Learning Hierarchical Interactions at Scale: A Convex Optimization Approach

Hussein Hazimeh
Massachusetts Institute of Technology

Rahul Mazumder
Massachusetts Institute of Technology

Abstract

In many learning settings, it is beneficial to augment the main features with pairwise interactions. Such interaction models can be often enhanced by performing variable selection under the so-called *strong hierarchy* constraint: an interaction is non-zero only if its associated main features are non-zero. Existing convex optimization-based algorithms face difficulties in handling problems where the number of main features $p \sim 10^3$ (with total number of features $\sim p^2$). In this paper, we study a convex relaxation which enforces strong hierarchy and develop a highly scalable algorithm based on proximal gradient descent. We introduce novel screening rules that allow for solving the complicated proximal problem in parallel. In addition, we introduce a specialized active-set strategy with gradient screening for avoiding costly gradient computations. The framework can handle problems having dense design matrices, with $p = 50,000$ ($\sim 10^9$ interactions)—instances that are much larger than state of the art. Experiments on real and synthetic data suggest that our toolkit *hierScale* outperforms the state of the art in terms of prediction and variable selection and can achieve over a 4900x speed-up.

1 Introduction

In many machine learning applications, augmenting main effects with pairwise interactions can lead to better statistical models. Given a response vector $y \in \mathbb{R}^n$ and data matrix $X = [X_1, X_2, \ldots, X_p] \in \mathbb{R}^{n \times p}$, we consider a linear model of the form:

$$y = \beta_0 + \sum\nolimits_i X_i \beta_i + \sum\nolimits_{i<j} \theta_{ij}(X_i * X_j) + \epsilon, \quad (1)$$

where $*$ denotes element-wise multiplication and ϵ is a noise vector. Above, β_0 is the intercept, $\sum_i X_i \beta_i$ corresponds to the main effects and $\sum_{i<j} \theta_{ij}(X_i * X_j)$ denotes the interaction effects. The goal here is to learn the coefficients β, θ. For small n, this quickly leads to an ill-posed problem as the total number of coefficients is $1+p+\binom{p}{2}$, which can far exceed n. Thus, imposing sparsity can be beneficial from both the statistical and computational viewpoints. While vanilla sparsity-inducing regularization methods (e.g., using the ℓ_0 or ℓ_1 norm) can help, structured sparsity can be much more effective in this setting. Particularly, we consider enforcing sparsity under the *strong hierarchy* (SH) constraint (McCullagh and Nelder, 1989; Bien et al., 2013), which states that an interaction term should be non-zero only if its corresponding main effect terms are both non-zero. SH can be expressed via the following combinatorial statement:

Strong Hierarchy (SH): $\theta_{ij} \neq 0 \Longrightarrow \beta_i \neq 0 \wedge \beta_j \neq 0$

SH is a natural property that is widely used in high-dimensional statistics: it leads to more interpretable models with good predictive performance (Cox, 1984; McCullagh and Nelder, 1989; Bien et al., 2013). Moreover, SH promotes *practical sparsity*, i.e., it reduces the number of main features that need to be measured when making new predictions—this can significantly save on future data collection costs (Bien et al., 2013).

An impressive line of work for learning sparse interactions under SH is based on a regularization framework (see Choi et al. (2010); Radchenko and James (2010); Bien et al. (2013); Lim and Hastie (2015); Yan and Bien (2017); She et al. (2018) and the references therein). These methods minimize the empirical risk with sparsity-inducing regularizers to enforce SH. While these lead to estimators with good statistical properties, computation remains a major challenge. Indeed, the complex nature of the regularizers used to

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

enforce SH prevents most current optimization algorithms from scaling beyond $p \sim 10^3$. However, in many real-world applications p can be in the order of tens of thousands, which is limiting the adoption of SH methods in practice. To address this shortcoming, we propose a convex regularization framework and develop a novel scalable algorithm, by carefully exploiting the problem-specific structural sparsity. Our algorithm can solve the SH learning problem with $p = 50,000$ ($\sim 10^9$ interactions) and achieves significant speed-ups (over 4900x) compared to the state of the art.

1.1 Problem Formulation

Various convex regularizers (a.k.a. penalties) for enforcing SH exist in the literature. In many cases, the choices seem somewhat ad hoc and involve auxiliary variables that can increase the memory and computational footprints. Here we transparently derive our regularizer via a convex relaxation of a mixed integer program (MIP) (Bertsimas and Tsitsiklis, 1997) that enforces SH. In what follows, we denote the interaction column $X_i * X_j$ by $\tilde{X}_{ij}$. Performing variable selection under SH for model (1) can be naturally expressed using ℓ_0 regularization:

$$\begin{aligned} \min_{\beta,\theta} \quad & f(\beta,\theta) + \alpha_1\|\beta\|_0 + \alpha_2\|\theta\|_0 \\ \text{s.t.} \quad & \theta_{ij} \neq 0 \implies \beta_i \neq 0 \text{ and } \beta_j \neq 0 \end{aligned} \tag{2}$$

where $f(\beta,\theta) = \frac{1}{2}\|y - X\beta - \sum_{i<j}\tilde{X}_{ij}\theta_{ij}\|_2^2$ and $\|u\|_0$ denotes the number of non-zeros in the vector u. The parameters α_1 and α_2 control the number of non-zeros. Note that we ignore the intercept term to simplify the presentation.

Problem (2) can be expressed using the following MIP[1], in which we introduce auxiliary binary variables to model the SH constraint and the ℓ_0 (pseudo) norms:

$$\begin{aligned} \min_{\beta,\theta,z} \quad & f(\beta,\theta) + \alpha_1 \sum\nolimits_i z_i + \alpha_2 \sum\nolimits_{i<j} z_{ij} \\ \text{s.t.} \quad & |\beta_i| \leq M z_i, \ \forall\ i \\ & |\theta_{ij}| \leq M z_{ij}, \quad z_{ij} \leq z_i, \quad z_{ij} \leq z_j, \ \forall\ i<j \\ & z_i \in \{0,1\} \ \forall\ i, \quad z_{ij} \in \{0,1\} \ \forall\ i<j \end{aligned} \tag{3}$$

where the optimization variables are β, θ, and z. In the above, M is a large constant chosen such that some optimal solution β^*, θ^* to (2) satisfies $\|\beta^*, \theta^*\|_\infty \leq M$. Here $z_i = 0$ implies $\beta_i = 0$ and similarly $z_{ij} = 0$ implies $\theta_{ij} = 0$. Problem (3) is known to be NP-Hard (Natarajan, 1995) and can be very difficult to scale. Thus, our focus will be on solving a convex relaxation.

