Screening Cut Generation for Sparse Ridge Regression

Haozhe Tan[1], Guanyi Wang[1]

[1]Department of Industrial Systems Engineering and Management,
National University of Singapore

May 5, 2025

Abstract

Sparse ridge regression is widely utilized in modern data analysis and machine learning. However, computing globally optimal solutions for sparse ridge regression is challenging, particularly when samples are arbitrarily given or generated under weak modeling assumptions. This paper proposes a novel cut-generation method, Screening Cut Generation (SCG), to eliminate non-optimal solutions for arbitrarily given samples. In contrast to recent safe variable screening approaches, SCG offers superior screening capability by identifying whether a specific $\{\pm 1\}$ combination of multiple features (binaries) lies in the set of optimal solutions. This identification is based on a convex relaxation solution rather than directly solving the original sparse ridge regression. Hence, the cuts generated by SCG can be applied in the pre-processing step of branch-and-bound and its variants to construct safe outer approximations of the optimal solution set. Numerical experiments are reported to validate the theoretical results and demonstrate the efficiency of SCG, particularly in hard real instances and synthetic instances with high dimensions, low ridge regularization parameters, or challenging modeling assumptions.

1 Introduction

Sparse ridge regression is a preponderantly used tool for feature selection while ensuring interpretability and generalization in various domains, including biological genetic analysis (Ogutu et al., 2012), sparsity-aware learning in compressed sensing (Theodoridis et al., 2014), healthcare diagnostics (Gramfort et al., 2013), and consumer defection prediction for cloud-based software (Kuswanto et al., 2015), to name but a few. The problem is formulated as follows:

$$v^* := \min_{\boldsymbol{\beta}\in\mathbb{R}^d} \ \mathcal{L}(\boldsymbol{\beta}) + \gamma\|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \le k \quad , \tag{$\mathcal{P}$}$$

where $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ represents a given convex loss, $\gamma > 0$ is a parameter for ℓ_2-norm ridge regularizer that promotes coefficient shrinkage, and the sparsity constraint $\|\boldsymbol{\beta}\|_0 \le k$ controls the size of support set $\mathtt{supp}(\boldsymbol{\beta}) := \{i \in [\![d]\!] \mid \beta_i \neq 0\}$ by at most k with $\|\boldsymbol{\beta}\|_0 := |\mathtt{supp}(\boldsymbol{\beta})| \le k$. In Mixed-Integer Programming (MIP) community, the sparsity constraint $\|\boldsymbol{\beta}\|_0 \le k$ can be reformulated by introducing binary variables $\{z_i\}_{i=1}^d \in \{0,1\}^d$ as $Z^k := \{(\boldsymbol{\beta}, \boldsymbol{z}) \in \mathbb{R}^d \times \{0,1\}^d \mid \mathbf{1}^\top \boldsymbol{z} \le k,\ \beta_i(1-z_i) = 0 \ \forall\, i \in [\![d]\!]\}$.

Given the NP-hardness of general sparse ridge regression (Chen et al., 2019) and its variants, a substantial body of research has focused on developing either inexact (approximate) iterative algorithms (i.e., local search, first-order based algorithm) or exact algorithms based on Branch-and-Bound (BnB) frameworks tailored to the original formulation ($\mathcal{P}$), as detailed in the literature review (see Section 2). A significant advancement in exact algorithm studies involves using (safe) variable screening approaches in the pre-processing step of BnB and its variants. Specifically, (safe) variable screening aims to determine whether a binary variable is guaranteed to be zero or one in an optimal solution of the original problem ($\mathcal{P}$), based on optimal solutions from convex relaxations rather than solving ($\mathcal{P}$) directly. Notably, for general MIPs (also ($\mathcal{P}$)), identifying a single binary variable roughly halves the number of feasible binary solutions. Such a reduction yields significant computational benefits, especially for enumeration-based optimization methods (e.g., BnB), since BnB usually scales exponentially with the number of binaries identified or eliminated. *Therefore, the effectiveness of variable screening approaches critically affects the computational efficiency on solving* ($\mathcal{P}$).

While promising results of safe screening techniques (Atamturk and Gomez, 2020; Deza and Atamturk, 2022) shed light on the efficiency and potential applicability of existing screening rules, challenges remain in many instances – particularly synthetic instances with relatively low ridge regularization parameter regime or signal-to-noise ratios, and real instances in machine learning community (see Section 4 for details). In these instances, a significant part of binary variables remain hard to identify using current screening rules in pre-processing step. These observations therefore lead to the following research question:

How and to what extent can convex relaxation solutions be leveraged to enhance screening approaches?

As a first step to address the above question, this paper introduces a novel cut-generation method, Screening Cut Generation (SCG). By leveraging fractional convex relaxation solutions, SCG safely eliminates non-optimal but feasible solutions for sparse ridge regression. Unlike existing approaches, SCG determines not only single binary variables but also specific $\{\pm 1\}$ combinations of multiple binaries is in the optimal solution set.

1.1 Main Contributions & Paper Organization

Our main contributions are given as follows: Section 2 provides a literature review followed by a brief preliminary on existing optimization algorithms and pre-processing techniques for sparse ridge regression. Section 3 presents our main contributions. Specifically, Section 3.1 introduces criteria of the proposed novel cut-generation method, Screening Cut Generation (SCG), which enhances existing variable screening approaches in the pre-processing step of BnB. Notably, we demonstrate that convex relaxation solutions can be more effectively utilized to generate screening cuts that eliminate not only single non-optimal binary variables but also specific $\{\pm 1\}$ combinations of multiple binaries that are not part of any optimal solutions, which generalizes existing screening approaches. Section 3.2 explores the theoretical properties of SCG by establishing necessary and sufficient conditions for certifying minimal (dominating) screening cuts. Additionally, Section 3.3 examines connections between SCG and the k-cardinality constrained binary knapsack polytope, providing an initial step toward investigating the closure of the proposed SCG cuts, which tackles the proposed research question on to what extent SCG cuts could outer approximate the optimal solution set. Section 3.4 introduces novel selection criteria for SCG cuts by quantifying their potential screening ability to further improve numerical efficiency, which is validated in Section 4. Numerical experiments are reported in Section 4 to demonstrate the advantages of the proposed SCG compared to existing baselines.

1.2 Notation

We use lowercase letters, e.g., x, boldface lowercase letters, e.g., $\boldsymbol{x}$, and boldface uppercase letters, e.g., $\boldsymbol{X}$, to represent scalars, vectors, and matrices, respectively. We use $\mathbf{1}$ to denote a vector of all ones. Given two integers a, b with $a \leq b$, we use $[\![a, b]\!]$ to denote the set of consecutive integers from a to b, i.e., $[\![a, b]\!] := \{a, \ldots, b\}$, and use $[\![b]\!] := \{1, \ldots, b\}$ as a shorthand notation for $[\![1, b]\!]$. Given a vector $\boldsymbol{x} \in \mathbb{R}^d$, for any $i \in [\![d]\!]$, we use $(\boldsymbol{x})_i$ to denote the i-th component with x_i a shorthand notation without further explanations, $(x)_{[k]}$ or $x_{[k]}$ to denote the k-th largest component, and $\underline{x} := x_{[d]}$ to be the smallest component (i.e., $\min_{i \in [\![d]\!]} x_i$) of $\boldsymbol{x}$. Given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, we use $\boldsymbol{x} \circ \boldsymbol{y} \in \mathbb{R}^d$ to denote the Hadamard product with $(\boldsymbol{x} \circ \boldsymbol{y})_i = x_i y_i$ for all $i \in [\![d]\!]$. We use $|\cdot|$ to denote the cardinality of a set and $\cdot \setminus \cdot$ to denote the set difference. Given an index subset $S \subseteq [\![d]\!]$, we use $\boldsymbol{x}_S \in \mathbb{R}^{|S|}$ as a subvector of $\boldsymbol{x}$ with its entries restricted on index subset S.

2 Preliminary & Literature Review

Sparse ridge regression and its optimization methods have been extensively explored over the past decades, and it is beyond the scope of this work to comprehensively review this vast literature. Instead, this section highlights the studies most relevant to our research. Ridge regression was originally introduced by Hoerl and Kennard (1970) to improve the robustness of regression coefficients $\boldsymbol{\beta}$. Subsequently, ℓ_1 regularization (LASSO) and combined ℓ_1-ℓ_2 regularization (elastic net) were developed to promote sparsity and robustness in regression coefficients. In recent years, there has been growing interest in solving sparse ridge regression directly, which can be, in general, separated into the following categories.

