2).

• **Upper bounds:** Better upper bounds can lead to aggressive pruning in the search tree, which can reduce the overall runtime. We obtain the initial upper bound using the approximate algorithms of Section 3. As we demonstrate in the experiments, our approximate algorithms typically obtain optimal or near-optimal solutions, making them a good choice to initialize BnB. Moreover, at every node of BnB, we attempt to improve the upper bound by using the sparsity pattern of the solution to the current node’s subproblem. More concretely, let $S \subseteq q$ denote the group support of the latter subproblem’s solution. Then, we obtain a new upper bound, by restricting optimization to $S$, i.e., we solve:

$$
\min_\theta \ \hat{\ell}(\theta) + \lambda_1 \sum_{g=1}^{q} \|\theta_g\|_2 + \lambda_2 \|\theta\|_2^2 \quad \text{s.t.} \quad \theta_{S^c} = 0, \ \|\theta_g\|_2 \leq M_U, \ g \in [q].
$$

• **Branching and search strategies:** The branching strategy selects the next variable to branch on, while the search strategy decides which unexplored node in the search tree to visit next. Many elaborate strategies for branching and search have been proposed in the literature – see [50] for a survey. When the initial upper bound is of high quality, more aggressive pruning is possible, and simple strategies tend to work relatively well in practice [for example, see the discussion in 19]. Since our approximate algorithms typically return good upper bounds, we rely on simple strategies. For branching, we use maximum fractional branching [7, 50], which branches on the factional variable $z_g$ whose value is closest to 0.5. For search, we use breadth-first search and switch to depth-first search if memory issues are encountered.

Our approach extends our recent work [31] for the best subset selection problem (with a group size of one). We note that there are important differences as the Group $\ell_0$ problem involves a different and more challenging optimization formulation. Specifically, the Big-M constraints in (26b) translate to second order cones, instead of box-constraints that appear when the group sizes are one. Furthermore, in the group setup, we have a non-smooth term $\sum_{g \in [q]} \|\theta_g\|_2$ in the objective of (26). The conic constraints and $\ell_2$ norms in our problem require special care when developing the subproblem solver (for example, when reformulating the subproblems in Section 4.2.1 and designing the active set algorithm in Section 4.2.2). It is also worth mentioning that in the simplest case where $\lambda_1 = \lambda_2 = 0$, our solver solves a MISOCP, whereas [31] solves a mixed integer quadratic program.

4.2.1. Relaxation reformulation. In this section, we study the convex relaxation arising at a node of the BnB search tree. We present a particular reformulation of this problem that leads to (i) useful insights about the sparsity in the solutions of the convex relaxation; and (ii) computational benefits. To simplify the presentation, we will first focus on the root relaxation of (26), which is obtained by relaxing all the binary variables in (26) to $[0, 1]$.

Note that the root relaxation involves the variables $(\beta, z, s)$. In Proposition 3, we show that the root relaxation can be reformulated in the $\beta$ space, leading to a regularized least squares problem. The associated regularizer can be characterized in terms of the reverse Huber penalty [54] (see also [23]), which is a function $\mathcal{H} : \mathbb{R} \rightarrow \mathbb{R}$ defined as follows:

$$
\mathcal{H}(t) = \begin{cases} 
|t| & \text{if } |t| \leq 1 \\
(t^2 + 1)/2 & \text{otherwise}.
\end{cases}
$$

(28)
Proposition 3. The root relaxation obtained by relaxing the binary variables in (26) to \([0, 1]\) is equivalent to:

\[
\min_{\theta} \ F(\theta) := \tilde{\ell}(\theta) + \sum_{g=1}^{q} \Psi(\theta_g; \lambda, \mathcal{M}_u) \quad \text{s.t.} \quad \|\theta_g\|_2 \leq \mathcal{M}_u, \ g \in [q].
\]

where \(\lambda = (\lambda_0, \lambda_1, \lambda_2)\) and

\[
\Psi(\theta_g; \lambda, \mathcal{M}_u) := \begin{cases} 
2\lambda_0 \mathcal{H}(\sqrt{\lambda_2/\lambda_0}\|\theta_g\|_2) + \lambda_1 \|\theta_g\|_2 & \text{if } \sqrt{\lambda_0/\lambda_2} \leq \mathcal{M}_u \\
(\lambda_0/\mathcal{M}_u + \lambda_1 + \lambda_2 \mathcal{M}_u)\|\theta_g\|_2 & \text{if } \sqrt{\lambda_0/\lambda_2} > \mathcal{M}_u.
\end{cases}
\]

The reformulation in (29) eliminates the the conic and Big-M constraints from the root relaxation, at the expense of introducing the non-smooth penalty \(\sum_{g=1}^{q} \Psi(\theta_g; \lambda, \mathcal{M}_u)\) which is separable across the blocks \(\{\theta_g\}_q\). Depending on the choices of \(\lambda\) and \(\mathcal{M}_u\), the penalty \(\Psi\) is either the \(\ell_2\) norm or a combination of the reverse Huber penalty and the \(\ell_2\) norm. In either case, the penalty is sparsity-inducing. In essence, Problem (29) is similar to the Group Lasso problem [71], with two exceptions: (i) Problem (29) has the additional constraints: \(\|\theta_g\|_2 \leq \mathcal{M}_u, \ g \in [q]\), and (ii) when \(\sqrt{\lambda_0/\lambda_2} \leq \mathcal{M}_u\), the penalty involves the reverse Huber penalty.

Node relaxations within the BnB tree: The convex relaxation subproblem encountered at a node of the BnB search tree is similar to the root relaxation, except that some of the \(z_g\)s are fixed to 0 or 1. The fixed \(z_g\)s are determined by the branching decisions made starting from the root until reaching the node. The convex relaxation at a particular node can be reformulated in the \(\beta\)-space similar to the reformulation of the root relaxation in (29), except that: (i) if \(z_g = 0\) then the corresponding group should be removed from the objective function; and (ii) if \(z_g = 1\), then the penalty \(\Psi(\theta_g; \lambda, \mathcal{M}_u)\) should be replaced with \(\tilde{\Psi}(\theta_g; \lambda) := \lambda_1 \|\theta_g\|_2 + \lambda_2 \|\theta_g\|_2^2\). More precisely, let \(\mathcal{Z}\) and \(\mathcal{N}\) be the sets of indices of the \(z_g\)s that are fixed to 0 and 1, respectively. Then, the following subproblem is solved at the corresponding node:

\[
(30) \ \min_{\theta} \ \tilde{\ell}(\theta) + \sum_{g \in \mathcal{N}^c} \Psi(\theta_g; \lambda, \mathcal{M}_u) + \sum_{g \in \mathcal{N}} \tilde{\Psi}(\theta_g; \lambda) \quad \text{s.t.} \quad \theta_\mathcal{Z} = 0, \|\theta_g\|_2 \leq \mathcal{M}_u, \ g \in [q].
\]

In the next section, we develop a scalable algorithm for solving Problem (29). The BnB subproblem (30) can be solved similarly after accounting for the fixed \(z_g\)s.