[1]There can be pathological cases where the MIP does not satisfy SH. However, when y is drawn from a continuous distribution, SH is satisfied w.p. 1. We discuss this in more detail in Section 1 of the appendix.

For easier presentation, we introduce some notation. For every $i \in \{1,2,\ldots,p\}$, we define G_i as the set of indices of all interactions corresponding to β_i, i.e.,

$$G_i \stackrel{\text{def}}{=} \{(1,i),(2,i),\ldots,(i-1,i),(i,i+1),\ldots,(i,p)\}.$$

We use the notation θ_{G_i} to refer to the vector of θ_{ij}'s whose indices are in G_i.

In Lemma 1, we derive a convex relaxation of Problem (3). Specifically, we relax the binary variables in Problem (3) to $[0,1]$, and then simplify the problem to remove the dependence on the z_i's and z_{ij}'s, i.e., we move from the extended (β,θ,z) space back to the original (β,θ) space. The resulting relaxation involves box constraints on the coefficients β,θ—we eliminate these constraints to simplify the problem.

Lemma 1. *(Convex Relaxation) The following is a convex relaxation of Problem* (3)*:*

$$\min_{\beta,\theta} \quad f(\beta,\theta) + \Omega(\beta,\theta), \tag{4}$$

where

$$\Omega(\beta,\theta) \stackrel{\text{def}}{=} \lambda_1 \sum_{i=1}^{p} \max\{|\beta_i|, \|\theta_{G_i}\|_\infty\} + \lambda_2\|\theta\|_1,$$

and $\lambda_1 = \alpha_1/M, \lambda_2 = \alpha_2/M$.

The focus of our paper is on solving (4). We note that the relaxation in (4) belongs to the original (β,θ) space. From an algorithmic perspective, this space is easier to work with compared to an extended space (e.g., the one involving z). She et al. (2018) propose a general family of problems for enforcing SH, which includes our relaxation above as a special case (however, they do not discuss the connections to the original Problem (2)). The solutions of (4) satisfy SH with probability one under model (1) when $\epsilon \sim N(0,\sigma^2 I)$ (see She et al. (2018)). For a general response y, there can be pathological cases where SH is not satisfied by (4), however, these cases rarely appear in practice and are usually not considered by current regularization-based approaches for SH (Radchenko and James, 2010; Bien et al., 2013; Lim and Hastie, 2015; She et al., 2018).

1.2 Contributions

The main contribution of our work is developing a scalable algorithm for solving (4). Our proposal is based on proximal gradient descent (PGD). However, PGD is limited by two major computational bottlenecks, due to the scale of the problem. First, the proximal problem does not admit a closed-form solution, so solving it requires running an iterative optimization algorithm over $\mathcal{O}(p^2)$ variables. Second, the repeated gradient

computations can be very expensive as each computation requires $\mathcal{O}(np^2)$ operations.

To mitigate these bottlenecks, we **(i)** introduce new *proximal screening rules* that can efficiently identify many of the zero variables and groups in the proximal problem, **(ii)** demonstrate how our proposed screening rules can decompose the proximal problem so that it can be solved in parallel, and **(iii)** develop a specialized active-set algorithm along with a novel *gradient screening* method for avoiding costly gradient evaluations. We open source the implementation through our toolkit hierScale[2]. Moreover, we demonstrate how our algorithm scales to high-dimensional problems having dense matrices with $p = 50,000$ ($\sim 10^9$ interactions), achieving over 4900x speed-ups compared to the state of the art.

1.3 Related Work

Many methods for enforcing SH exist in the literature. The methods can be broadly categorized into multi-step methods (e.g., Wu et al. (2010); Hao and Zhang (2014)), Bayesian and approximate methods (e.g., Chipman (1996); Thanei et al. (2018)), and optimization-based methods. We discuss relevant work in the latter category as they are directly related to our work. Choi et al. (2010) re-parameterize the interactions problem so that $\theta_{ij} = \gamma_{ij}\beta_i\beta_j$ (where γ_{ij} is an optimization variable) and enforce sparsity by adding an ℓ_1 norm regularization on β and θ—this approach, however, leads to a challenging non-convex optimization problem. Radchenko and James (2010) enforce SH by using ℓ_2 regularization on the predictions made by every group. Bien et al. (2013) propose the Hierarchical Lasso, which shares a similar objective with our problem, except that $\|\theta_{G_i}\|_1$ is used instead of $\|\theta_{G_i}\|_\infty$. They use ADMM (Boyd et al., 2011) for computation. Lim and Hastie (2015) propose an overlapped group Lasso formulation and solve it using a variant of the FISTA algorithm (Beck and Teboulle, 2009) along with strong screening rules (Tibshirani et al., 2012). Their toolkit glinternet seems to be the fastest toolkit for learning sparse interactions we are aware of. She et al. (2018) consider a formulation similar to (4), but with the ℓ_∞ norm replaced with ℓ_2 norm, and develop prediction error bounds and a splitting-based algorithm—the largest problem they consider has $p = 1000$.