The first category focuses on designing approximation algorithms for sparse ridge regression under additional statistical assumptions. Examples include greedy and rounding algorithms (Xie and Deng, 2020), dual iterative hard thresholding (Yuan et al., 2020), and projected gradient descent (Liu et al., 2019). *In comparison, this paper aims to solve* ($\mathcal{P}$) *exactly to optimality without relying on any sample generation or modeling assumptions.*

The second category explores MIP-based approaches for sparse ridge regression. Recent advances (Bertsimas et al., 2016; Park and Klabjan, 2020; Bertsimas et al., 2020; Gómez and Prokopyev, 2021) have demonstrated promising results in solving sparse ridge regression problems with hundreds of variables. Additionally, the state-of-the-art method (Liu et al., 2024) introduces a customized branch-and-bound (BnB) framework that integrates fast lower-bound estimation and heuristic search, enabling the resolution of instances with thousands of variables. *In contrast, this paper aims to enhance the pre-processing step of BnB by screening binary variables based on convex relaxation solutions.*

The third category studies variable screening approaches used in the pre-processing step, which reduce the searching space of feasible binary solutions based on convex relaxation solutions. For LASSO-type problems, a series of studies (Ghaoui et al., 2011; Wang et al., 2013; Liu et al., 2014; Fercoq et al., 2015; Dantas et al., 2021) have introduced *safe* screening rules, while Tibshirani et al. (2011) proposes *strong* rules that are computationally efficient but may occasionally exclude optimal solutions. More recently, Atamturk and Gomez (2020); Deza and Atamturk (2022) have developed *exact safe* screening rules for sparse ridge regression. We highlight that the convex relaxation solutions used in our work, as well as in Atamturk and Gomez (2020); Deza and Atamturk (2022), are derived by solving the following perspective transformation of the ℓ_2-norm regularization term,

$$v_{\texttt{conic}} := \min_{\boldsymbol{t},\boldsymbol{\beta},\boldsymbol{z}\in[0,1]^d} \mathcal{L}(\boldsymbol{\beta}) + \gamma \mathbf{1}^\top \boldsymbol{t} \quad \text{s.t.} \quad \mathbf{1}^\top \boldsymbol{z} \le k, \quad t_i \ge \frac{\beta_i^2}{z_i}, \; \forall\, i \in [\![d]\!] \tag{1}$$

$$= \max_{\boldsymbol{p}} \min_{\boldsymbol{\beta},\boldsymbol{z}\in[0,1]^d} \mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{i=1}^{d} \left(p_i\beta_i - \frac{p_i^2 z_i}{4} \right) \quad \text{s.t.} \quad \mathbf{1}^\top \boldsymbol{z} \le k \tag{2}$$

where (1) is shown as a tight convex relaxation of ($\mathcal{P}$) (Xie and Deng, 2020) and has been widely adopted in the literature (Xie and Deng, 2020; Atamturk and Gomez, 2020; Liu et al., 2024). The reformulation (2) is obtained by taking the Fenchel conjugate on β_i^2/z_i, then exchanging the minimization and maximization due to strong duality. Using convex relaxation solutions from solving (2), Atamturk and Gomez (2020) show that:

Proposition 1 (Restatement, Safe Screening Rules from Atamturk and Gomez (2020)[1])**.** *Denote the optimal solution of (2) as $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{p}}$. Assume there is no tie in element-wise square of $\hat{\mathbf{p}}$, i.e. $\hat{\boldsymbol{w}} := \hat{\mathbf{p}} \circ \hat{\mathbf{p}}$. Let $v_{\texttt{ub}}$ be any upper bound of v^*, usually obtained by any feasible solutions of ($\mathcal{P}$). Then, for any $j \in [\![d]\!]$, the following safe screening rule*

$$z_j = \begin{cases} 0, & \textit{if} \quad \gamma\frac{\hat{w}_j}{4} \le \gamma\frac{\hat{w}_{[k+1]}}{4} \quad \textit{and} \quad \gamma\frac{\hat{w}_{[k]}}{4} - \gamma\frac{\hat{w}_j}{4} > \texttt{gap} \\ 1, & \textit{if} \quad \gamma\frac{\hat{w}_j}{4} \ge \gamma\frac{\hat{w}_{[k]}}{4} \quad \textit{and} \quad -\gamma\frac{\hat{w}_{[k+1]}}{4} + \gamma\frac{\hat{w}_j}{4} > \texttt{gap} \end{cases}$$

identifies whether a binary z_i could be zero or one in optimal solution sets with $\texttt{gap} := v_{\texttt{ub}} - v_{\texttt{conic}}$.

To be concise, in later discussions, we use *comparison criteria* to denote the first if condition, i.e., $\gamma\frac{\hat{w}_j}{4} \le \gamma\frac{\hat{w}_{[k+1]}}{4}$ or $\gamma\frac{\hat{w}_j}{4} \ge \gamma\frac{\hat{w}_{[k]}}{4}$, respectively; and *reduced-cost criteria* to denote the second if condition, i.e., $\gamma\frac{\hat{w}_{[k]}}{4} - \gamma\frac{\hat{w}_j}{4} > \texttt{gap}$ or $-\gamma\frac{\hat{w}_{[k+1]}}{4} + \gamma\frac{\hat{w}_j}{4} > \texttt{gap}$, respectively. We call the second if condition as *reduced-cost criteria* since its left-hand-side term plays a similar role as the reduced-cost (see discussions after Theorem 1 for details). As we can observe, the above safe-screening rules have a strong dependence on the ℓ_2-norm regularization parameter γ and the gap between upper and lower bounds of ($\mathcal{P}$). Notably, for sparse ridge regression, its upper bound $v_{\texttt{ub}}$ is obtained by any feasible solution of ($\mathcal{P}$) and its lower bound $v_{\texttt{conic}}$ is computed from solving the conic relaxation (1), the corresponding $\texttt{gap}$ is therefore lower bounded by the integrality gap between original ($\mathcal{P}$) and corresponding relaxation (1).

3 Main Results

3.1 Screening Cut Generation Rule

Recall screening rules in Proposition 1, the j-th binary variable z_j is identified if both comparison and reduced-cost criteria are satisfied. Such a screening rule will be weakened/violated under the

[1]We realized that a recent work "*Logic Rules and Chordal Graphs for Sparse Learning*" by A. Deza, A. Gómez, A. Atamtürk proposed a refined safe screening rule using techniques about chordal graphs. However, to the best of our knowledge, there is only one presentation slide for the International Symposium on Mathematical Programming (ISMP 2024) without any accessible preprint online. Additionally, based on their slides, we believe our main result (see Theorem 1 below) is distinct from their main contributions by taking multiple binaries into consideration, and proposes a SCG framework with both theoretical and practical advantages by selecting "strong" screening cuts in the pre-processing step.

scenarios when regularization parameter γ or the differences between $\hat{w}_j, \hat{w}_{[k]}, \hat{w}_{[k+1]}$ are small (see Section 4 for detailed scenario settings). To overcome such scenarios, Theorem 1 proposes the SCG rule that greatly enhances the reduced-cost criteria in Proposition 1 by taking into account a specific subset of binaries, other than a single binary variable.

Before presenting detailed results in Theorem 1, let us start with the definition of SCG-tuple as follows: Recall $\hat{\boldsymbol{\beta}}$ the optimal solution of (1), $\hat{\boldsymbol{p}}$ the Fenchel dual variable obtained from (2) given $\hat{\boldsymbol{\beta}}$. Let $v_{\texttt{ub}}$ and $v_{\texttt{conic}}$ be an upper and lower (optimal value from (2)) bounds of ($\mathcal{P}$), respectively.

Definition 1 (**SCG-tuple** (S, N, C))**.** *Given two disjoint index subsets $S, N \subseteq [\![d]\!]$ with $|S| \leq k$, let R be the remaining set of their union, i.e. $R := [\![d]\!] \setminus (S \cup N)$. For any index set $C \subseteq R$, we say (S, N, C) is a* SCG-tuple *with respect to $\hat{\boldsymbol{w}} = \hat{\boldsymbol{p}} \circ \hat{\boldsymbol{p}}$, if index set C is constructed as follows:*

$$C := \left\{ i \in R \mid \hat{w}_i \geq (\hat{w}_R)_{[c]} \right\}$$

with $c := \min\{k - |S|,\ |R|\}$. Specifically, when $c = 0$, we set $C := \emptyset$. In the later of this paper, for a SCG-tuple (S, N, C), we call index set S as the candidate set of support*, index set N as the* candidate set of non-support*, and index set C as the* complement set of support *to S, which is established based on S, N and $\hat{\boldsymbol{w}}$.*

Intuitively, given a SCG-tuple (S, N, C), set S represents the set of indices that are believed to become support, N represents the set of indices that are believed not to become support, while set C is fully determined by sets S and R as the "best" possible set based on $\hat{\boldsymbol{w}}$ that, combining with S, makes-up a k-sparse support set for ($\mathcal{P}$). We are poised to present the theorem for Screening Cut Generation (SCG).

Theorem 1 (**Screening Cuts Generation**)**.** *Suppose components in $\hat{\boldsymbol{w}} = \hat{\boldsymbol{p}} \circ \hat{\boldsymbol{p}}$ are distinct. Given a SCG-tuple (S, N, C) with respect to $\hat{\boldsymbol{w}}$, if the inequality of reduced-costs*

$$\sum_{i=1}^{k} \hat{w}_{[i]} - \sum_{i \in S} \hat{w}_i - \sum_{i \in C} \hat{w}_i > \frac{4}{\gamma} \texttt{gap} \tag{3}$$

holds, then the following screening cut

$$\sum_{i \in S} z_i + \sum_{i \in N} (1 - z_i) \leq |S| + |N| - 1$$

eliminates some non-optimal but remains all optimal solutions for ($\mathcal{P}$). Additionally, we call a SCG-tuple (S, N, C) a valid SCG-tuple *if (S, N, C) also ensures the above inequality (3).*

The proof of Theorem 1 is presented in Appendix A.1.One can relax the assumption "components in $\hat{\boldsymbol{w}} = \hat{\boldsymbol{p}} \circ \hat{\boldsymbol{p}}$ are distinct" by breaking ties lexicographically. To simplify our analysis, in the latter of this paper, *we always assume that there is no tie in $\hat{\boldsymbol{w}}$ without special instructions.*