4.2.2. Active-Set subproblem solver. As discussed earlier, a solution to Problem (29) is expected to be sparse in \(\theta\) (this will be also true for the node sub-problems in the BnB tree). To exploit this sparsity, we use an active-set algorithm: We start by solving Problem (29) restricted to a small subset of groups (i.e., the active set). After convergence on the active set, we augment the active set with a collection of groups that violate the optimality conditions for the full problem (if any) and then resolve the problem restricted to the augmented active set. The algorithm keeps iterating between solving a reduced optimization problem and augmenting the active set, until the optimality conditions for the full problem are satisfied. Such active-set algorithms have proven to be effective in scaling up the solvers for group Lasso-type problems [for example, see 30]—our usage differs in that we use this active-set strategy within every node of the BnB tree.

Next, we describe our active-set algorithm more formally. Let \(A \subseteq [q]\) be the active set. The algorithm starts by solving (29) restricted to the active set, i.e.,

\[
(31) \quad \hat{\theta} \in \arg\min_{\theta} \ F(\theta) \quad \text{s.t.} \quad \|\theta_g\|_2 \leq \mathcal{M}_u, \ g \in [q], \ \theta_{\mathcal{A}^c} = 0.
\]
After solving (31), we check if \( \hat{\theta} \) satisfies the optimality condition for the full problem. Equivalently, for every group \( g \in A^c \), we check if the following holds

\[
0 \in \arg \min_{\theta_g} F(\hat{\theta}_1, \ldots, \hat{\theta}_g, \ldots, \hat{\theta}_q) \quad \text{s.t.} \quad \|\theta_g\|_2 \leq M_U. \tag{32}
\]

Since \( \theta_g = 0 \) is in the interior of the feasible set, condition (32) is equivalent to the zero-subgradient condition: \( 0 \in \partial_{\theta_g} F(\hat{\theta}_1, \ldots, \hat{\theta}_{g-1}, 0, \hat{\theta}_{g+1}, \ldots, \hat{\theta}_q) \), and can be checked in closed form.

We repeat the procedure of solving the restricted subproblem in (31) and augmenting \( A \) with groups that violate (32), until there are no more violations. The algorithm is summarized below.

**Algorithm 3: An Active-set Algorithm for (29)**

- **Input:** Initial solution \( \hat{\theta} \) and initial active set \( A \).
- **Repeat** Steps 1–3 till convergence:
  1. Solve the restricted problem (31) to get a solution \( \hat{\theta} \).
  2. \( V \leftarrow \{ g \in A^c \mid (32) \text{ is violated} \} \).
  3. If \( V \) is empty terminate, otherwise, \( A \leftarrow A \cup V \).

Algorithm 3 is guaranteed to converge to an optimal solution for Problem (29) in a finite number of steps, as there are finitely many groups.

**Choice of the active set:** The quality of the initial active set \( A \) can have an important effect on the number of iterations in Algorithm 3. Due to the choice of our branching rule, the parent and its two child nodes solve similar subproblems; the only difference between these subproblems is that a single \( z_g \) is fixed to 0 or 1 in the children. Thus, the solutions and supports of the parent and its children are unlikely to differ by much. We therefore initialize the active set of every node in the BnB tree (except the root) with the support of its parent. For the root node, we initialize the active set with the support of the warm start, obtained from the approximate algorithms that are discussed in Section 3.

**Solving the restricted subproblem:** The convex sub-problem (31) in Step 1 has a small active set and can be solved with a variety of optimization algorithms: for example, BCD, proximal gradient methods [8] or an interior point solver (as available in Gurobi). In our experiments, we use the latter due to its good performance in practice.

5. **Statistical Theory.** In this section we derive non-asymptotic prediction and estimation error bounds for the Group \( \ell_0 \) estimators, and compare them to the bounds that have been established for the corresponding Group Lasso-based approaches. We focus on linear regression models in Section 5.1 and on nonparametric additive models in Section 5.2.

In our analysis, we focus on constrained specifications of the proposed estimators, leaving the penalized case for future research. To simplify the presentation, we consider the setting where the model is correctly specified, so that the true regression function is a feasible solution to the corresponding optimization problem. However, our results can be generalized to allow for model misspecification.

We say that a constant is universal if it does not depend on other parameters, such as \( n, q \) or \( k \). We use the notation \( \gtrsim \) and \( \lesssim \) to indicate that inequalities \( \geq \) and \( \leq \), respectively, hold up to positive universal multiplicative factors, and write \( \asymp \) when the two inequalities hold simultaneously. We use \( a \lor b \) to denote \( \max(a, b) \).

---

6In some cases, \(|V|\) can be large, which can slow down the solver in Step 1. Thus, if \( V \) has more than \( K \) groups, we augment \( A \) with the \( K \) groups in \( V \) that have the largest violation (instead of \( A \leftarrow A \cup V \)). In our experiments we set \( K = 10 \). We found this helpful to keep the size of the active set manageable during the course of the algorithm.
5.1. **Linear Model.** We assume that the observed data follows the model $y = X\beta^* + \epsilon$, where $X$ is deterministic and the elements of $\epsilon$ are independent $N(0, \sigma^2)$ with $\sigma > 0$. We define $k_* = G(\beta^*)$ and refer to $n^{-1}\|X\hat{\beta} - X\beta^*\|^2$ as the prediction error for estimator $\hat{\beta}$. Given $\beta \in \mathbb{R}^p$ and $J \subseteq [q]$, we write $\beta_J$ for the sub-vector of $\beta$ indexed by $\cup_{g \in J} G_g$. Consider the following definition, in which we use the notation $\|\beta\|_{2,1} = \sum_{g=1}^q \|\beta_g\|_2$.

**DEFINITION 1.** Given a positive integer $k$ and a constant $c \geq 1$, let

$$
\gamma_k = \min_{\beta \neq 0, G(\beta) \leq k} \frac{\sqrt{k}\|X\beta\|_2}{\sqrt{n}\|\beta\|_{2,1}} \quad \text{and} \quad \kappa_{k,c} = \min_{J \subseteq [q], |J| \leq k} \left\{ \min_{\beta \neq 0, \|\beta_J\|_{2,1} \leq c} \frac{\sqrt{k}\|X\beta\|_2}{\sqrt{n}\|\beta_J\|_{2,1}} \right\}.
$$

The above definition is most meaningful under the scaling of the features where $\|x_j\|_2 \asymp \sqrt{n}$ for all $j$. As we discuss below, constants $\kappa_{k,c}$, with $c > 1$, appear in the prediction and estimation error bounds for the Group Lasso estimator, while $\gamma_{2k}$ appears in the estimation error bound for the Group $\ell_0$ estimator. The following result establishes a useful relationship for these quantities.

**PROPOSITION 4.** $\gamma_{2k} \geq \kappa_{k,c}/\sqrt{2}$, for all positive integers $k$ and all $c \geq 1$.

We study estimator $\hat{\beta}$ that solves the following optimization problem:

$$
\min_{\beta} \|y - X\beta\|^2_2 \quad \text{s.t.} \quad \sum_{g=1}^q \mathbf{1}(\beta_g \neq 0) \leq k,
$$

where $k$ is a fixed parameter that controls the sparsity level. We note that (33) is a special case of the cardinality constrained problem considered in Section 3.3. We write $T_g$ for the number of features in group $g \in [q]$ and define $\bar{T} = \max_{|J| \leq k} \sum_{g \in J} T_g / k$, noting the following relationships in the special case where every group has the same number of $T$ features: $\bar{T} = T$ and $p = qT$. Our first result provides the prediction error bound for $\hat{\beta}$, which holds without any assumptions on the design.