Mairal et al. (2011) consider learning problems regularized with sum of ℓ_∞ norms over groups (with potential overlaps), which includes our problem as a special case. They show that the proximal operator can be efficiently solved using network flow algorithms and propose algorithms based on PGD and ADMM. Our approaches differ: here we exploit the specific structure of our objective function (particularly the presence of both the ℓ_∞ and ℓ_1 norms) to derive the proximal screening rules and decompose the proximal problem—such screening/decomposition rules are not discussed in Mairal et al. (2011). We also note that Jenatton et al. (2011) develop a scalable PGD-based algorithm for problems regularized with sums of ℓ_∞ or ℓ_2 norms, under a tree structure constraint. However, our model does not satisfy this constraint and thus their algorithms are not applicable. Many other works also consider algorithms for structured sparsity and discuss interesting connections to submodularity (e.g., see Bach (2009, 2010, 2013) and the references therein). Unfortunately, prior work cannot easily scale beyond p in the hundreds to few thousands. A main reason is that the standard algorithms (e.g., PGD and ADMM) are limited by costly gradient evaluations and by solving expensive sub-problems (e.g., solving the proximal problem in PGD requires an iterative optimization method). Our proposal addresses these key computational bottlenecks.

Notation and supplement: We use the notation $[p]$ to refer to the set $\{1, 2, \ldots, p\}$. We denote the complement of a set A by A^c. For a set $A \subseteq [p]$, β_A refers to the sub-vector of β restricted to coordinates in A. We use $\nabla_{\beta_A, \theta_B} f(\beta, \theta)$ to refer to the components of $\nabla f(\beta, \theta)$ corresponding to the vectors β_A and θ_B. For any scalar a, we define $[a]_+ = \max\{a, 0\}$. A function $u \mapsto g(u)$ is said to be Lipschitz with parameter L if $\|g(u) - g(v)\|_2 \leq L\|u - v\|_2$ for all u, v. Proofs of all lemmas and theorems are in the supplementary file.

2 Proximal Screening and Decomposition

To solve the non-smooth convex problem (4), we use PGD (Beck and Teboulle, 2009; Nesterov, 2013), which is an effective and popular choice for handling structured sparse learning problems (e.g., see Bach (2010); Mairal et al. (2011); Chen et al. (2012) and references therein). However, there are two major bottlenecks: (i) solving the proximal problem which does not admit a closed-form solution and (ii) the repeated gradient computations each requiring $\mathcal{O}(np^2)$ operations. In this section, we address bottleneck (i) through new proximal screening and decomposition rules, and we handle (ii) in Section 3 using active-set updates and gradient screening.

We first present the basic PGD algorithm below.

Algorithm 1: Proximal Gradient Descent

- Input: β^0, θ^0 and L: the Lipschitz parameter of the

[2] `https://pypi.org/project/hierScale`

gradient map $(\beta, \theta) \mapsto \nabla f(\beta, \theta)$.

- **For** $k \geq 0$ repeat the following till convergence:

$$\tilde{\beta} \leftarrow \beta^k - \nabla_\beta f(\beta^k, \theta^k)/L, \quad \tilde{\theta} \leftarrow \theta^k - \nabla_\theta f(\beta^k, \theta^k)/L$$

$$\begin{bmatrix} \beta^{k+1} \\ \theta^{k+1} \end{bmatrix} \leftarrow \arg\min_{\beta,\theta} \frac{L}{2} \left\| \begin{bmatrix} \beta - \tilde{\beta} \\ \theta - \tilde{\theta} \end{bmatrix} \right\|_2^2 + \Omega(\beta, \theta) \quad (5)$$

The sequence $\{\beta^k\}$ generated by Algorithm 1 is guaranteed to converge to an optimal solution (Beck, 2017). The objective values converge at a rate of $\mathcal{O}(1/k)$, and this can be improved to $\mathcal{O}(1/k^2)$ by using accelerated PGD (Nesterov, 2013; Beck, 2017). However, there is no closed-form solution for the proximal problem in (5)—this is due to the overlapping variables in the ℓ_∞ norms. Thus, iterative optimization algorithms are needed to solve (5). For example, Mairal et al. (2011) presents a dual reformulation and an efficient network flow algorithm for solving a class of proximal problems which includes (5). In the appendix, we present an alternative dual which uses less variables (as we exploit the specific structure of our problem) and present a dual block coordinate ascent (BCA) algorithm for solving it. However, iterative algorithms (e.g., Mairal et al. (2011)'s or our proposed BCA) require significant time to solve (5) when p is large, which can lead to a serious bottleneck.

In what follows, we present algorithm-agnostic schemes to speed up solving the proximal problem. In Section 2.1, we introduce screening rules to eliminate variables from the proximal problem (prior to optimization). In Section 2.2, we demonstrate how these rules can decompose the proximal problem, which allows for solving it in parallel.

2.1 Proximal Screening

Typically, we expect the solutions of (5) to be sparse, as $\Omega(\beta, \theta)$ incorporates sparsity-inducing norms. To exploit this sparsity, we propose new *proximal screening* rules, which can efficiently identify many of the zero groups and variables in (5). We present the rules in Theorem 1.

Theorem 1. *(Proximal Screening) Let β^*, θ^* be the optimal solution of Problem (5). Then, the following group-level rule holds for every $i \in [p]$:*

$$\sum_{t \in G_i} \left[|\tilde{\theta}_t| - \frac{\lambda_2}{L} \right]_+ \leq \frac{\lambda_1}{L} - |\tilde{\beta}_i| \implies \beta_i^*, \theta_{G_i}^* = 0, \quad (6)$$

and the following feature-level rule holds for $i < j$:

$$|\tilde{\theta}_{ij}| \leq \frac{\lambda_2}{L} \implies \theta_{ij}^* = 0. \quad (7)$$

The rules in Theorem 1 can be used to optimize (5) over a smaller set of variables (i.e., only over the variables that do not pass the screening checks). These rules are easy to check. Particularly, the rule in (6) requires $\mathcal{O}(p)$ operations to screen a group of p variables. The feature-level rule allows us to set θ_{ij}'s with $|\tilde{\theta}_{ij}| \leq \lambda_2/L$ to zero. The group-level rule is less restrictive: in group i, the θ_{ij}'s with $|\tilde{\theta}_{ij}| > \lambda_2/L$ can be still set to zero if $|\tilde{\beta}_i|$ is sufficiently small (i.e., if the contribution of main effect i is weak).