To be concise, Theorem 1 provides a generalized screening rule that identifies whether a specific $\{\pm 1\}$ combination of binaries, i.e., $z_i = 1$ for $i \in S$ and $z_i = 0$ for $i \in N$, is in one of optimal solutions or not for the original sparse ridge regression ($\mathcal{P}$). The validness of the corresponding SCG-tuple (S, N, C) is determined by the inequality (3), which is a linear underestimate of Lagrangian relaxation constrained by candidate sets S and N given $\hat{\boldsymbol{p}}$,

$$\begin{array}{rl} \max_{\boldsymbol{p}} \min_{\boldsymbol{\beta},\boldsymbol{z}} & \mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{i=1}^d (p_i\beta_i - \frac{p_i^2}{4} z_i) \\ \text{s.t.} & \mathbf{1}^\top \boldsymbol{z} \le k,\ \boldsymbol{z} \in [0,1]^d \\ & z_i = 1\ \forall\ i \in S,\quad z_i = 0\ \forall\ i \in N \end{array} \ge v_{\texttt{conic}} + \frac{\gamma}{4}\left(\sum_{i=1}^k \hat{w}_{[i]} - \sum_{i\in S}\hat{w}_i - \sum_{i\in C}\hat{w}_i\right) > v_{\texttt{ub}}$$

$$\Leftrightarrow \text{ inequality (3)} \ .$$

That is to say, if inequality (3) holds, the corresponding choice of support subset S and non-support subset N will not be included in any of the optimal solutions of $(\mathcal{P})$. To further enhance the validity or effectiveness of inequality (3), in addition to increasing the term on the left-hand-side by taking multiple binaries into account, it is reasonable to reduce the right-hand-side term $4\texttt{gap}/\gamma$. Notably, Theorem 1 does not specify which upper bounds $v_{\texttt{ub}}$ of $(\mathcal{P})$ are used in SCG. Indeed, any upper bounds are allowed, and *a tighter relaxation or a method for improved primal feasible solutions will lead to a smaller gap, thereby enhancing the effectiveness of the proposed screening criterion.* However, in some real applications, the ridge regularization parameter γ lies in a relatively low regime, which weakens the effectiveness of Inequality (3). One possible direction (also discussed in Section 5) to handle low/zero γ regime is to study the following sequence of optimal solutions $\{\hat{\boldsymbol{\beta}}_i\}_{i\ge1}$ and their support sets obtained from a sequence of auxiliary sparse ridge regressions

$$\hat{\boldsymbol{\beta}}_i := \underset{\boldsymbol{\beta}}{\text{argmin}}\ \ \mathcal{L}(\boldsymbol{\beta}) + \gamma_i\|\boldsymbol{\beta}\|_2^2 \ \text{ s.t. }\ \|\boldsymbol{\beta}\|_0 \le k \quad ,$$

where $\{\gamma_i\}_{i\ge1}$ is a sequence of monotonically decreasing positive reals that tends to zero. Some preliminary numerical results (will be investigated as future directions, also mentioned in Section 5) indicate that in many cases, the supports of the solutions $\{\hat{\boldsymbol{\beta}}_i\}_{i\ge1}$ remain unchanged for sufficiently small γ_i. *This stability on support suggests that the screening cuts*[2] *computed from sparse ridge regression with small γ_i could extend to the general convex sparse regression setting.*

Remark 1. *Compared to existing screening rules (Proposition 1), since the left-hand-side term of inequality (3) involves multiple binaries other than a single binary, the reduced-cost criteria for SCG are more likely to ensure the validity than the reduced-cost criteria in Proposition 1. Moreover, let $\texttt{Top}_k(\hat{\boldsymbol{w}}) := \{i \in [\![d]\!] \mid \hat{w}_i \ge \hat{w}_{[k]}\}$ be the set of indices for the top-k components. By setting SCG-tuple as*

$$(S,N,C) = \begin{cases} (\{i\},\ \emptyset,\ \texttt{Top}_k(\hat{\boldsymbol{w}})\setminus\{i\}) & \textit{if} \quad i \in \texttt{Top}_k(\hat{\boldsymbol{w}}) \\ (\{i\},\ \emptyset,\ \texttt{Top}_{k-1}(\hat{\boldsymbol{w}})) & \textit{otherwise} \end{cases}$$
$$\textit{or} \begin{cases} (\emptyset,\ \{i\},\ \texttt{Top}_{k+1}(\hat{\boldsymbol{w}})\setminus\{i\}) & \textit{if} \quad i \in \texttt{Top}_k(\hat{\boldsymbol{w}}) \\ (\emptyset,\ \{i\},\ \texttt{Top}_k(\hat{\boldsymbol{w}})) & \textit{otherwise} \end{cases} .$$

The proposed SCG criteria reduce to existing criteria in Proposition 1. The proof of this reduction is given in Appendix A.2.

Additionally, a corollary (Corollary 1) of Theorem 1 is presented in Appendix A.3, which gives an equivalent algebraic (and more direct) approach to verify whether a given SCG-tuple is valid or not. The Corollary 1 will be used later in the proof of Proposition 3.

[2]These screening cuts may not be the safe screening cuts that only eliminates non-optimal solutions.

3.2 Minimal SCG-Tuple & Dominating Screening Cuts

This subsection introduces minimal SCG-tuple and corresponding dominating screening cuts.

Proposition 2. *Consider two valid SCG-tuples* $(S_1, N_1, C_1), (S_2, N_2, C_2)$*, if* $S_1 \subseteq S_2$ *and* $N_1 \subseteq N_2$*, then the screening cut generated by* (S_2, N_2, C_2) *is dominated by the screening cut generated by* (S_1, N_1, C_1)*.*

The proof of Proposition 2 is given in Appendix A.4. Note that the above dominating/dominated relationship is established based on the inclusion of index sets S and N. To avoid dominated cuts and improve computational efficiency in numerical experiments, we first define the following minimal SCG-tuple (from the perspective of set inclusion).

Definition 2. *We call a valid SCG-tuple* (S, N, C) *with respect to* $\hat{\boldsymbol{w}}$ *as a* minimal SCG-tuple*, if the following two conditions hold simultaneously:*

(1) $N = \emptyset$*, or* $(S, N \setminus \{i\}, C)$ *cannot form a valid SCG-tuple for any* $i \in N$*,*

(2) $S = \emptyset$*, or* $(S \setminus \{i\}, N, C \cup \{i\})$ *cannot form a valid SCG-tuple for any* $i \in S$*.*

Next, we give necessary and sufficient conditions that ensure a valid SCG-tuple is minimal.

Proposition 3. *Suppose components in* $\hat{\boldsymbol{w}} = \hat{\boldsymbol{p}} \circ \hat{\boldsymbol{p}}$ *are distinct. For a valid SCG-tuple* (S, N, C) *with respect to* $\hat{\boldsymbol{w}}$*, we have:*

- ***Case 1.*** *If* $|N| > d - k$*, then* (S, N, C) *is minimal if and only if* $S = \emptyset$*.*
- ***Case 2.*** *If* $|N| \leq d - k$ *and* $C = \emptyset$*, then* (S, N, C) *is minimal if and only if* $N = \emptyset$ *and* $\max_{i \in S} \ \hat{w}_i < \hat{w}_{[1]}$*.*
- ***Case 3.*** *If* $|N| \leq d - k$ *and* $C \neq \emptyset$*, then* (S, N, C) *is minimal if and only if (1) & (2) hold simultaneously,*

  (1) $N = \emptyset$*, or* $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$*;*

  (2) $S = \emptyset$*, or there exists some index* $j \notin (S \cup N \cup C)$ *such that* $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \ \hat{w}_i$*.*

The proof of Proposition 3 is given in Appendix A.5. The results proposed in Proposition 3 will be later used to construct screening cuts with relatively strong "potential screening ability" in Section 3.4. One quick example is that the valid SCG-tuple $(\{i\}, \emptyset, \texttt{Top}_{k-1}(\hat{\boldsymbol{w}}))$ guaranteed in Remark 1 with $i \notin \texttt{Top}_k(\hat{\boldsymbol{w}})$ is minimal. It can be shown by verifying: (1) $N = \emptyset$; (2) the index i^* with respect to k-th largest component of $\hat{\boldsymbol{w}}$, i.e., $\hat{w}_{i^*} = \hat{w}_{[k]}$, satisfies $i^* \notin \big(\texttt{Top}_{k-1}(\hat{\boldsymbol{w}}) \cup \{i\}\big)$ and $\hat{w}_{[k-1]} > \hat{w}_{i^*} > \hat{w}_i$.

3.3 Connections With k-Cardinality Binary Knapsack Polytope

Recall the inequality (3) proposed in Theorem 1 can be represented as: $\sum_{i \in S} \hat{w}_i + \sum_{i \in C} \hat{w}_i < -\frac{4}{\gamma}\texttt{gap} + \sum_{i=1}^{k} \hat{w}_{[i]} =: c(\hat{\boldsymbol{w}})$. Consider the following k-cardinality binary knapsack polytope $K(\hat{\boldsymbol{w}})$:

$$K(\hat{\boldsymbol{w}}) := \left\{ \boldsymbol{x} \in \{0,1\}^d \;\middle|\; \sum_{i=1}^{d} \hat{w}_i x_i < c(\hat{\boldsymbol{w}}), \ \|\boldsymbol{x}\|_0 \leq k \right\} \tag{4}$$

It is easy to conclude that: On one side, every valid SCG-tuple (S, N, C) (with respect to $\hat{\boldsymbol{w}}$) corresponds to one feasible point $\boldsymbol{x} \in K(\hat{\boldsymbol{w}})$ by letting $\mathtt{supp}(\boldsymbol{x}) = S \cup C$. On the other side, if $K(\hat{\boldsymbol{w}}) \neq \emptyset$ with $c(\hat{\boldsymbol{w}}) > 0$, every feasible point $\boldsymbol{x} \in K(\hat{\boldsymbol{w}})$ corresponds to at least $2^{|\mathtt{supp}(\boldsymbol{x})|}$ valid SCG-tuples (S, N, C) by setting $N = [\![d]\!] \setminus \mathtt{supp}(\boldsymbol{x})$ and disjoint subsets S, C such that $S \cup C = \mathtt{supp}(\boldsymbol{x})$.

To simplify the analysis in this subsection, *we assume that values of components in $\hat{\boldsymbol{w}}$ are in decreasing order, i.e., $\hat{w}_1 > \hat{w}_2 > \cdots > \hat{w}_d$*. Otherwise, we can define a new vector $\tilde{\boldsymbol{w}} := \boldsymbol{E}\hat{\boldsymbol{w}}$ with $\boldsymbol{E}$ a permutation matrix that re-orders every component of $\hat{\boldsymbol{w}}$ in a decreasing order. In the latter part of the section, we use $\tilde{\boldsymbol{w}}$ to denote the re-ordering of $\hat{\boldsymbol{w}}$ for simplicity.