**THEOREM 3.** Let $\delta_0 \in (0, 1)$ and suppose that $\hat{\beta}$ is a global solution to optimization problem (33) for $k \geq k_*$. Then,

$$
\frac{1}{n}\|X\hat{\beta} - X\beta^*\|_2^2 \lesssim \sigma^2 k \left[ \frac{\bar{T} + \log(q/k)}{n} \right] + \sigma^2 \left[ \frac{\log(1/\delta_0)}{n} \right]
$$

with probability at least $1 - \delta_0$.

Letting $\delta_0 = (k/q)^k$ and using Definition 1, we derive the following result.

**COROLLARY 1.** If $\hat{\beta}$ is a global solution to optimization problem (33) for $k = k_*$, then

$$
\frac{1}{n}\|X\hat{\beta} - X\beta^*\|_2^2 \lesssim \sigma^2 k_* \left[ \frac{\bar{T}_* + \log(q/k_*)}{n} \right]
$$

$$
\|\hat{\beta} - \beta^*\|_{2,1} \lesssim \sigma k_* \left[ \frac{\bar{T}_* + \log(q/k_*)}{n} \right]^{1/2} [\gamma_{2k_*}]^{-1}
$$

with probability at least $1 - (k_*/q)^{k_*}$.
We make several observations regarding the established error bounds, comparing them to the bounds for the Group Lasso estimator, denoted by $\hat{\beta}_{GL}$, which replaces the $\ell_0$ constraint in Problem (33) with a penalty on $\|\beta\|_{2,1}$. To simplify the comparison of the corresponding rates, we focus on the setting where $T_g = T$ for all $g \in \{q\}$ and $k = k_*$.

**Remark 1.** The Group $\ell_0$ prediction error rate provided in Corollary 1 matches the corresponding optimal prediction error rate established in [41]. The estimation error rate in Corollary 1 is also optimal provided that $\gamma_{2k_*}^{-1}$ is bounded by a universal constant under the aforementioned feature scaling $\|x_j\|_2 \propto \sqrt{n}$.

**Remark 2.** Let $\|x_j\|_2 \propto \sqrt{n}$ for all $j$ and assume that $\kappa_{k_*,c}^{-1}$ is bounded by a universal constant for some $c > 1$. Then, the error bounds for the Group Lasso estimator [see, for example, Section 8.3 of 16] are

$$n^{-1}\|X\hat{\beta}_{GL} - X\beta^*\|_2^2 \lesssim \sigma^2 k_* \left[\frac{T + \log(q)}{n}\right]$$

The Group $\ell_0$ rates discussed in Remark 1 are better than those in display (34), because they replace the $\log(q)$ term with $\log(q/k_*)$. Moreover, in view of Proposition 4, the assumption on $\gamma_{2k_*}$ in Remark 1 is weaker than the Group Lasso assumption on $\kappa_{k_*,c}$. Finally, the Group $\ell_0$ prediction error bound holds without any assumptions on the design.

The last observation represents an important non-trivial advantage of $\ell_0$-based approaches over Lasso-type methods. [75] provide examples of design matrices in the usual linear regression context for which the Lasso prediction error is lower-bounded by a constant multiple of $1/\sqrt{n}$, generally leading to a much larger prediction error than the one for the $\ell_0$-based method.7

**Remark 3.** One advantage of estimator (33) is that tuning parameter $k$ directly controls the sparsity of the proposed estimator. In particular, the $\hat{\beta}$ that achieves the bounds in Corollary 1 satisfies $G(\hat{\beta}) \leq k_*$. On the other hand, the $\hat{\beta}_{GL}$ that achieves bounds (34) is typically much more dense. The following inequality, which holds with high probability, is provided in [41]:

$$G(\hat{\beta}_{GL}) \leq \left[\frac{64\phi_{\max}}{\kappa_{k_*,3}}\right] k_*$$

Here, $\phi_{\max}$ is the maximum eigenvalue of $X^\top X/n$. Thus, the right-hand side is at least $64k_*$.  

The error rates presented above can also apply to approximate solutions obtained after an early termination of the MIO solver. Upon termination, the solver provides the upper and lower bounds on the value of the objective function. We denote these bounds by $UB$ and $LB$, respectively, and write $\tau = (UB - LB)/UB$ for the corresponding optimality gap. The next result considers an approximate solution $\beta$ and demonstrates that the bounds in Corollary 1 hold for $\beta$ when $\tau$ is bounded away from one and $\tau \lesssim k_*[T_{k_*} + \log(q/k_*)]/n$.

**Corollary 2.** Let $k = k_*$ and suppose that $\tau \leq 1 - c$ for some positive universal constant $c$. Then,

$$\frac{1}{n} \|X\beta - X\beta^*\|_2^2 \lesssim \sigma^2 k_* \left[\frac{T_{k_*} + \log(q/k_*)}{n}\right] + \sigma^2 \tau$$

Footnote 7: The lower-bound applies to a wide class of coordinate-separable M-estimators, including local optima of nonconvex regularizers such as SCAD and MCP.
with probability at least \(1 - (k_\ast/q)^{k_\ast}\).

An attractive feature of Theorem 3 is that the uncertainty parameter \(\delta_0\) is independent of the tuning parameter \(k\). This allows us to control the expected prediction error\(^8\), as we demonstrate in the following result.

**Corollary 3.** Under the conditions of Theorem 3,
\[
\mathbb{E}\|\mathbf{X}\hat{\beta} - \mathbf{X}\beta^\ast\|_2^2 \lesssim \sigma^2 k \left[\hat{T}_k + \log(q/k)\right].
\]

5.1.1. **A BIC-type penalized formulation.** The results presented above rely on the knowledge that \(k \geq k_\ast\). We now analyze a BIC-type approach for selecting \(k\) that does not require knowing the true sparsity level \(k^\ast\). Approaches of this type are often used in the setting of high-dimensional linear regression for the purposes of model selection [36, 67].

We denote the global solution to Problem (33) by \(\hat{\beta}_k\) and let \(\hat{T} = \max_{g \in [q]} T_g\). The next result focuses on the estimator \(\hat{\beta}_B = \hat{\beta}_{\hat{k}}\), where
\[
\hat{k} = \arg\min_{k \leq q} \|\mathbf{y} - \mathbf{X}\hat{\beta}_{\hat{k}}\|_2^2 + a\hat{k} [\hat{T} + \log(q/k)]
\]
and \(a\) is a non-negative tuning parameter.