Related Work on Screening Rules: Our proposed rules are safe in the sense that only variables that are zero in the optimal solution of (5) can be discarded. However, our rules are different from the *safe screening rules* used in the literature (e.g., Ghaoui et al. (2010); Wang et al. (2013); Bonnefoy et al. (2015); Lee and Xing (2014); Wang and Ye (2014); Ndiaye et al. (2016); Nakagawa et al. (2016); Ndiaye et al. (2017)) that are designed to identify zero variables in the full problem. In particular, our rules can potentially eliminate more variables in the proximal problem compared to the safe rules, since a variable can be safe to eliminate from a proximal problem but not from the full problem. This allows for exploiting more parallelism when solving the proximal problem (see Theorem 2). Moreover, the state-of-the-art safe rules (e.g., the sequential rules of Wang et al. (2013), and the dynamic and Gap rules of Bonnefoy et al. (2015) and Ndiaye et al. (2017)), update (improve) the rules as the algorithm progresses, by leveraging previous solutions. However, every such rule update entails a full gradient computation, which can be very costly in our problem (see Ndiaye et al. (2017) for a survey and a discussion on this computational issue). On the other hand, our rules do not require gradient computations. In our experiments, we compare against glinternet (Lim and Hastie, 2015) which uses strong screening rules (an approximate and aggressive variant of safe rules proposed in Tibshirani et al. (2012)).

2.2 Proximal Decomposition

An important consequence of Theorem 1 is that it usually decomposes Problem (5) into many independent smaller optimization problems. This allows solving (5) in parallel. Before presenting the decomposition formally, we give a simple motivating example.

Example 1. *Suppose $p = 2$ with one interaction effect; and rule (7) has identified $\theta_{12}^* = 0$. We can now eliminate θ_{12} and solve the proximal problem (5) with $\Omega(\beta, \theta) = \lambda_1|\beta_1| + \lambda_1|\beta_2|$. This decomposes the problem into two independent optimization tasks involving β_1 and β_2—the solutions can be easily obtained via soft-thresholding.*

Next, we formalize the idea of decomposition. Let us define $\mathcal{V}$ as the set of indices of the groups that do not pass the screening test in (6), i.e.,

$$\mathcal{V} = \Big\{ i \in [p] \;\; \Big| \sum_{t \in G_i} \Big[|\tilde{\theta}_t| - \frac{\lambda_2}{L} \Big]_+ > \frac{\lambda_1}{L} - |\tilde{\beta}_i| \Big\}. \quad (8)$$

We also define $\mathcal{E}$ as the set of interaction indices that do not pass the test in (7) and whose corresponding main indices are in $\mathcal{V}$, i.e.,

$$\mathcal{E} = \Big\{ (i,j) \in \mathcal{V}^2 \;\; | \; |\tilde{\theta}_{ij}| > \lambda_2 / L \Big\}. \quad (9)$$

Definition 1. *(Connected Components) Define the simple undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V}$ is defined in* (8) *and the edge set $\mathcal{E}$ is defined in* (9)*. Let the κ connected components of $\mathcal{G}$ be denoted by $\{\mathcal{G}_l\}_1^\kappa$, where $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l),\ l \in [\kappa]$.*

Theorem 2 states that Problem (5) can be solved by decomposing it into smaller independent optimization problems, each corresponding to a connected component of the graph $\mathcal{G}$.

Theorem 2. *Let β^*, θ^* be the optimal solution of the proximal problem in* (5)*. Then, $\beta^*_{\mathcal{V}^c} = 0, \quad \theta^*_{\mathcal{E}^c} = 0$, and for every $l \in [\kappa]$:*

$$\begin{bmatrix} \beta^*_{\mathcal{V}_l} \\ \theta^*_{\mathcal{E}_l} \end{bmatrix} = \underset{\beta_{\mathcal{V}_l}, \theta_{\mathcal{E}_l}}{\arg\min} \frac{L}{2} \left\| \begin{bmatrix} \beta_{\mathcal{V}_l} - \tilde{\beta}_{\mathcal{V}_l} \\ \theta_{\mathcal{E}_l} - \tilde{\theta}_{\mathcal{E}_l} \end{bmatrix} \right\|_2^2 + \Omega(\beta_{\mathcal{V}_l}, \theta_{\mathcal{E}_l}).$$

Theorem 2 allows for solving (5) in parallel. The extent of parallelism depends on the number of the connected components and their sizes. In practice, we solve the problem for a regularization path with warm starts[3], along with active-set updates (discussed in Section 3). These can significantly reduce the sizes of the connected components. Indeed, our experiments on real high-dimensional datasets (with $p \sim 5000$), indicate that the maximum number of vertices and edges in each connected component is in the order of hundreds to few thousands (for all solutions in the path)—see the supplementary for details. The majority of the connected components are typically isolated (i.e., composed of a single vertex), so their corresponding optimization problems can be solved by a simple soft thresholding operation. However, we note that, in the absence of warm starts, the number of edges can grow into hundreds of thousands for the same datasets.

3 Active Sets and Gradient Screening

In Section 2, we addressed the bottleneck of solving the proximal problem. In this section, we focus on another major bottleneck: the repeated full gradient computations in PGD. Particularly, we develop an active-set algorithm that exploits the screening rules and decomposition we introduced in Section 2 to reduce the number of full gradient computations. Moreover, we introduce a new gradient screening method which aids in reducing the cost of every gradient computation.

3.1 Active Set Updates

Every evaluation of $\nabla f(\beta, \theta)$ in Algorithm 1 requires $\mathcal{O}(np^2)$ operations, which can take several minutes on a modern machine when $p \sim 50,000$. To minimize the number of these full gradient evaluations, we propose an active-set algorithm: we run Algorithm 1 over a small subset of variables, namely, the active set. After obtaining an optimal solution restricted to the active set, we augment the set with the variables that violate the optimality conditions (if any) and resolve the problem on the new set. Such an approach is effectively used for speeding up sparse learning algorithms in other contexts (Meier et al., 2008; Friedman et al., 2010; Morvan and Vert, 2018).