Based on previous discussions, we note that every feasible $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$ leads to multiple valid SCG-tuples. However, as hinted from the Proposition 3, we are only interested in minimal SCG-tuples for dominating screening cuts. To figure out the connections between any feasible vector $\boldsymbol{x} \in K(\tilde{\boldsymbol{w}})$ and minimal SCG-tuples, we give the following definition.

Definition 3 (Consecutive index set). *A non-empty index set $A \subseteq [\![d]\!]$ is called a* consecutive index set *if, for every index $i \in A$, one of the following three cases holds: $\{i\} = A$, $i + 1 \in A$, or $i - 1 \in A$.*

Note that for a given index set I, there are usually multiple ways to partition I into several disjoint consecutive index sets. For example, one can partition index set $I = \{2, 3, 5, 6, 9\}$ into

$$I = \quad \{2,3\} \cup \{5,6\} \cup \{9\} \quad \text{or} \quad \{2\} \cup \{3\} \cup \{5,6\} \cup \{9\} \quad \text{or} \quad \{2\} \cup \{3\} \cup \{5\} \cup \{6\} \cup \{9\} \quad .$$

Later in the proof, we use *minimum consecutive partition* to denote the approach that partite a given index set into disjoint consecutive index sets with the minimum number of partitions. For example, the minimum consecutive partition of I is $\{2, 3\} \cup \{5, 6\} \cup \{9\}$. Moreover, based on our assumption that values of components in $\tilde{\boldsymbol{w}}$ are in decreasing order, the complementary index set can be defined as follows:

Definition 4 (Complementary index set). *Given a non-empty index subset $A \subseteq [\![d]\!]$, define* complementary index set *of A as $\mathtt{comp}(A) := \{i \in [\![d]\!] \mid i \notin A$ and $\exists\, j \in A$ such that $i < j\}$.*

Based on Defintion 3, for every $\boldsymbol{0} \neq \boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$, the minimum consecutive partition of $\mathtt{supp}(\boldsymbol{y})$ can be represented as

$$\mathtt{supp}(\boldsymbol{y}) = \cup_{j=1}^{m} A_j(\boldsymbol{y}) \ \text{ for some } m \leq k \text{ with } \ A_j(\boldsymbol{y}) := \left\{ i_1^{(j)}, \ldots, i_{|A_j|}^{(j)} \right\} \subseteq \mathtt{supp}(\boldsymbol{y})$$

for all $j = 1, \ldots, m$. Then, the minimum consecutive partition of $\mathtt{comp}(\mathtt{supp}(\boldsymbol{y}))$ can be partitioned into at most m disjoint blocks:

$$\mathtt{comp}(\mathtt{supp}(\boldsymbol{y})) = \cup_{j=1}^{m} B_j(\boldsymbol{y}) \ \text{ with } \ B_j(\boldsymbol{y}) := \left\{ i_{|A_{j-1}|}^{(j-1)} + 1, \ldots, i_1^{(j)} - 1 \right\}$$

for all $j = 1, \ldots, m$, where we let $i_{|A_0|}^{(0)} = 0$. Here, $B_1(\boldsymbol{y})$ might be empty. Now, we are ready to characterize the minimal SCG-tuple (S, N, C) based on any non-trivial $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$.

Proposition 4. *Suppose components in $\tilde{\boldsymbol{w}}$ are distinct. Given any non-trivial $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$. Let the minimum consecutive partitions of $\mathtt{supp}(\boldsymbol{y})$ and $\mathtt{comp}(\mathtt{supp}(\boldsymbol{y}))$ be*

$$\mathtt{supp}(\boldsymbol{y}) = \cup_{j=1}^{m} A_j(\boldsymbol{y}) \quad \textit{and} \quad \mathtt{comp}(\mathtt{supp}(\boldsymbol{y})) = \cup_{j=1}^{m} B_j(\boldsymbol{y})$$

with some $m \leq k$. Then, one can construct minimal SCG-tuple (S, N, C) with $S \cup C = \mathtt{supp}(\boldsymbol{y})$ as follows:

(1) If $|\mathtt{supp}(\boldsymbol{y})| < k$, then

$$(S, N, C) = (\emptyset,\ [\![d]\!] \setminus \mathtt{supp}(\boldsymbol{y}),\ \mathtt{supp}(\boldsymbol{y}))$$

is a minimal SCG-tuple.

(2) If $|\mathtt{supp}(\boldsymbol{y})| = k$, for $\ell = 1, \ldots, m-1$, we have the following SCG-tuple

$$(S, N, C) = \left(\cup_{j=\ell+1}^{m} A_j(\boldsymbol{y}),\ \cup_{j=1}^{\ell} B_j(\boldsymbol{y}),\ \cup_{j=1}^{\ell} A_j(\boldsymbol{y})\right)$$

is a minimal SCG-tuple. For $\ell = 0$ or $\ell = m$, define

$$(S_1, N_1, C_1) = (\mathtt{supp}(\boldsymbol{y}),\ \emptyset,\ \emptyset) \quad \text{or} \quad (S_2, N_2, C_2) = (\emptyset,\ \mathtt{comp}(\mathtt{supp}(\boldsymbol{y})),\ \mathtt{supp}(\boldsymbol{y}))\ ,$$

respectively. Then (S_2, N_2, C_2) is minimal. Moreover, if $B_1(\boldsymbol{y}) \neq \emptyset$, (S_1, N_1, C_1) is minimal as well.

The proof of Proposition 4 is presented in Appendix A.6. Based on the construction procedure given in Proposition 4, one can generate at most $m+1$ minimal SCG-tuples (and corresponding screening cuts) from any non-trivial point $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$.

3.4 Cuts With Strong Screening Ability

However, from numerical results reported in Section 4, it is relatively inefficient (in practice) to incorporate all possible screening cuts (obtained by minimal SCG-tuples) into the original sparse ridge regression. To enhance computational efficiency, this subsection selects screening cuts (obtained by minimal SCG-tuples) with relatively strong "*potential screening ability*," a criterion that quantifies the largest possible number of feasible solutions eliminated by corresponding screening cuts.

Definition 5. *Given a valid SCG-tuple (S, N, C), sorted vector $\tilde{\boldsymbol{w}}$ given in Section 3.3, and corresponding screening cut $\sum_{i \in S} z_i + \sum_{i \in N}(1 - z_i) \leq |S| + |N| - 1$, its* "potential screening ability" *is $\sum_{i=0}^{|C|} \binom{d-|S \cup N|}{i}$, which exactly denotes the number of feasible solutions that is eliminated by incorporating the above screening cuts into the original sparse ridge regression.*

The proof of bound given in Definition 5 is presented in Appendix A.7.

It is easy to observe that a smaller cardinality on set $S \cup N$ and a larger cardinality on set C will lead to a relatively greater "potential screening ability" for a given valid SCG-tuple (S, N, C). Note that, in practice, a set of screening cuts are selected to incorporate into the original sparse ridge regression. Thus, the additional number of non-optimal solutions removed by a newly added screening cut heavily depends on the rest of the selected screening cuts, and may be smaller than the "potential screening ability" provided above. We refer to the above number of eliminated non-optimal solutions as "*(marginal) screening ability*", which may be difficult to compute or estimate in practice.

High-level idea of SCG algorithm: To further enhance the computational efficiency for BnB, the proposed algorithm (Algorithm 1) makes a trade-off between the number of screening cuts selected and their corresponding "screening ability" based on the following three high-level criterion (decreasing in importance): (1) first, selecting cuts with relatively higher potential screening ability; (2) second, selecting cuts with relatively higher (marginal) screening ability; (3) third, if two cuts enjoy the same screening ability or potential screening ability, selecting the cut with greater left-hand-side value of the inequality in (4). Thus, this subsection focuses on selecting minimal SCG-tuple (dominating screening cuts) of the following two types.

Inclusive cuts. The first type of dominating screening cuts takes the form: $\sum_{i\in N} z_i \geq 1$. In short, such cuts indicate that any optimal solutions of ($\mathcal{P}$) contain at least one support index from N. According to Proposition 4, for any $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$, by setting $S = \emptyset$ and $C = \mathtt{supp}(\boldsymbol{y})$, the resulting SCG-tuple should be

$$(S, N, C) = \begin{cases} (\emptyset,\ [\![d]\!] \setminus \mathtt{supp}(\boldsymbol{y}),\ \mathtt{supp}(\boldsymbol{y})) & \text{if} \quad |\mathtt{supp}(\boldsymbol{y})| < k \\ (\emptyset,\ \mathtt{comp}(\mathtt{supp}(\boldsymbol{y})),\ \mathtt{supp}(\boldsymbol{y})) & \text{if} \quad |\mathtt{supp}(\boldsymbol{y})| = k \end{cases}$$

to ensure its validity and minimality. Note that when $|\mathtt{supp}(\boldsymbol{y})| < k$, the cardinality of set N in above SCG-tuple (S, N, C) is $d - |\mathtt{supp}(\boldsymbol{y})|$, which greatly impedes the "potential screening ability". Thus, inclusive cuts generation method only focuses on finding points in the following subset $\mathcal{O} := \{\boldsymbol{y} \in K(\tilde{\boldsymbol{w}}) \mid |\mathtt{supp}(\boldsymbol{y})| = k\}$ with minimal tuples $(S, N, C) = (\emptyset, \mathtt{comp}(\mathtt{supp}(\boldsymbol{y})), \mathtt{supp}(y))$. To simplify algorithm implementation (as will be mentioned in subSection 3.4.1), partition $\mathcal{O}$ into disjoint subsets based on the last index of $\mathtt{supp}(\boldsymbol{y})$ as $\mathcal{O} = \cup_{i=1}^{d} \mathcal{T}_i$ such that

$$\mathcal{T}_i := \{\boldsymbol{y} \in K(\tilde{\boldsymbol{w}}) \mid |\mathtt{supp}(\boldsymbol{y})| = k, \mathtt{supp}(\boldsymbol{y}) \subseteq [\![i]\!], i \in \mathtt{supp}(\boldsymbol{y})\} \tag{5}$$

a subset of $\mathcal{O}$ with i as the last index of $\mathtt{supp}(\boldsymbol{y})$ for all $i \in [\![d]\!]$. Given components of $\tilde{\boldsymbol{w}}$ are in decreasing order, we have $\mathcal{O} = \cup_{i=k+1}^{d} \mathcal{T}_i$ since $\mathcal{T}_i = \emptyset$ for all $i = 1, \ldots, k$. In particular, $\mathcal{T}_k = \emptyset$ holds because, if not, for any $\boldsymbol{y}' \in \mathcal{T}_k$ with $\mathtt{supp}(\boldsymbol{y}') = [\![k]\!]$, we have $\sum_{i=1}^{d} \tilde{w}_i y'_i = \sum_{i=1}^{k} \tilde{w}_i \geq c(\tilde{\boldsymbol{w}}) = -\frac{4}{\gamma}\mathtt{gap} + \sum_{i=1}^{k} \tilde{w}_i$, which leads to a contradiction. Then we present the Claim 1, which is used in Algorithm 1.