**Theorem 4.** There exists a universal constant \(a_0\), such that if \(a_0\sigma^2 \leq a \lesssim \sigma^2\), then
\[
\mathbb{E}\|\mathbf{X}\hat{\beta}_B - \mathbf{X}\beta^\ast\|_2^2 \lesssim \sigma^2 (k_\ast \lor 1) \left[\hat{T} + \log(q/k^\ast)\right] \quad \text{and} \quad \mathbb{E}G(\hat{\beta}_B) \lesssim k_\ast \lor 1.
\]

Theorem 4 demonstrates that, for an appropriate choice of \(a\), estimator \(\hat{\beta}_B\) achieves the optimal prediction error rate in Remark 1 and has the same order of group sparsity as \(\beta^\ast\). The recommended range for the tuning parameter \(a\) depends on the standard deviation of the noise in the model, i.e., \(\sigma\). This approach may be used when the practitioner has knowledge of \(\sigma\), but \(k^\ast\) is unknown.

When the parameter \(\sigma\) is unknown, it needs to be estimated – one possible approach is to use noise variance estimation methods studied in the context of high-dimensional linear models [see, for example, 70, 6, and the references therein]. Alternatively, in practice, one can tune the parameter \(a\) based on a separate validation set or via cross-validation\(^9\).

5.1.2. **The case of large group sizes.** The results presented so far illustrate that, at least on average, the group sizes need to be of a smaller order than \(n\) to achieve prediction consistency. We now consider the challenging setting where some groups may have more than \(n\) features. In this setting, additional regularization within each group is required for good predictive performance. One way to impose additional regularization is to encourage sparsity within groups and, thus, perform bi-level variable selection [32]. This can be achieved by including an additional \(\ell_0\) penalty on the number of features within each group or an additional \(\ell_1\) penalty on the coefficients within each group. Here, we take the approach of regularizing using the \(\ell_2\)-penalty, without encouraging additional sparsity. As discussed in Section 2.2, we

\(^8\)An application of Definition 1 yields a corresponding bound on the expected estimation error.

\(^9\)We acknowledge, however, that when the number of observations is small, this approach may not be ideal.
take a similar approach in nonparametric additive modeling, where groups sizes of order $n$ arise naturally via the cubic spline representation of the functional components.

We assume, for concreteness, that the features have been normalized to achieve $\|x_j\|_2 = \sqrt{n}$ for all $j \in [p]$, and focus on the following optimization problem:

$$\min_{\beta} \quad \|y - X\beta\|_2^2 + \lambda \sum_{g \in [q]} \sqrt{T_g} \|\beta_g\|_2 \quad \text{s.t.} \quad \sum_{g=1}^q 1(\beta_g \neq 0) \leq k. \quad (35)$$

Because the $\ell_2$-penalty parameter is allowed to vary across groups, (35) is a slight generalization of the cardinality constrained problem considered in Section 3.3, but the algorithms presented here will apply to (35) with minor adjustments. We define $\tilde{q} = p/\min_{g \in [q]} T_g$ and note that $\tilde{q} = q$ when all the groups are of the same size. The next result establishes a prediction error bound for the large group-size setting.

**Theorem 5.** Let $\delta_0 \in (0, 1)$ and suppose that $\hat{\beta}$ is a global solution to problem (35) for $k \geq k_s$. There exist universal positive constants $c_0$, such that if $\lambda \geq c_0 \sigma \sqrt{n \log(2e\tilde{q})}$, then

$$\frac{1}{n} \|X\hat{\beta} - X\beta^*\|_2^2 \lesssim \frac{\lambda}{n} \sum_{g \in [q]} \sqrt{T_g} \|\beta_g^*\|_2$$

with probability at least $1 - 1/[4e\tilde{q}]$.

Focusing on the case $T_g = T$ for all $g \in [q]$ and $k = k_s$, for concreteness, we derive the corresponding prediction error rate\(^{10}\) of $\sigma \sqrt{T \log(q)/n} \|\beta^*\|_{2,1}$. This error rate demonstrates that prediction consistency can be achieved even when $T \gg n$, provided that the coefficients in $\beta^*$ are sufficiently small. For example, when each coefficient equals $1/T$, the resulting rate is $\sigma k_s \sqrt{\log(q)/n}$.

### 5.2. Nonparametric Additive Model

We study the performance of the proposed approach in the deterministic design setting. We write $\| \cdot \|_{L_2}$ for the $L_2$ norm of a real-valued function on $[0, 1]$. Using the notation in Section 2.2, we let $\mathcal{C}_j = \mathcal{C}$ for all $j$ and focus on the case where $\mathcal{C}$ is an $L_2$-Sobolev space:

$$\mathcal{C} = \left\{ g : [0, 1] \rightarrow \mathbb{R}, \|g\|_{L_2} + \|g^{(m)}\|_{L_2} < \infty \right\} \quad \text{and} \quad \text{Pen}(g) = \|g^{(m)}\|_{L_2}.$$

We define $\mathcal{C}_{gr} = \{ f : [0, 1]^q \rightarrow \mathbb{R}, f(x) = \sum_{j=1}^q f_j(x_j), f_j \in \mathcal{C} \}$ as the corresponding space of additive functions. We associate each $f \in \mathcal{C}_{gr}$ with the vector $f = \sum_{j=1}^q f_j$, where $f_j = (f_j(x_{1j}), \ldots, f_j(x_{nj}))$, and let

$$G(f) = \sum_{j=1}^q 1(f_j \neq 0), \quad \text{Pen}_{gr}(f) = \sum_{j=1}^q \text{Pen}(f_j).$$

We focus on the estimator that globally solves the following optimization problem:

$$\min_{f \in \mathcal{C}_{gr}} \quad \|y - f\|_n^2 + \lambda_n \text{Pen}_{gr}(f) \quad \text{s.t.} \quad G(f) \leq k, \quad (36)$$

where $\| \cdot \|_n$ denotes the Euclidean norm divided by $\sqrt{n}$.\(^{11}\) To ensure identifiability of the representation $f(x) = \sum_{j=1}^q f_j(x_j)$, additional restrictions are typically imposed. For example, a popular method is to separate out the constant term and require that $\sum_{i=1}^n f_j(x_{ij}) = 0$

---

\(^{10}\)When $T = 1$, this rate matches the slow error rate for the Lasso estimator [for example, 16].

\(^{11}\)We acknowledge the notational inconsistency when $n \leq 2$. 

for each $j$. Here we follow the approach of [61] and avoid specifying a particular set of restrictions. We treat every representation of $f$ as equivalent, with the understanding that one particular representation is used when evaluating properties of the components, such as $\|f_j\|_n$.

We are interested in comparing estimator (36), denoted by $\hat{f}$, with the widely popular Group Lasso-based approach, which replaces the $\ell_0$ constraint in Problem (36) with a penalty on $\sum_{j=1}^q \|f_j\|_n$. Theoretical properties of the latter approach have been investigated extensively [see, for example, 48, 37, 57, 60, 72, 61, and the references therein]. To compare the error bounds for the two estimators, we need the following definition.