Our active set is defined by the sets $\mathcal{A}$ and $\mathcal{T}$ containing indices of the main effects and interaction effects (respectively) to be included in the model. Given $\mathcal{A}$ and $\mathcal{T}$ we consider

$$\begin{aligned} \hat{\beta}, \hat{\theta} \in \arg\min_{\beta,\theta} \quad & f(\beta, \theta) + \Omega(\beta, \theta) \\ \text{s.t.} \quad & \beta_{\mathcal{A}^c} = 0, \quad \theta_{\mathcal{T}^c} = 0, \end{aligned} \quad (10)$$

which is solved with Algorithm 1 *restricted* to the active set. To check whether $\hat{\beta}, \hat{\theta}$ is optimal for Problem (4), we run a single iteration of Algorithm 1 over all the variables (including those outside the active set) – we refer to this as the *master iteration*. If the master iteration does not change the support (i.e., the non-zeros), then the solution $\hat{\beta}, \hat{\theta}$ is optimal. Otherwise, we augment $\mathcal{A}$ and $\mathcal{T}$ with the variables that became non-zero (after the iteration) and solve (10) again. To speed up the costly master iteration, we perform screening and decompose the master iteration as described in Theorem 2 so that it can be solved in parallel. We present the algorithm more formally below:

Algorithm 2: Parallel Active-set Algorithm

- Input: Initial estimates of $\mathcal{A}$ and $\mathcal{T}$
- **Repeat** until convergence:
  1. Solve the problem restricted to the active set, i.e., (10) to get a solution $\hat{\beta}, \hat{\theta}$.
  2. Compute $\nabla_\beta f(\hat{\beta}, \hat{\theta})$ and $\nabla_\theta f(\hat{\beta}, \hat{\theta})$. Set $\tilde{\beta} \leftarrow \hat{\beta} - \frac{1}{L} \nabla_\beta f(\hat{\beta}, \hat{\theta}), \quad \tilde{\theta} \leftarrow \hat{\theta} - \frac{1}{L} \nabla_\theta f(\hat{\beta}, \hat{\theta})$.

[3] A regularization path is the sequence of solutions obtained by solving (4) for a sequence of λ_1's and λ_2's. The solution of the current λ_1, λ_2 is used as a warm start (initial solution) when solving for the next λ_1, λ_2 in the sequence.

3. Construct the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in Definition 1 and find its connected components.
4. Solve (5) in parallel using Theorem 2. If the support stays the same, **terminate**, o.w., augment $\mathcal{A}$ and $\mathcal{T}$ with variables that just became non-zero.

Steps 2-4 in Algorithm 2 are the equivalent of performing one iteration of Algorithm 1 over all variables, while using screening and decomposition. We note that screening and decomposition are also very useful at the active-set level (i.e., for step 1) as we need to repeatedly solve the proximal problem till convergence. Typically, the active-set sizes are relatively small, which can help further mitigate the cost of solving the proximal problem.

3.2 Gradient Screening

Algorithm 2 effectively reduces the total number of gradient computations. However, even a single gradient computation can take minutes for $p \sim 50,000$. Here we propose a novel gradient screening method to reduce the cost of every gradient computation in Algorithm 2. Specifically, every time a gradient is needed, we identify parts of the gradient that are not essential to optimization—this allows for computing a smaller (and cheaper) gradient. This is a major improvement over current active-set approaches, which require a full gradient computation to check optimality.

We note that the full gradient in step 2 of Algorithm 2 is only used to construct the graph $\mathcal{G}$ in step 3. The next lemma, states that only a part of the gradient is needed to construct $\mathcal{G}$.

Lemma 2. *The graph $\mathcal{G}$ in step 2 of Algorithm 2 can be constructed from the following gradients: $\nabla_\beta f(\hat\beta, \hat\theta)$, $\nabla_{\theta_\mathcal{T}} f(\hat\beta, \hat\theta)$, and $\nabla_{\theta_\mathcal{S}} f(\hat\beta, \hat\theta)$, where $\mathcal{S}$ is the critical set defined by*

$$\mathcal{S} = \{(i,j) \in \mathcal{T}^c \mid |\nabla_{\theta_{ij}} f(\hat\beta, \hat\theta)| > \lambda_2\}. \tag{11}$$

Note that $\nabla_\beta f(\hat\beta, \hat\theta)$ is relatively easy to compute, and $\nabla_{\theta_\mathcal{T}} f(\hat\beta, \hat\theta)$ is available as a byproduct of step 1. Thus, if the size of $\mathcal{S}$ in (11) is small, then Lemma 2 suggests a significant reduction in the computation time of step 2. Specifically, if $\mathcal{S}$ is given, computing the gradients in Lemma 2 has a cost $\mathcal{O}(n(p + |\mathcal{S}|))$, which can be much smaller than the cost of full gradient computation $\mathcal{O}(np^2)$. As discussed below, we can obtain $\mathcal{S}$ without explicitly computing $\nabla_{\theta_{ij}} f(\hat\beta, \hat\theta)$ for all $(i,j) \in \mathcal{T}^c$—an operation that costs $\mathcal{O}(np^2)$.

Our key idea is to obtain a set $\hat{\mathcal{S}}$ that is guaranteed to contain $\mathcal{S}$ (i.e., $\mathcal{S} \subset \hat{\mathcal{S}}$), by using currently available information on gradients. Note that $|\hat{\mathcal{S}}|$ can be larger than $|\mathcal{S}|$, but we will require $|\hat{\mathcal{S}}|$ to be significantly smaller than p^2. The next lemma, presents a way to construct $\hat{\mathcal{S}}$ by using the gradient of a solution β^w, θ^w that was obtained prior to $\hat\beta, \hat\theta$ (e.g., from a warm start or a previous iteration of Algorithm 2).

Lemma 3. *Let $\beta^w \in \mathbb{R}^p$ and $\theta^w \in \mathbb{R}^{\binom{p}{2}}$ be arbitrary vectors, and let $\mathcal{S}$ be the critical set defined in* (11). *Define $\gamma = (X\beta^w + \tilde X\theta^w) - (X\hat\beta + \tilde X\hat\theta)$ and $C = \max_{i,j} \|\tilde X_{ij}\|_2$. Then, the following holds:*

$$\mathcal{S} \subseteq \hat{\mathcal{S}} \stackrel{\text{def}}{=} \{(i,j) \in \mathcal{T}^c \mid |\nabla_{\theta_{ij}} f(\beta^w, \theta^w)| > \lambda_2 - C\|\gamma\|_2\}.$$

The set $\hat{\mathcal{S}}$ in Lemma 3 is constructed based on the gradient at a previous estimate (β^w, θ^w); and not at the current point $(\hat\beta, \hat\theta)$. When the predictions made by the estimators $(\hat\beta, \hat\theta)$ and (β^w, θ^w) are close (i.e., small $\|\gamma\|_2$), $\hat{\mathcal{S}}$ is a good estimate of $\mathcal{S}$.