Claim 1. *Given $\mathcal{T}_i$ with $i \in [\![k+1, d]\!]$, $\mathcal{T}_i \neq \emptyset$ if and only if there exists some $\boldsymbol{v} \in K(\tilde{\boldsymbol{w}})$ such that $\mathtt{supp}(\boldsymbol{v}) = [\![i - (k-1), i]\!]$. Moreover, if $\mathcal{T}_i \neq \emptyset$, we have $\mathcal{T}_j \neq \emptyset$ for all $j = i+1, \ldots, d$.*

The proof of Claim 1 is given in Appendix A.8. In short, Claim 1 provides a necessary and sufficient condition to verify whether $\mathcal{T}_i$ is empty or not. Notably, finding the smallest index $s \in [\![k+1, d]\!]$ such that $\mathcal{T}_s \neq \emptyset$ is equivalent to finding all non-empty $\mathcal{T}_i$s, such a property is also used in Algorithm 1.

Exclusive cuts. The second type of screening cuts takes the form $\sum_{i\in S} z_i \leq |S| - 1$, which ensures that not all indices from S are support indices for any optimal solutions of ($\mathcal{P}$). Recall the minimal consecutive partitions of $\mathtt{supp}(\boldsymbol{y})$ and $\mathtt{comp}(\mathtt{supp}(\boldsymbol{y}))$ are $\mathtt{supp}(\boldsymbol{y}) = \cup_{j=1}^{m} A_j(\boldsymbol{y})$ and $\mathtt{comp}(\mathtt{supp}(\boldsymbol{y})) = \cup_{j=1}^{m} B_j(\boldsymbol{y})$. According to Proposition 4, for any $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$ with $|\mathtt{supp}(\boldsymbol{y})| = k$, the resulting SCG-tuple should be

$$(S, N, C) = \begin{cases} (\mathtt{supp}(\boldsymbol{y}),\ \emptyset,\ \emptyset) & \text{if} \quad B_1(\boldsymbol{y}) \neq \emptyset \\ (\cup_{j=2}^{m} A_j(\boldsymbol{y}),\ \emptyset,\ A_1(\boldsymbol{y})) & \text{if} \quad B_1(\boldsymbol{y}) = \emptyset \end{cases}$$

to ensure its validity and minimality. Note that when $B_1(\boldsymbol{y}) \neq \emptyset$, the SCG-tuple $(\mathtt{supp}(\boldsymbol{y}), \emptyset, \emptyset)$ only eliminates one single point based on the "potential screening ability". Therefore, exclusive cuts generation method focuses on finding solutions in the following subset

$$\mathcal{O}' := \{\boldsymbol{y} \in K(\tilde{\boldsymbol{w}}) \mid |\mathtt{supp}(\boldsymbol{y})| = k,\ B_1(\boldsymbol{y}) = \emptyset\} = \{\boldsymbol{y} \in K(\tilde{\boldsymbol{w}}) \mid |\mathtt{supp}(\boldsymbol{y})| = k,\ 1 \in A_1(\boldsymbol{y})\}$$

with minimal tuples $(\cup_{j=2}^{m} A_j(\boldsymbol{y}),\ \emptyset,\ A_1(\boldsymbol{y}))$. Similar to inclusive cuts generation, partition $\mathcal{O}'$ into disjoint subsets based on $A_1(\boldsymbol{y})$ as $\mathcal{O}' = \cup_{i=1}^{d} \mathcal{T}_i'$ such that

$$\mathcal{T}_i' := \{\boldsymbol{y} \in K(\tilde{\boldsymbol{w}}) \mid |\mathtt{supp}(\boldsymbol{y})| = k, A_1(\boldsymbol{y}) = [\![i]\!]\} \tag{6}$$

Given components of $\tilde{\boldsymbol{w}}$ are in decreasing order, we have $\mathcal{O}' = \cup_{i=1}^{k-1} \mathcal{T}_i'$ since $\mathcal{T}_i' = \emptyset$ for $i = k, \ldots, d$. The following claim provides a necessary and sufficient condition to certify whether each $\mathcal{T}_i'$ is empty or not.

Claim 2. *Given $\mathcal{T}_i'$ with $i \in [\![1, k-1]\!]$, we have $\mathcal{T}_i' \neq \emptyset$ if and only if there exists some $\boldsymbol{v} \in K(\tilde{\boldsymbol{w}})$ such that $\mathtt{supp}(\boldsymbol{v}) = [\![i]\!] \cup [\![d-(k-i-1), d]\!]$; Furthermore, if $\mathcal{T}_i' \neq \emptyset$, we have $\mathcal{T}_j' \neq \emptyset$, for all $j = 1, \ldots, i-1$.*

The proof of Claim 2 is given in A.8. This claim is then used in designing Algorithm 1.

In summary, the SCG algorithm prefers to select cuts with a relatively stronger potential screening ability than other criteria. For inclusive cut generation, given $\boldsymbol{y} \in \mathcal{T}_i \neq \emptyset$, its corresponding inclusive cut involves $|\mathtt{comp}(\mathtt{supp}(\boldsymbol{y}))| = i - k$ binary variables. Thus, SCG algorithm prefers to consider $\mathcal{T}_i$ with a smaller index i $(\geq k+1)$ to ensure a relatively stronger "potential screening ability". In contrast, for exclusive cut generation, given $\boldsymbol{y}' \in \mathcal{T}_i' \neq \emptyset$, its corresponding exclusive cut involves $|\cup_{j=2}^{m} A_j(\boldsymbol{y})| = k - |A_1(\boldsymbol{y})| = k - i$ binary variables. Therefore, SCG algorithm prefers to consider $\mathcal{T}_i'$ with a larger index i $(\leq k-1)$ to ensure a relatively stronger "potential screening ability".

3.4.1 SCG Algorithm Design

According to the discussions in Section 3.4 above, the SCG algorithm is presented in Algorithm 1. In particular, the proposed Algorithm 1 includes two hyper-parameters that control the maximum number of generated cuts $\sharp_{\mathtt{max}}$, and the maximum number of binary variables (potential screening ability) involved in each cut $\sharp_{\mathtt{len}}$. Additionally, we give brief discussions on two key steps of Algorithm 1 as follows:

STEP 1. Given a fixed $\sharp_{\mathtt{len}}$ that controls the number of binary variables involved, for inclusive or exclusive cut, it is sufficient to take indices i such that $i \leq k + \sharp_{\mathtt{len}}$ or $i \geq k - \sharp_{\mathtt{len}}$, respectively. That is to say, SCG algorithm only considers feasible points in $\mathcal{T}_i$s or $\mathcal{T}_i'$s with index i in the corresponding searching range,

$$i \in \mathtt{SR}_{\mathtt{inc}} := [\![k+1, \min\{k + \sharp_{\mathtt{len}}, d\}]\!] \quad \text{for inclusive cuts,} \tag{7}$$

$$i \in \mathtt{SR}_{\mathtt{exc}} := [\![\max\{1, k - \sharp_{\mathtt{len}}\}, k-1]\!] \quad \text{for exclusive cuts.} \tag{8}$$

Therefore, in Step 1, the starting search index s is initialized by

$$s_{\mathtt{inc}} := \operatorname{argmin}\left\{i \in \mathtt{SR}_{\mathtt{inc}} \;\middle|\; \exists\, \boldsymbol{v} \in K(\tilde{\boldsymbol{w}}) \text{ s.t. } \mathtt{supp}(\boldsymbol{v}) = [\![i-(k-1), i]\!]\right\} \tag{9}$$

$$s_{\mathtt{exc}} := \operatorname{argmax}\left\{i \in \mathtt{SR}_{\mathtt{exc}} \;\middle|\; \exists\, \boldsymbol{v} \in K(\tilde{\boldsymbol{w}}) \text{ s.t. } \mathtt{supp}(\boldsymbol{v}) = [\![i]\!] \cup [\![d-(k-i-1), d]\!]\right\} \tag{10}$$