**Definition 2.** Given a positive integer $k$, a constant $\xi \in (1, \infty]$ and an index set $J \subseteq [q]$, let

$$A_{k, \xi} = \{ f \in C_{gr} : \sum_{j=1}^q \|f_j\|_n \neq 0, G(f) \leq k, 2n^{-m/(2m+1)}\text{Pen}_{gr}(f) \leq (\xi - 1) \sum_{j=1}^q \|f_j\|_n \}$$

$$B_{k, \xi} = \{ f \in C_{gr} : \sum_{j=1}^q \|f_j\|_n \neq 0, \sum_{j \notin J} \|f_j\|_n + n^{-m/(2m+1)}\text{Pen}_{gr}(f) \leq \xi \sum_{j \in J} \|f_j\|_n \}$$

$$\psi(k, \xi) = \min_{f \in A_{k, \xi}} \frac{\sqrt{k}\|f\|_n}{\sum_{j=1}^q \|f_j\|_n} \quad \text{and} \quad \phi(k, \xi) = \min_{J \subseteq [q], |J| \leq k} \left\{ \min_{f \in B_{k, \xi}} \frac{\sqrt{k}\|f\|_n}{\sum_{j \notin J} \|f_j\|_n} \right\}.$$

As we discuss below, constants $\phi(2k, \xi)^{-1}$ appear in the error bounds for the Group Lasso-based approach, while constants $\psi(k, \xi)^{-1}$ appear in some of the bounds that we establish for $\hat{f}$. The following result establishes a useful relationship for these quantities.

**Proposition 5.** For all positive integers $k$ and all $\xi \in (1, \infty]$, $\psi(2k, \xi) \geq \phi(k, \xi) / \sqrt{2}$.

We assume that the observed data follows the model $y = f^* + \epsilon$, where $f^* \in C_{gr}$, and the elements of $\epsilon$ are independent $N(0, \sigma^2)$ with $\sigma > 0$. We refer to $\|\hat{f} - f^*\|_n^2$ as the prediction error for estimator $\hat{f}$. We write $r_n = n^{-m/(2m+1)}$, suppressing the dependence on $m$ for notational simplicity, noting that $r_n^2$ is the optimal prediction error rate in the univariate regression setting where $f^* \in C$. For example, in the case where $C$ is the second order Sobolev space, which corresponds to $m = 2$, the above rate is $r_n^2 = n^{-4/5}$. We define $\alpha = 1/(4m + 2)$ and note that $\alpha = 1/10$ when $m = 2$. The next result, in which we treat $m \geq 1$ as a fixed integer, establishes prediction error bounds for the proposed approach.

**Theorem 6.** Let $k_* = G(f^*)$ and consider optimization Problem (36) with $k \geq k_*$. There exists a universal constant $c_1$, such that if $\lambda_n \geq c_1 \sigma \left[ k^{2\alpha} r_n^2 + k^\alpha r_n \sqrt{\log(eq/k)/n} \right]$, then

$$\|\hat{f} - f^*\|_n^2 \lesssim \sigma^2 k \left[ k^{2\alpha} r_n^2 + \frac{\log(eq/k)}{n} \right] + \lambda_n \text{Pen}_{gr}(f^*)$$

with probability at least $1 - (k/q)^k$. Furthermore, for every $\xi \in (1, \infty]$, there exists a finite constant $c_2$, which depends only on $\xi$, such that if $\lambda_n \geq c_2 \sigma \left[ r_n^2 + r_n \sqrt{\log(q)/n} \right]$, then

$$\|\hat{f} - f^*\|_n^2 \lesssim \sigma^2 k \left[ \frac{\log(q)}{n} \right] \left[ \psi(2k, \xi) \right]^{-2} + \lambda_n \text{Pen}_{gr}(f^*)$$

with probability at least $1 - 1/q$. 
We make the following observations regarding the established error bounds. To simplify the comparison of the error rates, we focus on the setting where \( k = k_\ast \) and \( \text{Pen}_{gr}(f^*) \propto \sigma k_\ast \). The last relationship holds, for example, when the scaled roughness of each nonzero component, \( \text{Pen}(f_j^*)/\sigma \), is bounded above and below by positive universal constants.

**Remark 4.** The expression in error bound (37) is optimized for the setting where \( \text{Pen}_{gr}(f^*) \propto \sigma k_\ast \). However, as we show in the proof, the bound can be improved when \( \sigma k_\ast \) and \( \text{Pen}_{gr}(f^*) \) have different orders of magnitude.

**Remark 5.** The prediction error rate provided in (38) is analogous to the rate established in [61] for the Group Lasso-based approach\(^{12}\), however, the latter rate replaces \( \psi(2k_\ast, \xi)^{-2} \) with \( \phi(k_\ast, \xi)^{-2} \). By Proposition 5, the former rate is at least as good as the latter, with a potential improvement due to the additional \( \ell_0 \) group sparsity requirement in the definition of \( \psi \). If for some fixed \( \xi > 1 \) quantity \( \psi(2k_\ast, \xi)^{-1} \) is bounded by a universal constant, then inequality (38) yields the following prediction error rate:

\[
\|
\hat{f} - f^* \|_n^2 \lesssim \sigma^2 k_\ast \left[ r_n^2 + \frac{\log(q)}{n} \right].
\]

This rate matches the one established in [61] for the Group Lasso-based approach under an analogous (but somewhat stronger) assumption\(^{13}\) on \( \phi(k_\ast, \xi)^{-1} \).

**Remark 6.** Bound (37) yields the following error rate without imposing assumptions on the design:

\[
\|
\hat{f} - f^* \|_n^2 \lesssim \sigma^2 k_\ast \left[ k_\ast^{2\alpha} r_n^2 + \frac{\log(eq/k_\ast)}{n} \right].
\]

If \( k_\ast \lesssim 1 \) or \( k_\ast^{2\alpha} r_n^2 \lesssim \log(eq/k_\ast)/n \), then the above expression can be upper-bounded by

\[
\sigma^2 k_\ast \left[ r_n^2 + \frac{\log(eq/k_\ast)}{n} \right].
\]

Thus, \( \hat{f} \) achieves the corresponding minimax lower bound on the prediction error [57, 60, 61].

**Remark 7.** When \( q = k_\ast \), the prediction error rate given by bound (37) is \( k_\ast^{1+1/(2m+1)} r_n^2 \), which improves over the corresponding \( k_\ast^{1+\ell_0/(2m+1)} r_n^2 \) rate\(^{14}\) derived in [40]. In particular, when \( m = 2 \), the former rate is \( k_\ast^{6/5} n^{-4/5} \), while the latter is \( k_\ast^{8/5} n^{-4/5} \). The improvement in the rate is a consequence of the more refined entropy bounds derived in our proofs.

**Remark 8.** In the special case of \( m = 2 \) and \( k_\ast \lesssim 1 \), bound (37) yields the prediction error rate of \( n^{-4/5} + \log(q)/n \), which matches the optimal univariate rate of \( n^{-4/5} \) when \( \log(q) \lesssim n^{1/5} \).

---

\(^{12}\)To the best of our knowledge, the bounds in [61] are overall the strongest in the literature for the Group Lasso-based approach, due to the relative weakness of the imposed conditions: see the discussion in Remark 12 of [61].

\(^{13}\)For a comprehensive discussion of this assumption, we refer the reader to [48, 61], and the references therein.