Lemma 3 lays the foundation of our *gradient screening* procedure. During the course of Algorithm 2, we always maintain a solution β^w, θ^w for which we store $|\nabla_\theta f(\beta^w, \theta^w)|$. We replace step 2 in Algorithm 2 with the following gradient screening module:

Gradient Screening

1. Compute $\hat{\mathcal{S}}$ (defined in Lemma 3) and $\nabla_{\theta_{\hat{\mathcal{S}}}} f(\hat\beta, \hat\theta)$ to obtain $\nabla_{\theta_\mathcal{S}} f(\hat\beta, \hat\theta)$.
2. Compute $\nabla_\beta f(\hat\beta, \hat\theta)$ and obtain $\nabla_{\theta_\mathcal{T}} f(\hat\beta, \hat\theta)$. Set $\tilde\beta \leftarrow \hat\beta - \frac{1}{L}\nabla_\beta f(\hat\beta, \hat\theta)$ and $\tilde\theta_{ij} \leftarrow \hat\theta_{ij} - \frac{1}{L}\nabla_{\theta_{ij}} f(\hat\beta, \hat\theta)$ for every $(i,j) \in \mathcal{T} \cup \mathcal{S}$.
3. If $|\hat{\mathcal{S}}| > p$, set $(\beta^w, \theta^w) \leftarrow (\hat\beta, \hat\theta)$ and compute/store $|\nabla_\theta f(\beta^w, \theta^w)|$.

The set $\hat{\mathcal{S}}$ can be identified in $\mathcal{O}(\log p)$ by using a variant of binary search on the sorted entries of $|\nabla_\theta f(\beta^w, \theta^w)|$ (the latter can be sorted once at a cost of $\mathcal{O}(p^2 \log p)$ and stored). Moreover, computing $\nabla_{\theta_{\hat{\mathcal{S}}}} f(\hat\beta, \hat\theta)$ and obtaining $\nabla_{\theta_\mathcal{S}} f(\hat\beta, \hat\theta)$ is $\mathcal{O}(n(p + |\hat{\mathcal{S}}|))$. Thus, the complexity of gradient screening can be much smaller than $\mathcal{O}(np^2)$. For gradient screening to yield speed-ups, we need $|\hat{\mathcal{S}}| \ll p^2$ (otherwise, the overall complexity will be similar to that with no gradient screening). Thus, in Step 3 in the above, we update β^w, θ^w when $|\hat{\mathcal{S}}|$ becomes large (recall that improving the estimate β^w, θ^w can reduce the size of $|\hat{\mathcal{S}}|$). In particular, Step 3 is performed when $|\hat{\mathcal{S}}| > p$; we choose the threshold p to ensure that $|\hat{\mathcal{S}}|$ stays in the order of p in subsequent iterations[4]. The initial β^w, θ^w can be obtained from a previous solution in the regularization

[4]Other choices are possible, but it is important that the chosen threshold keeps $|\hat{\mathcal{S}}|$ from growing in the order of p^2.

path. In practice, predictions across consecutive solutions in the path are usually close, making $\hat{\mathcal{S}}$ close to $\mathcal{S}$—this explains the multi-fold speed-ups we observe in our experiments.

4 Experiments

We study the empirical performance of our algorithm and compare with popular toolkits for learning interactions under SH: hierNet (Bien et al., 2013) and glinternet (Lim and Hastie, 2015); in addition to Boosting with trees using XGBoost (which is often used to learn interactions). We also use the optimization toolbox SPAMS (Mairal et al., 2011) to solve (4) and compare its running time with hierScale.

Our Toolkit hierScale: We open-sourced our toolkit hierScale (`https://pypi.org/project/hierScale/`), written in Python with critical code sections compiled into machine code using Numba (Lam et al., 2015). hierScale has a low memory footprint (it generates interaction columns on the fly) and supports multi-core computation.

Synthetic Data Generation: We generate $X_{n\times p}$ to be an iid standard Gaussian ensemble; and form $y = X\beta^0 + \tilde{X}\theta^0 + \epsilon$, where $\epsilon_i \overset{\text{iid}}{\sim} N(0,\sigma^2)$ is independent of X. For β^0 and θ^0, we consider three settings: (I) Hierarchical Truth: β^0 and θ^0 satisfy SH, (II) Anti-Hierarchical Truth: $\theta^0_{ij} \neq 0 \implies \beta^0_i = 0, \beta^0_j = 0$, and (III) Main-Only Truth: $\theta^0 = 0$. In (I)–(III), all non-zero coefficients are set to 1. For (I) and (II), we set $\|\theta^0\|_0 = \|\beta^0\|_0$. We take σ^2 such that the signal-to-noise-ratio (SNR) $= \text{Var}(X\beta^0 + \tilde{X}\theta^0)/\sigma^2 = 10$.

Timings: We compare the running time of hierScale versus hierNet, glinternet, and SPAMS, on both synthetic and real datasets. For SPAMS, we use the same objective function of hierScale, and fit a regularization path (with warm starts) using FISTA. We note that SPAMS is not designed to solve interactions problems, so we had to generate the interaction columns apriori, which limits us to problems where the interactions can fit in memory. We generate the synthetic data under hierarchical truth with $n = 1000$ and $\|\beta^0\|_0 = \|\theta^0\|_0 = 5$ and consider p up to $50,000$ ($\sim 10^9$ interactions). Among real data, we consider the Amazon Reviews dataset (Hazimeh and Mazumder, 2018) and use two variants of it: Amazon-1 ($p = 10160$, $n = 1000$) and Amazon-2 ($p = 5000$, $n = 1000$). We also consider the dataset from CoEPrA 2006 (Regression Problem 1) ($p = 5786$, $n = 89$)[5], and the Riboflavin dataset ($p = 4088, n = 71$) (Bühlmann et al., 2014). For all toolkits, we set the tolerance level to 10^{-6}, $\lambda_{\min} = 0.05\lambda_{\max}$ and generate a path with 100 solutions. For hierScale and SPAMS, we set $\lambda_2 = 2\lambda_1$. Computations are carried out on a machine with a 12-core Intel Xeon E5 @ 2.7 GHz and 64GB of RAM.