for inclusive or exclusive cuts, respectively, which denotes the starting search index of $\mathcal{T}_\cdot$ or $\mathcal{T}_\cdot'$ that Algorithm 1 enumerates based on Claim 1 and Claim 2. Additionally, if $\mathcal{T}_i = \emptyset$, $\mathcal{T}_i' = \emptyset$ for all i

```
Algorithm 1: Inclusive/Exclusive Screening Cuts Generation (SCG)
  Input: Sorted $\tilde{\boldsymbol{w}}$, sparsity $k$, max number of cuts $\sharp_{\texttt{max}}$, max length of cuts $\sharp_{\texttt{len}}$.
  Set $\texttt{SR} = \texttt{SR}_{\texttt{inc}}$ or $\texttt{SR} = \texttt{SR}_{\texttt{exc}}$ from (7) or (8), set $\texttt{stopflag} = \texttt{false}$;
  Initialize set of generated cuts: $S_{\texttt{cuts}} = \emptyset$;
  Initialize starting search index $s$ based on (9) or (10)                    ▷STEP 1
  repeat
      Enumerate solutions in $\mathcal{T}_s$ or $\mathcal{T}'_s$ recursively using Algorithm 2;   ▷STEP 2
      for $\boldsymbol{y} \in \mathcal{T}_s$ or $\mathcal{T}'_s$ do
          Add inclusive/exclusive cut into $S_{\texttt{cuts}}$;
          Update $\sharp_{\texttt{max}} = \sharp_{\texttt{max}} - 1$;
          if $\sharp_{max} \le 0$ then
              set $\texttt{stopflag} = \texttt{true}$;
              break;
      Update search index $s$ based on discussion of STEP 2 in Section 3.4.1;
      If $s \in \texttt{SR}$, set $\texttt{stopflag} = \texttt{false}$; otherwise set $\texttt{stopflag} = \texttt{true}$;
  until $\texttt{stopflag} = \texttt{true}$;
  Output: Set of generated cuts $S_{\texttt{cuts}}$.
```

within the searching range $\texttt{SR}_{\texttt{inc}}, \texttt{SR}_{\texttt{exc}}$, there is no feasible solution in $K(\tilde{\boldsymbol{w}})$ that satisfies our needs, and the Algorithm 1 terminates.

STEP 2. If current search index s lies in the searching range $\texttt{SR}_{\texttt{inc}}$ or $\texttt{SR}_{\texttt{exc}}$, we run a recursive algorithm (Algorithm 2) to enumerate solutions in $\mathcal{T}_s$ or $\mathcal{T}'_s$. Note that enumerating all solutions in $\mathcal{T}_s$ or $\mathcal{T}'_s$ with $\sharp_{\texttt{max}} = +\infty$ leads to a worst-case computational complexity $O\big((k-1)^2 \cdot \binom{k+\sharp_{\texttt{len}}-1}{k-1}\big)$ or $O\big((\sharp_{\texttt{len}})^2 \cdot \binom{d}{\sharp_{\texttt{len}}}\big)$, respectively, which is exponential in k or $\sharp_{\texttt{len}}$. However, in practice, we always set the hyper-parameter $\sharp_{\texttt{max}} \le O(d)$ and $\sharp_{\texttt{len}} \in \{2, 3\}$ (see default hyper-parameter settings in Section 4.1), which makes STEP 2 very efficient (takes at most 2 seconds for synthetic datasets and 30 seconds for hard real instances, see Table 1 for details). Moreover, based on the design of our recursive algorithm, STEP 2 first enumerates solutions with a greater value on the left side of the inequality in (4), which follows the high-level idea mentioned before.

Then, Algorithm 1 updates its current searching index s by adding one for inclusive cuts generation and deleting one for exclusive cuts generation, respectively. Based on Claim 1 and Claim 2, the updated searching index s ensures $\mathcal{T}_s \neq \emptyset$. The stopping criterion of Algorithm 1 is set by: if one of the following two conditions is satisfied: (1) the number of cuts in $S_{\texttt{cuts}}$ exceeds $\sharp_{\texttt{max}}$, (2) search index s beyond its searching range $\texttt{SR}_{\texttt{inc}}$ or $\texttt{SR}_{\texttt{exc}}$.

Based on the above analysis, we claim the following result.

Claim 3. *By setting the maximal number of inclusive or exclusive cuts $\sharp_{\texttt{max}} = +\infty$, Algorithm 1 outputs all minimal inclusive cuts with $\sharp_{\texttt{len}} \le d - k$, or exclusive cuts with maximum length $\sharp_{\texttt{len}} \le k - 1$, respectively.*

The proof of Claim 3 is given in Appendix A.8, which ensures that, in theoretical, Algorithm 1 outputs all dominating cuts with deserved length (number of binaries involved).

4 Numerical Experiment

This section conducts numerical experiments on both synthetic and real datasets to demonstrate the efficiency of our proposed SCG method compared with the existing baseline, Safe Screening Rules (SSR, Atamturk and Gomez (2020)), in the pre-processing step.

4.1 Experimental Setup and Implementation

Loss function. Numerical experiments are conducted via the following sparse linear ridge regression model:

$$\min_{\|\boldsymbol{\beta}\|_0 \le k} \frac{1}{n}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \gamma\|\boldsymbol{\beta}\|_2^2 \tag{11}$$

where we use n as the number of samples, d as the dimension of each input sample and decision variable $\boldsymbol{\beta}$, coefficient matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is the input sample matrix with its i-th row $\boldsymbol{X}_{i,:}$ as the i-th input sample, coefficient vector $\boldsymbol{Y} \in \mathbb{R}^n$ is the output (response) vector with its i-th component $\boldsymbol{Y}_i$ as the response of $\boldsymbol{X}_{i,:}$, $\gamma > 0$ is a pre-determined parameter for ℓ_2-norm regularization, and k is the sparsity level. We fix the sparsity level $k = 10$ by default.

Performance measures. The numerical performance of the proposed SCG and SSR are measured based on the following metrics:

1. *Cut Characteristics*: We collect the characteristics or properties of (screening) cuts generated by SCG and SSR. In particular, for SSR, we compute the average number of binaries that can be screened/fixed. For SCG, we check the following two aspects: (i) the average number of generated inclusive and exclusive cuts, and (ii) the average number of binary variables involved in each screening cut.

2. *Total Running Time*: For SCG and SSR, we measure the total running time (in seconds) required to solve Problem (11) to same MIPGap, i.e. MIPGap $:= |v_p - v_d|/|v_p|$, where v_p is the primal objective bound and v_d is the dual objective bound. We use $t_{\mathtt{SCG}}$ to denote total running time of SCG, which is given by $t_{\mathtt{SCG}} := t_{\mathtt{SCG}}^{\mathtt{pre}} + t_{\mathtt{SCG}}^{\mathtt{sol}}$ with $t_{\mathtt{SCG}}^{\mathtt{pre}}$ its pre-processing time (i.e., time used to select screening cuts by SCG Algorithm 1) and $t_{\mathtt{SCG}}^{\mathtt{sol}}$ the time used to solving the original problem with added cuts by BnB. Similarly for SSR, its total running time is given by $t_{\mathtt{SSR}} := t_{\mathtt{SSR}}^{\mathtt{pre}} + t_{\mathtt{SSR}}^{\mathtt{sol}}$.

Testing procedures. For each dataset $(\boldsymbol{X}, \boldsymbol{Y})$ with regularization parameter γ and default sparsity level $k = 10$, the numerical experiments take the following procedures:

1. We first solve the corresponding Problem (11) using SSR via Gurobi. The Gurobi solver for SSR terminates if one of the following two stopping criteria (SC) is satisfied: (SSR-SC1) the time limit goes to 15 minutes; (SSR-SC2) the MIPGap is less than 1%. We use $t_{\mathtt{SSR}}^{\mathtt{sol}}$ and $\mathtt{Gap}_{\mathtt{SSR}}$ to denote the running time and MIPGap when Gurobi terminates for SSR method.

2. We then solve the corresponding Problem (11) using the proposed SCG method via Gurobi. The Gurobi solver for SCG terminates if one of the following two stopping criteria is satisfied: (SCG-SC1) the time limit goes to $t_{\mathtt{SSR}}^{\mathtt{sol}}$ in first bullet; (SCG-SC2) the MIPGap of SCG is less than $\max\{\mathtt{Gap}_{\mathtt{SSR}} - \varepsilon, 1\%\}$ for some small ε (we set $\varepsilon = 0.1\%$ by default). The (SCG-SC2) is designed to prevent scenarios in which SSR has been stuck at a particular MIP gap value.

Default hyper-parameter setting for SCG algorithm. To exceed the maximum number of inclusive cuts ($z_i = 1$ for some i) and exclusive cuts ($z_i = 0$ for some i) that SSR would select, the hyper-parameter $\sharp_{\texttt{max}}$ in proposed SCG Algorithm 1 is set as follows: For inclusive cuts generation, we set max number of inclusive cuts $\sharp_{\texttt{max}} = k = 10$ by default; and for exclusive cuts generation, we set max number of exclusive cuts $\sharp_{\texttt{max}} = d - k = d - 10$ by default. As mentioned above, a larger value on $\sharp_{\texttt{max}}$ gives a tighter outer approximation to the optimal set, while it may impede whole computational efficiency by adding too many constraints. Hence, $\sharp_{\texttt{max}}$ should not be too large in practice. To ensure relatively strong potential screening ability, the hyper-parameter $\sharp_{\texttt{len}}$ is set as 2 for inclusive cuts generation, and as 3 for exclusive cuts generation.

Hardware & Software. All experiments are conducted in Dell workstation Precision 7920 with a 3GHz 48Cores Intel Xeon CPU and 128GB 2934MHz DDR4 Memory. The proposed method and other baselines are solved using Gurobi 12.0.0 in Python 3.12.8.

4.2 Experimental Dataset

Synthetic datasets. We conducted numerical experiments on synthetic datasets with the following settings.