\(^{14}\)Theorem 1 in [40] treats the number of predictors \( (q = k_\ast) \) as fixed and omits it from the expression for the error rate. However, an examination of the proof of their Theorem 1 and the entropy bound in their Lemma A.1, which explicitly accounts for the number of predictors, reveals the effect of the dimension \( k_\ast \).
**Remark 9.** If for some fixed $\xi > 1$ quantity $\psi(2k_*, \xi)^{-1}$ is bounded by a universal constant, then a direct consequence of Theorem 6 is the following estimation error rate:

$$\sum_{j=1}^{q} \| \hat{f}_j - f_j^* \|_n \lesssim \sigma k_* \left[ r_n + \sqrt{\log(q) \over n} \right].$$

### 6. Experiments.
We present experiments that shed light on the practical performance of our proposals compared to the state of the art. Our algorithms are implemented in Python and are available at [https://github.com/hazimehh/L0Group](https://github.com/hazimehh/L0Group). In Section 6.1, we investigate the statistical properties of our algorithms for the Group $\ell_0$ problem. In Section 6.2, we present computation times of our MIP algorithm. Section 6.3 investigates nonparametric sparse additive models. Additional numerical experiments can be found in the supplementary material.

We consider both synthetic and real datasets in our experiments, as discussed below.

**Synthetic data generation.** The underlying model is $y = X\beta^* + \epsilon$, where $\beta^* \in \mathbb{R}^p$ has $q$ groups, all with the same size. Once we generate $X$ (see below), every column is standardized to have unit $\ell_2$-norm. The errors $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, $i = 1, \ldots, n$, are independent of $X$, and $\sigma^2$ is chosen to achieve a desired signal-to-noise ratio (SNR)$^{15}$. We note that the SNR values in our experiments are sufficiently high to make the true model support recovery possible.

Two different types of $X$ are considered: (a) **example=1:** We first generate group representatives $\gamma_1, \ldots, \gamma_q \sim \text{MVN}_q(0, \Sigma)$, where, $\Sigma_{q \times q} = (\langle \gamma_i, \gamma_j \rangle)$, with $\langle \gamma_i, \gamma_j \rangle = \rho^{i-j}$. Given a $\gamma_q$, the covariates $x_j, j \in G_q$ are generated by adding independent Gaussian noise to a scalar multiple of $\gamma_q$, to achieve pairwise correlation of 0.9 within the group. (b) **example=2:** Here we take $X \sim \text{MVN}_p(0, \Sigma)$, where $\sigma_{ij} = \rho$, for all $i \neq j$, with $\sigma_{jj} = 1$ for all $j$.

To generate the true population regression coefficients, the $k_*$ nonzero groups are taken to be equally spaced in $\{1, \ldots, q\}$. All the nonzero entries of $\beta^*$ are drawn independently from a standard Gaussian distribution.

**Competing algorithms and tuning.** In the experiments of this section, we focus on the Group $\ell_0$ problem defined in (1), and study the performance of our algorithms. We compare against the following state-of-the-art grouped variable selection methods: Group Lasso (based on $\ell_{2,1}$ regularization), Group MCP, and Group SCAD – these estimators are computed by using the R package `grpreg` [15]. For synthetic data (Sections 6.1.1 and 6.1.2), we construct a separate validation set with a fixed design. We tune the parameters of the different problems to minimize the prediction error on the validation set. Specifically, for each of Group $\ell_0$ and Group Lasso, we tune the regularization parameter over a (one-dimensional) grid with 100 values. For MCP and SCAD, we tune the first parameter $\lambda$ over a grid with 100 values, and leave the second parameter $\gamma$ to its default value in `grpreg`. For MCP and SCAD, high-dimensional BIC (HBIC) is another method for doing tuning parameter selection [67] – we present results for MCP and SCAD based on HBIC tuning following [67] in the Supplementary material (Section C).

**Performance measures.** Given an estimator $\hat{\beta}$, we consider the following performance measures:

- **True Positives (TP):** The number of nonzero groups that are in both $\hat{\beta}$ and $\beta^*$.
- **False Positives (FP):** The number of nonzero groups in $\hat{\beta}$ but not in $\beta^*$

---

$^{15}$For a generative model of the form $y_i = \mu_i + \epsilon_i$, we define SNR = $\text{Var}(\mu)/\text{Var}(\epsilon)$. 

• **Recovery F1 Score**: The harmonic mean of precision and recall, i.e., $F1 \text{ Score} = \frac{2PR}{P + R}$, where $P = \frac{TP}{TP + FP}$ is precision and $R = \frac{TP}{k_\ast}$ is recall. We note that an F1 Score of 1 implies perfect support recovery.

• **Test MSE**: This is defined as $\frac{1}{n} \| \hat{\beta} - \hat{\beta}^* \|_2^2$.

6.1.1. **Statistical performance for varying number of observations.** In this experiment, we study the effect of varying the number of observations $n$ on the performance of Group $\ell_0$ and other state-of-the-art group regularizers (Group Lasso, MCP, and SCAD). We obtain approximate estimators to the Group $\ell_0$ problem using Algorithms 1 and 2 (with $m = 1$). We generate 500 datasets having exponentially decaying correlation (i.e., under example=1) with a correlation parameter $\rho = 0.9$, $p = 5000$, a group size of 4, number of nonzero groups $k_\ast = 25$, and SNR = 10. This setting is relatively difficult for recovery as each group is highly correlated with a few others. We report the mean and the standard error for each performance measure in Figure 1.

![Graphs showing the performance measures](image)

**Figure 1.** Performance measures for varying number of observations on a synthetic dataset with highly correlated features. The measures are averaged over 500 repetitions, and the standard error is represented using error bars. Alg. 1 and Alg. 2 are our proposed algorithms. Here, “Lasso” is a shorthand for Group Lasso, we use the same convention for SCAD, MCP. All methods are tuned by minimizing the prediction error on a validation set. The average running time (in seconds) across $n$ is 11 for Algorithm 1, 42 for Algorithm 2, and 4 for Group Lasso, MCP, and SCAD. This difference in running time may be in part due to the efficient C implementation used in `grpreg`, as opposed to our prototype in Python.

Figure 1 shows that Algorithm 2 notably outperforms the other methods in terms of variable selection; it perfectly recovers the support for $n \approx 2000$. Group MCP and SCAD require roughly 4500 observations to recover the true support, whereas Group Lasso does not recover the support even when $n = p$. Moreover, Algorithms 1 and 2 attain the smallest support sizes for any $n$, whereas the other methods require much larger supports, especially for small $n$. Algorithm 2 has the lowest test MSE for all $n$. The test MSE of MCP matches that of Algorithm 2 in most of the cases, while the other methods lag behind. We also note that there is a gap between the test MSE of Algorithms 1 and 2. This difference is likely due to Algorithm 2 doing a better job in optimization.

In Supplementary Material Section C, we report the results of the same experiment but with lower feature correlation ($\rho = 0.5$ and $\rho = 0$). The performance of all methods improve as correlation decreases, but the results are qualitatively similar to what we see in Figure 1. In addition, we also report results based on HBIC tuning for Group MCP and SCAD regularizers – with HBIC tuning, the sparsity of Group MCP and SCAD estimators generally improves, but their prediction performance suffers.
6.1.2. **Statistical performance on high-dimensional instances.** We compare the performance of the different methods under two high-dimensional settings. In both settings, we generate data with constant correlation (i.e., under example = 2) and SNR = 10. Below is a description of the settings:

- Setting 1: $\rho = 0.9, n = 1000, p = 100,000, k = 10$, and a group size of 10.
- Setting 2: $\rho = 0.3, n = 1000, p = 100,000, k = 20$, and a group size of 4.