The results are in Table 1. hierScale achieves over a 4900x speed-up compared to hierNet, 700x speed-up compared to SPAMS, and 690x speed-up compared to glinternet (e.g., on the Amazon dataset glinternet cannot terminate in a week). We note that She et al. (2018) recently proposed an algorithm for a problem similar to ours, but do not provide a public toolkit—the largest problem reported in their paper is for $p = 10^3$: In the best case, their method takes ~ 51 seconds per solution, whereas hierScale takes 0.2 seconds.

Variable Selection: We compare the False Discovery Rate (FDR) at different sparsity levels. We generate synthetic data with $n = 100, p = 200$, and $\|\beta^0\|_0 = 10$ and report the FDR averaged over 100 datasets, in Figure 1. Under hierarchical truth, all the methods perform roughly similarly. However, under anti-hierarchy or main-only truth, our method can perform significantly better.

Prediction Tasks: We now compare the prediction performance of hierScale with the competing methods on both synthetic and real datasets.

Synthetic data: We use the same dataset as for variable selection (above). We tune on a separate validation set and report the prediction MSE on the testing data (training, validation, and test sets are of the same size). For hierNet and glinternet, we tune over 50 parameter values. For hierScale we use 50 λ_1-values and $\lambda_2 \in \{\lambda_1, 2\lambda_1\}$. For XGBoost we use 50 tree-sizes (between 20 and 1000) and learning rates $\in \{0.01, 0.1\}$. Results across 20 runs (for Main Only Truth) are in Figure 2: Our method shows significant improvements in prediction accuracy. The results for hierarchical and anti-hierarchical truth are in the supplementary, with results across methods being comparable.

Real data: We consider the Riboflavin dataset ($p = 4088$, $n = 71$) (Bühlmann et al., 2014) for predicting Vitamin B_2 production from gene expression levels. We train on 50 randomly chosen samples and compute the test RMSE on the remaining 21 samples. To reduce the variability, we repeat this training/testing procedure 30 times and report the average RMSE. We plot the RMSE versus the sparsity level in Figure 2. For hierScale we vary $\lambda_2 \in \{\lambda_1, 10\lambda_1, 100\lambda_1\}$, and for XGBoost we vary the learning rate $\in \{0.01, 0.1, 1\}$. Figure 2 reports the best RMSE (across the different λ_2's for hierScale the learning rates for XGBoost). We see that the SH methods (hierScale and glinternet) can be more effective than boosting at low/moderate sparsity levels (note $n = 71$ is small), which is desirable for interpretable learning. On this dataset, hierScale

[5] Available at `http://www.coepra.org`

Table 1: Average time (s) for obtaining a solution in the regularization path. The symbols * and ** indicate that the toolkit does not terminate in 3 days and 1 week, respectively. The dash (-) indicates a crash due to memory issues. The dot indicates that the data matrix could not fit in memory.

Toolkit	Synthetic datasets						Real datasets			
	p=500	1000	2000	5000	15000	50000	Ribo. (p=4088)	Coepra (p=5786)	Amazon-1 (p=5000)	Amazon-2 (p=10160)
hierScale	0.1	0.2	0.5	3.2	26.6	730	2.2	3.3	1.7	8.7
glinternet	0.3	1	3.8	24.4	226.9	*	2.7	7.8	*	**
hierNet	490	-	-	-	-	-	-	-	-	-
SPAMS	17.5	67.9	351.9	.	.	.	1042.3	-	.	.

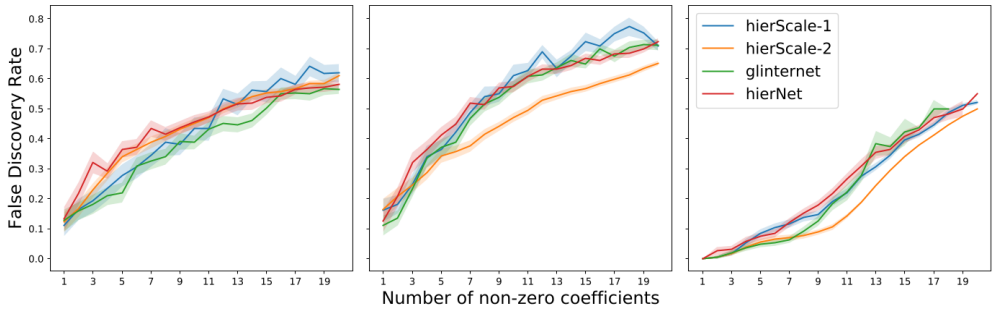

Figure 1: False Discovery Rate (FDR) for different methods, under 3 settings (I) Hierarchical Truth, (II) Anti-Hierarchical Truth and (III) Main-Only Truth (left to right). hierScale-1 and hierScale-2 refer to our method with $\lambda_2 = \lambda_1$ and $\lambda_2 = 2\lambda_1$, respectively. The shaded regions correspond to the standard error.