- **Data generation procedure.** Synthetic datasets $(\boldsymbol{X}, \boldsymbol{Y})$ are generated as follows (also used in Bertsimas et al. (2020)): Each sample $\boldsymbol{X}_{i,:}$ is i.i.d. drawn from a d-dimensional Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with covariance $\boldsymbol{\Sigma}_{ij} = (\rho^{|i-j|})_{i,j}$ for some fixed covariance parameter $\rho \in (0, 1)$ for all $i, j \in [\![d]\!]$. The response vector $\boldsymbol{Y}$ is therefore generated by $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}$ for some ground-truth $\boldsymbol{\beta}^*$, where $\boldsymbol{\beta}^* \in \{-1, 0, 1\}^d$ is randomly sampled with exactly k_0 non-zero entries, and every component of noise vector $\boldsymbol{\epsilon}$ is i.i.d. generated from a Gaussian distribution $\mathcal{N}(0, \frac{1}{n}\frac{\|\boldsymbol{X}\boldsymbol{\beta}^*\|_2^2}{\text{SNR}^2})$ with SNR denotes the signal-to-noise ratio.
- **Parameter settings.** We set input dimension $d \in \{100, 300, 500\}$, signal-to-noise ratio $\text{SNR} \in \{0.5, 3.5\}$, and $\gamma \in \{1.4, 1.6, 1.8, 2, 2.2\}$. We fix number of samples $n = 50$, covariance parameter $\rho = 0.5$, and number of non-zero entries $k_0 = 10$. For each parameter configuration $(d, \text{SNR}, \gamma, k_0 = 10, \rho = 0.5, n = 50)$, we generate 10 random independent synthetic datasets, i.e., $(\boldsymbol{X}^{(1)}, \boldsymbol{Y}^{(1)}), \ldots, (\boldsymbol{X}^{(10)}, \boldsymbol{Y}^{(10)})$.

Real datasets. We conducted numerical experiments on two real datasets: CNAE (856 features, 1080 samples) and UJIndoorLoc (520 features, 19937 samples) from the UCI Machine Learning Repository (Dua and Graff, 2017), which are also tested in Section 4.2 in Atamturk and Gomez (2020).

- **Parameter settings.** For each dataset, the sample-dependent parameter γ_0 for ℓ_2-norm regularization is set as $\gamma_0 := \frac{d}{k \max_j \|\boldsymbol{X}_{j,:}\|_2^2}$, which follows the setting used in Section 4.2 of Atamturk and Gomez (2020). The γ's used in our experiments are then ranging from $0.04/\gamma_0$ to $0.16/\gamma_0$ under a relatively lower γ regime (ranging from 0.04 to 0.16) ignoring the sample-dependent parameter γ_0. In particular, we set $\gamma \in \{0.04, 0.05, 0.055, 0.06, \ldots, 0.09\}$ for dataset CNAE and select $\gamma \in \{3000, 4000, \ldots, 14000\}$ for dataset UJIndoorLoc.
- **Reduced sample set.** We further test SSR and SCG on more challenging tasks (in statistics) with reduced sample sets. In particular, we pick the first 200 samples as the reduced sample

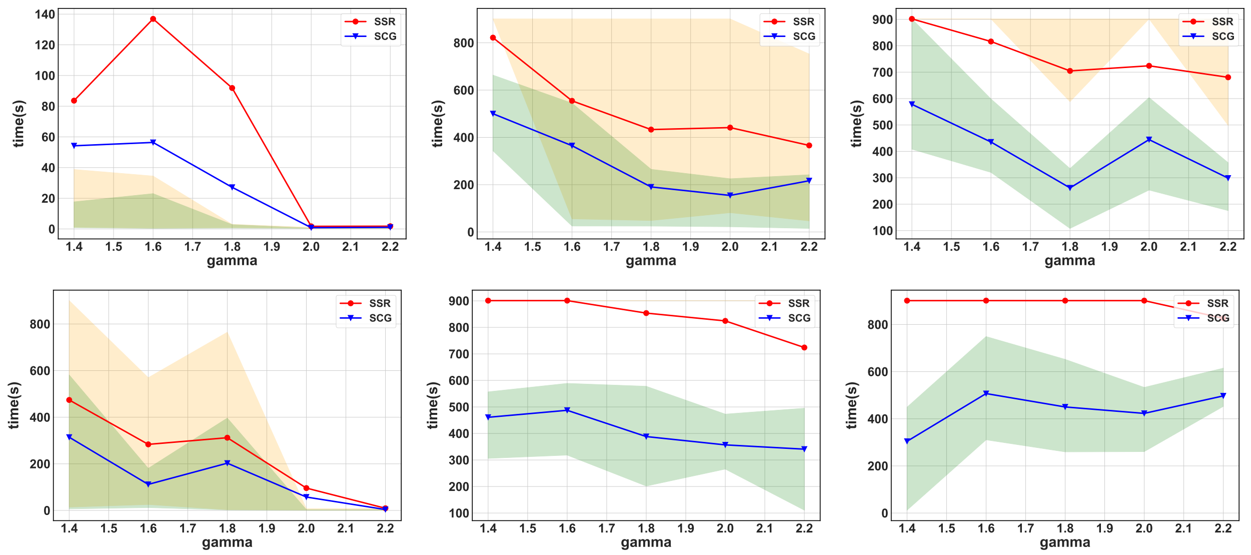

Figure 1. Total running time for synthetic datasets. The parameter configurations are specified in Section 4.2. The first and second row corresponds to SNR = 3.5 and SNR = 0.5, respectively. The three columns, from left to right, are set as $d = 100$, $d = 300$, $d = 500$, respectively. Each dot represents the average value over 10 independent datasets. The shaded area represents the inter-quartile range (25th to 75th percentiles).

set for CNAE, and pick 300 samples (first 60 samples among five distinct classes) as the reduced sample set for UJIndoorLoc.

4.3 Discussions on Numerical Results

4.3.1 Factors That Influence SCG

We start with figuring out which factors may greatly influence the effectiveness of selecting screening cuts, i.e., (i) *number of screening cuts selected* and (ii) *potential screening ability with respect to a selected screening cut* by SCG. Due to the space limit, the cut characteristic tables are provided in Appendix B. Numerical results reported in Table 2, 3, 4, 5, 6, 7, 8, 9 show that:

1. γ value has a primary impact on the effectiveness of selecting screening cuts. In particular, as γ decreases, the number of generated inclusive and exclusive cuts reduces for both SCG and SSR (see Table 2, 3, 4, 5, 6, 7), which validates the theoretical result presented in Theorem 1. Additionally, we would like to point out that, in low γ regime, *with sufficient samples, SCG can still generate strong screening cuts, while SSR cannot.* See Table 8, 9 for details.

2. Other factors, such as lower SNR and higher dimension d, have limited impacts on the effectiveness of selecting inclusive/exclusive cuts, compared with γ value. As SNR decreases (see Table 2, 3, and Table 4, 5, and Table 6, 7) or dimension increases (see Table 2, 4, 6, and Table 3, 5, 7), the number of selected inclusive or exclusive cuts reduces for both SSR and SCG, and the length (number of binaries involved) of selected screening cuts increases for SCG.

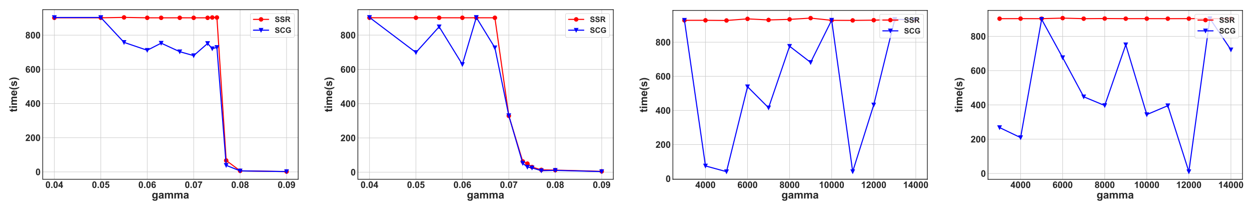

Figure 2. Total running time for real datasets. The parameter configurations are specified in Section 4.2. The four plots, arranged from left to right, represent the total running time under different γ values with: whole CNAE dataset, reduced CNAE dataset with 200 samples, whole UJIndoorLoc dataset, reduced UJIndoorLoc dataset with 300 samples.

In summary, SCG ensures stronger effectiveness in selecting inclusive or exclusive cuts than SSR, particularly in hard real datasets or synthetic datasets with challenging parameter configurations, as described above.

4.3.2 Running Time Improved By SCG

We then compare the total running time between the proposed SCG algorithm and the existing SSR.

Synthetic dataset. Figure 1 shows that, as the dimension increases, the cuts selected by SCG lead to better numerical performances on solving (11) than the cuts screened by SSR under different choices of SNR values.

In particular, the relative time gaps between SCG and SSR (defined as $\texttt{RTgap} := \frac{t_{\texttt{SSR}} - t_{\texttt{SCG}}}{t_{\texttt{SSR}}}$) keep increasing as dimension increases, i.e., $\texttt{RTgap}$ is about $10\% \sim 20\%$ in $d = 100$, $20\% \sim 30\%$ in $d = 300$, and roughly 40% in $d = 500$.

Real-world dataset. Numerical results for CNAE (first two plots in Figure 2) show that the running time of SSR and SCG heavily depends on the choice of γ. Specifically, as γ decreases, the running time enjoys a "*three phase*" trend, which can be explained as follows:

1. **Strong SCG v.s. Strong SSR.** When γ starts with a relatively large value, SSR could screen a significant part of binary variables, leading to comparable numerical performances in contrast with SCG.

2. **Strong SCG v.s. Weak SSR.** When γ decreases, we enter a parameter regime of γ where SSR's screening ability is significantly weakened, while SCG maintains a good screening ability by generating effective inclusive and exclusive cuts in the pre-processing step of BnB.

3. **Weak SCG v.s. Weak SSR.** When γ decreases to sufficiently small values, both methods' capabilities to generate minimal screening cuts become limited. Thus, BnB reduces to solve the original problem without any screening cuts generated in the pre-processing step.

Numerical results for UJIndoorLoc (last two plots in Figure 2) show that the gaps between running times of SCG and SSR are more significant but unstable, compared with gaps for CNAE. Such unstable numerical performances may stem from the ill condition number of the raw UJIndoorLoc dataset.