For each setting, we generate 500 replications, on which we train and tune the algorithms. The tuning here is based on validation MSE — results based on HBIC tuning for (group) SCAD and MCP estimators are presented in the supplementary material (Section C). To ensure a fair comparison in terms of running time, we solve the Group $\ell_0$ problem approximately using Algorithm 2 (with $m = 1$), which typically has the same order of running time (seconds in this case) as the other group selection methods considered here. We report the averaged results for Settings 1 and 2 in Table 1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$|\hat{\beta}|_0$</th>
<th>TP</th>
<th>FP</th>
<th>MSE</th>
<th>$|\hat{\beta} - \beta^*|_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setting 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group $\ell_0$</td>
<td>100.3 (1.2)</td>
<td>8.9</td>
<td>1.1</td>
<td>19.2</td>
<td>1.17 (0.03)</td>
</tr>
<tr>
<td>Group Lasso</td>
<td>2086.1 (27.8)</td>
<td>9.6</td>
<td>199.0</td>
<td>26.9</td>
<td>1.44 (0.02)</td>
</tr>
<tr>
<td>Group MCP</td>
<td>294.8 (6.7)</td>
<td>9.4</td>
<td>20.1</td>
<td>19.8</td>
<td>1.10 (0.04)</td>
</tr>
<tr>
<td>Group SCAD</td>
<td>686.3 (18.6)</td>
<td>9.5</td>
<td>59.1</td>
<td>24.5</td>
<td>1.21 (0.03)</td>
</tr>
<tr>
<td><strong>Setting 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group $\ell_0$</td>
<td>79.4 (0.1)</td>
<td>19.7</td>
<td>0.2</td>
<td>1.12</td>
<td>0.356 (0.007)</td>
</tr>
<tr>
<td>Group Lasso</td>
<td>1126.5 (11.7)</td>
<td>19.8</td>
<td>261.8</td>
<td>5.06</td>
<td>0.703 (0.007)</td>
</tr>
<tr>
<td>Group MCP</td>
<td>139.6 (2.1)</td>
<td>19.8</td>
<td>15.1</td>
<td>1.20</td>
<td>0.352 (0.006)</td>
</tr>
<tr>
<td>Group SCAD</td>
<td>310.4 (4.9)</td>
<td>19.9</td>
<td>57.8</td>
<td>1.56</td>
<td>0.415 (0.008)</td>
</tr>
</tbody>
</table>

Under both settings, Group $\ell_0$ selects significantly smaller support sizes and false positives than other methods, and is more consistent across the replications (as evidenced by the small standard error). For example, in Table 1 (top), Group $\ell_0$ has a support size which is roughly 20 times smaller than the one for the Lasso and 3 times smaller than one for MCP. For fewer instances, one true positive is missed in Group $\ell_0$, but the difference with the other methods is marginal. In terms of MSE and the estimation error (i.e., $\|\hat{\beta} - \beta^*\|_\infty$), Group $\ell_0$ appears to outperform the other methods, with the differences being most pronounced in the high correlation setting of Table 1 (top). This aligns with the results in Figure 1, where we saw that Group $\ell_0$ leads to important improvements when features are highly correlated and $n$ is small.

6.1.3. **Real data.** We study the performance of the different methods on the Amazon Reviews dataset [29]. As the focus of this paper is on group variable selection in the case where the non-overlapping groups are pre-specified, we perform pre-processing of the features to obtain a grouping of the features (based on simple exploratory analysis). All the different group-sparse estimators make use of the same group structure — this allows us to fairly compare their performance given the chosen group-structure. We note that the downstream results depend upon the input group structure. After preprocessing, the dataset consists of 3482 predictors divided into 100 groups. We use 3500 and 2368 observations for training and testing, respectively. Additional details on the dataset and preprocessing are discussed in the Supplement D. On this dataset, we fit regularization paths for Group $\ell_0$, Lasso, and SCAD.\(^\text{16}\) For

\(^{16}\)We also tried group MCP, but the solver faced numerical problems — hence, their results are not reported.
GROUPED VARIABLE SELECTION WITH MIP

27

F
IG
2

Test MSE on the Amazon Reviews dataset \((n = 3500, p = 3368,\text{ and } q = 100)\). For Group \(\ell_0\), we consider additional ridge regularization and vary the corresponding regularization parameter \(\lambda_2 \in \{0.5, 1, 2\}\).

In Figure 2, we plot the test MSE at different sparsity levels. The results indicate that the lowest MSE is roughly the same for Group \(\ell_0\) \((\lambda_2 = 1)\), Lasso, and SCAD; with Group \(\ell_0\) having a clear advantage in terms of the support size. Specifically, Group \(\ell_0\) with \(\lambda_2 = 1\) attains the lowest MSE at 5 groups whereas Group Lasso and SCAD require around 60 groups to achieve a similar MSE performance.

In the Supplement C.4, we report results on another real dataset; and our conclusions are qualitatively similar to the example in Figure 2.

6.2. MIP-based global optimality certificates: Timing comparisons. Here, we compare the running time of our BnB solver with Gurobi for obtaining globally optimal solutions (we note that Algorithms 1, 2 presented earlier are approximate algorithms.) We generate synthetic data under \texttt{example=2}, and we study the effect of the number of predictors \(p\) on the running time. Specifically, we vary \(p \in \{10^3, 10^4, 10^5, 5 \times 10^6\}\) and fix the other data generation parameters as follows: group size of 10, \(n = 10^3\), \(\rho = 0.1\), \(k_* = 5\), \(\text{SNR} = 10\), and set all nonzero coefficients in \(\beta^*\) to 1. We limit our largest problem instance to \(n = 10^3\) and \(p = 5 \times 10^6\) due to memory limitations – as our data-matrix \(X\) is dense, generating and storing a copy of it is memory intensive. We solve the MIP in (26) to optimality, for two cases: (i) with ridge regularization \((\lambda_2 > 0)\) and (ii) without ridge regularization \((\lambda_2 = 0)\). In both cases, we fix \(\lambda_1 = 0\). For case (i), we choose \((\lambda_0, \lambda_2)\) so that the solution obtained has \(k_*\) nonzero groups and minimizes the \(\ell_2\) estimation error. More formally, for a fixed choice of \((\lambda_0, \lambda_2)\), let \(\theta(\lambda_0, \lambda_2)\) denote a solution of (26). Then, we choose the parameters of case (ii) as follows:

\[
(\lambda_0^*, \lambda_2^*) \in \arg\min_{(\lambda_0, \lambda_2)} \|\theta(\lambda_0, \lambda_2) - \beta^*\|_2 \quad \text{s.t.} \quad G(\theta(\lambda_0, \lambda_2)) = k_*,
\]

We estimate \((\lambda_0^*, \lambda_2^*)\) by running Algorithm 2 on a two-dimensional grid with \(\lambda_0 \in \{10^3, 2 \times 10^3, \ldots, 10^4\}\) and \(\lambda_2 \in \{10^{-5}, 10^{-4}, \ldots, 10^{5}\}\). For case (ii), we choose \(\lambda_0\) so that the corresponding solution has \(k_*\) nonzero groups. Let \(S^*\) be the support of the true solution \(\beta^*\), and let \(\hat{\beta}\) be the solution obtained by solving \(\min_{\beta} \ell(\beta) \quad \text{s.t.} \quad \beta(S^*) = 0\). Then, in both cases, we set \(M_u\) to \(\max_{g \in [q]} \|\hat{\beta}_g\|_2\). For the two solvers, we set the optimality gap\(^{18}\) to 1% and use a

\(^{17}\)This is found to be useful here due to high feature correlations within a group.