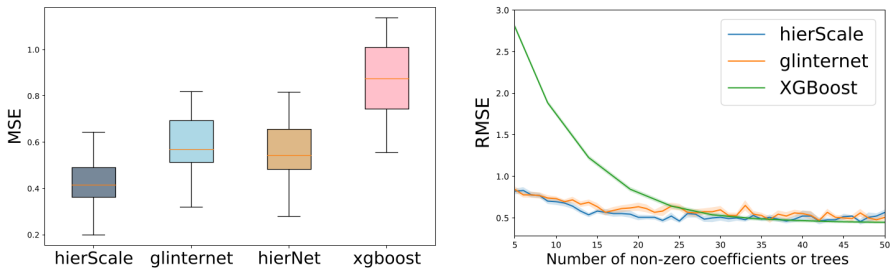

Figure 2: **Left**: Test MSE on the synthetic data (Main-Only truth). **Right**: Test RMSE on the Riboflavin dataset ($n = 50$, $p = 4088$). XGBoost is limited to depth 2. The shaded regions represent the standard error.

shows marginal improvement over glinternet. We note that Lim and Hastie (2015)'s experiments also indicate that SH methods can be more effective than boosted trees in terms of FDR.

5 Conclusion and Future Work

We studied the problem of learning sparse interactions under the strong hierarchy constraint. We introduced a transparent convex relaxation for the problem and developed a scalable algorithm based on proximal gradient descent. Our algorithm employs new screening rules for decomposing the proximal problem along with a specialized active-set method and gradient screening for mitigating costly gradient computations. Experiments on real and synthetic data show that our method achieves significant speed-ups over the state of the art and more robust variable selection.

There are several promising directions for future work. For instance, our proximal and gradient screening methods can be extended to general group-sparsity problems with overlaps between the group norms, which can potentially speed up the current solvers. Moreover, it would be interesting to establish bounds on the sizes of the connected components and the number of variables eliminated by gradient screening.

Acknowledgments

We would like to thank Jacob Bien for pointing us to relevant references on hierarchical variable selection. Our work was partially supported by ONR-N000141512342, ONR-N000141812298 (YIP), and NSF-IIS1718258.

References

Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373, 2013.

Francis R Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in neural information processing systems*, pages 105–112, 2009.

Francis R Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems*, pages 118–126, 2010.

Amir Beck. *First-Order Methods in Optimization*, volume 25. SIAM, 2017.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1): 183–202, 2009.

Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

Antoine Bonnefoy, Valentin Emiya, Liva Ralaivola, and Remi Gribonval. Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso. *IEEE Transactions on Signal Processing*, 63 (19):5121–5132, 2015.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Peter Bühlmann, Markus Kalisch, and Lukas Meier. High-dimensional statistics with a view toward applications in biology. *Annual Review of Statistics and Its Application*, 1:255–278, 2014.

Xi Chen, Qihang Lin, Seyoung Kim, Jaime G Carbonell, and Eric P Xing. Smoothing proximal gradient method for general structured sparse regression. *The Annals of Applied Statistics*, 6(2):719–752, 2012.

Hugh Chipman. Bayesian variable selection with related predictors. *Canadian Journal of Statistics*, 24 (1):17–36, 1996.

Nam Hee Choi, William Li, and Ji Zhu. Variable selection with the strong heredity constraint and its oracle property. *Journal of the American Statistical Association*, 105(489):354–364, 2010.

David R Cox. Interaction. *International Statistical Review/Revue Internationale de Statistique*, pages 1–24, 1984.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL `http://www.jstatsoft.org/v33/i01/`.

Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv preprint arXiv:1009.4219*, 2010.

Ning Hao and Hao Helen Zhang. Interaction screening for ultrahigh-dimensional data. *Journal of the American Statistical Association*, 109(507):1285–1301, 2014.

Hussein Hazimeh and Rahul Mazumder. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *arXiv preprint arXiv:1803.01454*, 2018.

Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12(Jul):2297–2334, 2011.

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, page 7. ACM, 2015.

Seunghak Lee and Eric P Xing. Screening rules for overlapping group lasso. *arXiv preprint arXiv:1410.6880*, 2014.

Michael Lim and Trevor Hastie. Learning interactions via hierarchical group-lasso regularization. *Journal of Computational and Graphical Statistics*, 24 (3):627–654, 2015.

Julien Mairal, Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Convex and network flow optimization for structured sparsity. *Journal of Machine Learning Research*, 12(Sep):2681–2720, 2011.

Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989.

Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of*

the Royal Statistical Society: Series B (Statistical Methodology), 70(1):53–71, 2008.

Marine Le Morvan and Jean-Philippe Vert. Whinter: A working set algorithm for high-dimensional sparse second order interaction models. *arXiv preprint arXiv:1802.05980*, 2018.

Kazuya Nakagawa, Shinya Suzumura, Masayuki Karasuyama, Koji Tsuda, and Ichiro Takeuchi. Safe pattern pruning: An efficient approach for predictive pattern mining. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 1785–1794. ACM, 2016.

Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24 (2):227–234, 1995.

Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap safe screening rules for sparse-group lasso. In *Advances in Neural Information Processing Systems*, pages 388–396, 2016.

Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap safe screening rules for sparsity enforcing penalties. *The Journal of Machine Learning Research*, 18(1):4671–4703, 2017.

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

Peter Radchenko and Gareth M James. Variable selection using adaptive nonlinear interaction structures in high dimensions. *Journal of the American Statistical Association*, 105(492):1541–1553, 2010.

Yiyuan She, Zhifeng Wang, and He Jiang. Group regularized estimation under structural hierarchy. *Journal of the American Statistical Association*, 113 (521):445–454, 2018.

Gian-Andrea Thanei, Nicolai Meinshausen, and Rajen D Shah. The xyz algorithm for fast interaction search in high-dimensional data. *The Journal of Machine Learning Research*, 19(1):1343–1384, 2018.

Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):245–266, 2012.

Jie Wang and Jieping Ye. Two-layer feature reduction for sparse-group lasso via decomposition of convex sets. In *Advances in Neural Information Processing Systems*, pages 2132–2140, 2014.

Jie Wang, Jiayu Zhou, Peter Wonka, and Jieping Ye. Lasso screening rules via dual polytope projection. In *Advances in Neural Information Processing Systems*, pages 1070–1078, 2013.

Jing Wu, Bernie Devlin, Steven Ringquist, Massimo Trucco, and Kathryn Roeder. Screen and clean: a tool for identifying interactions in genome-wide association studies. *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society*, 34(3):275–285, 2010.

Xiaohan Yan and Jacob Bien. Hierarchical sparse modeling: A choice of two group lasso formulations. *Statistical Science*, 32(4):531–560, 2017.