\(^{18}\)Given an upper bound UB and a lower bound LB, the optimality gap is defined as \((UB-LB)/UB\).
warm start obtained from Algorithm 2. The running times were measured on a cluster with CentOS 7. Each job (i.e., a single run of a solver over one dataset) was allocated 4 cores of an Intel Xeon Gold 6130 CPU @ 2.10GHz processor and up to 120 GB of RAM. For each job, we set a time limit of 24 hours.

Table 2
Running time in seconds for solving Problem (26) to optimality. A dash (-) indicates that Gurobi cannot solve the problem in 24 hours and has an optimality gap of 100% upon termination.

<table>
<thead>
<tr>
<th>$p$</th>
<th>Case (i): $\lambda_2 = \lambda_2^*$</th>
<th>Case (ii): $\lambda_2 = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ours</td>
<td>Gurobi</td>
</tr>
<tr>
<td>$10^4$</td>
<td>96</td>
<td>24223</td>
</tr>
<tr>
<td>$10^4$</td>
<td>199</td>
<td>-</td>
</tr>
<tr>
<td>$10^5$</td>
<td>231</td>
<td>-</td>
</tr>
<tr>
<td>$10^6$</td>
<td>386</td>
<td>-</td>
</tr>
<tr>
<td>$5 \times 10^6$</td>
<td>1922</td>
<td>-</td>
</tr>
</tbody>
</table>

In Table 2, we report the running time (in seconds) for cases (i) and (ii). In both cases, the results indicate that our BnB can solve instances with $p = 5 \times 10^6$ in the order of minutes to hours, whereas Gurobi cannot solve the problem beyond $p = 10^3$ within the 24-hour time limit. Specifically, for $p \geq 10^4$, Gurobi’s optimality gap is 100%. The reason behind this large gap is that Gurobi cannot solve the root relaxation in the 24-hour time limit, so the best lower bound upon termination is 0. The running times for our BnB solver in case (i) are lower than case (ii), and this can be attributed the perspective reformulation which exploits the presence of the ridge regularizer to speed up computation. It is also worth mentioning that our implementation of BnB is a prototype that does not exploit parallelism (commercial solvers like Gurobi can exploit parallelism). Parallelizing our BnB implementation is expected to make it faster, especially on difficult instances where the search tree is large. In the Supplement Section C.5, we report the running times of our BnB algorithm and Gurobi for different choices of $M_U$.

In Supplement Section C.5, we present an example showing how the runtime of our BnB algorithm changes as $n$ is increased.

6.3. Nonparametric Additive Models. We study an expanded version of the popular Boston Housing dataset19 as an application of our MIP framework to $\ell_0$-sparse modeling. The dataset consists of 13 covariates. To get a better idea about the performance in the presence of irrelevant covariates, we augmented the data with 50 irrelevant covariates. Specifically, we selected 5 covariates uniformly at random. For each selected covariate, we randomly permuted the entries of the covariate vector and augmented the data with the permuted vector—we repeated this step 10 times. This led to 63 covariates in total. We randomly sampled 406 observations for training and 50 observations for validation, and we standardized the response and the covariates. We predict house price using the 63 covariates.

We compare the performance of sparse additive models based on Group $\ell_0$ and Group Lasso. In both approaches, we used B-splines of degree 3 for the basis functions, with 10 knots equi-spaced in the covariates. For the Group $\ell_0$-based approach, we used formulation (6) and tuned $\lambda$ over a grid of 100 values between $10^{-5}$ and $10^{-2}$ (equi-spaced on a logarithmic scale). We obtained the Group Lasso-based approach by relaxing all the binary variables in the MIP formulation of (6) to the interval $[0, 1]$, and we tuned $\lambda$ over a grid of 100 values.

19The dataset was downloaded from https://archive.ics.uci.edu/ml/datasets/Housing.
ranging from $10^{-4}$ to 1 (equi-spaced on a logarithmic scale). In Figure 3, we plot the test MSE versus the number of nonzeros, for each of the two models. The results indicate that the Group $\ell_0$-based approach achieves the minimum test MSE at 7 nonzeros, whereas the Group Lasso-based method achieves its minimum MSE at around 60 nonzeros (without matching the performance of Group L0).

![Figure 3](image.png)

**Fig 3.** Test MSE versus the number of nonzeros on the Boston Housing dataset (with additional noisy covariates).

7. **Concluding Remarks.** We revisit a well-known family of problems in sparse learning where the variables are naturally organized into a collection of pre-specified non-overlapping groups. We study both the associated linear regression problem and the problem of nonparametric additive modelling with smooth components. In contrast to the earlier work, we pursue MIP-based methods to solve the underlying discrete optimization problem to optimality, and at scale. Our algorithmic contributions include (i) fast algorithms based on coordinate descent and combinatorial local search to obtain good feasible solutions; and (ii) an exact algorithm based on a custom branch-and-bound procedure, which exhibits significant speedups compared to off-the-shelf commercial MIP solvers. We present statistical theory for our estimators encompassing both the linear model and the nonparametric additive model settings.

Our paper contributes to the growing body of work exploring MIP-based tools to address a broad range of computational problems that arise in statistics. An incomplete list of recent references includes classification [21], factor driven regression [38], learning acyclic graphs [44], sparse PCA [22], signal estimation [1, 39], among others – we also refer the reader to a recent book [9] and survey [63].

There are several directions for future work. One important application of group-sparse models arises in sparse representation of multiple measurement vectors, where the response is multivariate and the regression coefficients form a matrix [64, 43, 20]. In a series of works, [17, 59, 65] study different convex relaxations of group-$\ell_0$-type formulations and their theoretical properties. Other examples of group-sparse problems can be found in [3, 76]. It will be interesting to extend the approaches presented here to these more general settings.

8. **Acknowledgements.** We would like to thank the Associate Editor and the referees for their thoughtful and constructive comments that helped us improve the paper. We thank Shibal Ibrahim for his help with the Boston Housing dataset experiment. The research was partially supported by the Office of Naval Research (ONR-N000141512342, ONR-N000141812298), National Science Foundation (NSF-IIS-1718258).
SUPPLEMENTARY MATERIAL

Proofs, additional technical details and numerical experiments are presented in the supplementary material.

REFERENCES

GROUPED VARIABLE SELECTION WITH MIP