5 Conclusion & Future Directions

In conclusion, this paper introduces a novel cut-generating method, Screening Cut Generation (SCG), to improve and enhance existing variable screening approaches used in the pre-processing step of BnB and its variants. We show that convex relaxation solutions can be more effectively leveraged to generate screening cuts that eliminate both single non-optimal binary variables and specific $\{\pm 1\}$ combinations of multiple binaries that are not part of any optimal solutions. Furthermore, we establish necessary and sufficient conditions for certifying minimal (dominating) screening cuts. Combined with the cut-selection criteria based on potential screening ability, these theoretical insights form the foundation for our screening cut generation algorithm. Numerical experiments in Section 4 further validate the efficiency and effectiveness of the proposed method. We close with some potential SCG extensions and research questions for future investigation. The first is to study the closure of minimal screening cuts generated by every feasible point in the k-cardinality binary knapsack polytope (4). Second, it is natural to extend existing screening approaches to constrained sparse ridge regression. Lastly, motivated by some preliminary numerical results, whether one can generate screening cuts that are valid under a low γ regime case based on "stabilized" supports for sufficiently small γ_i as discussed after Theorem 1 remains an open and compelling question.

Acknowledgments

Haozhe Tan and Guanyi Wang were supported by the Ministry of Education, Singapore, under the Academic Research Fund (AcRF) Tier-1 grant (A-8000607-00-00) 22-5539-A0001.

References

A. Atamturk and A. Gomez. Safe screening rules for l0-regression from perspective relaxations. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 421–430. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/atamturk20a.html.

D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813 – 852, 2016. doi: 10.1214/15-AOS1388. URL https://doi.org/10.1214/15-AOS1388.

D. Bertsimas, J. Pauphilet, and B. Van Parys. Sparse regression: Scalable algorithms and empirical performance. *Statistical Science*, 35(4), Nov. 2020. ISSN 0883-4237. doi: 10.1214/19-sts701. URL http://dx.doi.org/10.1214/19-STS701.

Y. Chen, Y. Ye, and M. Wang. Approximation hardness for a class of sparse optimization problems. *Journal of Machine Learning Research*, 20(38):1–27, 2019.

C. F. Dantas, E. Soubies, and C. Févotte. Expanding boundaries of gap safe screening. *Journal of Machine Learning Research*, 22(236):1–57, 2021. URL http://jmlr.org/papers/v22/21-0179.html.

A. Deza and A. Atamturk. Safe screening for logistic regression with ℓ_0-ℓ_2 regularization, 2022. URL https://arxiv.org/abs/2202.00467.

D. Dua and C. Graff. Uci machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

O. Fercoq, A. Gramfort, and J. Salmon. Mind the duality gap: safer rules for the lasso. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 333–342, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/fercoq15.html.

L. E. Ghaoui, V. Viallon, and T. Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems, 2011. URL https://arxiv.org/abs/1009.4219.

A. Gómez and O. A. Prokopyev. A mixed-integer fractional optimization approach to best subset selection. *INFORMS J. Comput.*, (ijoc.2020.1031), Mar. 2021.

A. Gramfort, D. Strohmeier, J. Haueisen, M. S. Hämäläinen, and M. Kowalski. Time-frequency mixed-norm estimates: Sparse m/eeg imaging with non-stationary source activations. *NeuroImage*, 70:410–422, 2013.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

H. Kuswanto, A. Asfihani, Y. Sarumaha, and H. Ohwada. Logistic regression ensemble for predicting customer defection with very large sample size. *Procedia Computer Science*, 72:86–93, 2015.

J. Liu, Z. Zhao, J. Wang, and J. Ye. Safe screening with variational inequalities and its application to lasso. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 289–297, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/liuc14.html.

J. Liu, S. Rosen, C. Zhong, and C. Rudin. Okridge: Scalable optimal k-sparse ridge regression, 2024. URL https://arxiv.org/abs/2304.06686.

L. Liu, Y. Shen, T. Li, and C. Caramanis. High dimensional robust sparse regression, 2019. URL https://arxiv.org/abs/1805.11643.

J. O. Ogutu, T. Schulz-Streeck, and H.-P. Piepho. Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. In *BMC proceedings*, volume 6, pages 1–6. Springer, 2012.

Y. W. Park and D. Klabjan. Subset selection for multiple linear regression via optimization. *J. Glob. Optim.*, 77(3):543–574, July 2020.

S. Theodoridis, Y. Kopsinis, and K. Slavakis. Sparsity-aware learning and compressed sensing: An overview. *Academic Press Library in Signal Processing*, 1:1271–1377, 2014.

R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 74(2):245–266, 11 2011. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2011.01004.x. URL https://doi.org/10.1111/j.1467-9868.2011.01004.x.

J. Wang, J. Zhou, P. Wonka, and J. Ye. Lasso screening rules via dual polytope projection. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/8b16ebc056e613024c057be590b542eb-Paper.pdf.

W. Xie and X. Deng. Scalable algorithms for the sparse ridge regression. *SIAM Journal on Optimization*, 30(4):3359–3386, 2020. doi: 10.1137/19M1245414. URL https://doi.org/10.1137/19M1245414.

X.-T. Yuan, B. Liu, L. Wang, Q. Liu, and D. N. Metaxas. Dual iterative hard thresholding. *Journal of Machine Learning Research*, 21(152):1–50, 2020. URL http://jmlr.org/papers/v21/18-487.html.

A Proofs in Section 3

A.1 Proof of Theorem 1

Theorem 1 (Screening Cuts Generation). *Suppose components in $\hat{\boldsymbol{w}} = \hat{\boldsymbol{p}} \circ \hat{\boldsymbol{p}}$ are distinct. Given a SCG-tuple (S, N, C) with respect to $\hat{\boldsymbol{w}}$, if the inequality of reduced-costs*

$$\sum_{i=1}^{k} \hat{w}_{[i]} - \sum_{i \in S} \hat{w}_i - \sum_{i \in C} \hat{w}_i > \frac{4}{\gamma}\texttt{gap} \tag{3}$$

holds, then the following screening cut

$$\sum_{i \in S} z_i + \sum_{i \in N} (1 - z_i) \leq |S| + |N| - 1$$

eliminates some non-optimal but remains all optimal solutions for $(\mathcal{P})$. Additionally, we call a SCG-tuple (S, N, C) a valid SCG-tuple *if (S, N, C) also ensures the above inequality (3).*

Proof. Adding constraints $z_i = 1$ for all $i \in S$ and $z_j = 0$ for all $j \in N$ to sparse ridge regression $(\mathcal{P})$ gives

$$\begin{aligned} v_{S,N} := \min_{\boldsymbol{\beta}, \boldsymbol{z}} \quad & \mathcal{L}(\boldsymbol{\beta}) + \gamma \|\boldsymbol{\beta}\|_2^2 \\ \text{s.t.} \quad & (1 - z_i)\beta_i = 0 \quad \forall i \in [\![d]\!] \\ & \mathbf{1}^\top \boldsymbol{z} \leq k, \ \ \boldsymbol{z} \in \{0, 1\}^d \\ & z_i = 1 \ \forall \ i \in S, \quad z_i = 0 \ \forall \ i \in N \end{aligned}$$

and its naive conic relaxation becomes:

$$\begin{aligned} \tilde{v}_{S,N} := \max_{\boldsymbol{p}} \min_{\boldsymbol{\beta}, \boldsymbol{z}} \quad & \mathcal{L}(\boldsymbol{\beta}) + \gamma \textstyle\sum_{j=1}^{d} (p_j \beta_j - \frac{p_j^2}{4} z_j) \\ \text{s.t.} \quad & \boldsymbol{z} \in \mathcal{Z}_{S,N} := \left\{ \boldsymbol{z} \in [0,1]^d \mid \mathbf{1}^\top \boldsymbol{z} \leq k, \ z_i = 1 \ \forall \ i \in S, \quad z_i = 0 \ \forall \ i \in N \right\} \end{aligned}$$

A lower bound of $\tilde{v}_{S,N}$ can be obtained by plugging in $\hat{\boldsymbol{p}}$ (obtained from (2)) to the outer maximization:

$$v_{\texttt{res}} := \min_{\boldsymbol{\beta}, \ \boldsymbol{z} \in \mathcal{Z}_{S,N}} \quad \mathcal{L}(\boldsymbol{\beta}) + \gamma \textstyle\sum_{j=1}^{d} (\hat{p}_j \beta_j - \frac{\hat{p}_j^2}{4} z_j) \quad . \tag{12}$$

Note the objective function of (12) can be separated into two parts:

$$\mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{j=1}^{d} (\hat{p}_j \beta_j - \frac{\hat{p}_j^2}{4} z_j) = \underbrace{\left(\mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{j=1}^{d} \hat{p}_j \beta_j \right)}_{=:f_1(\boldsymbol{\beta})} - \underbrace{\left(\gamma \sum_{j=1}^{d} \frac{\hat{p}_j^2}{4} z_j \right)}_{=:f_2(\boldsymbol{z})} \quad ,$$

where, given $\hat{\boldsymbol{p}}$, the first part $f_1(\boldsymbol{\beta})$ only depends on $\boldsymbol{\beta}$ and the second part $f_2(\boldsymbol{z})$ only depends on $\boldsymbol{z}$. Thus, minimizing (12) is equivalent to minimizing the first part $f_1(\boldsymbol{\beta})$ and maximizing the second part $f_2(\boldsymbol{z})$ at the same time, i.e.,

$$v_{\texttt{res}} \ = \ \min_{\boldsymbol{\beta}} \ f_1(\boldsymbol{\beta}) \ - \ \max_{\boldsymbol{z} \in \mathcal{Z}_{S,N}} \ f_2(\boldsymbol{z})$$

To minimize the first part $f_1(\boldsymbol{\beta})$, it is easy to verify that $\hat{\boldsymbol{\beta}}$ obtained from (1) is also an optimal solution. To maximize linear objective function $f_2(\boldsymbol{z})$, we need to choose at most k largest values of $\frac{\gamma}{4}\hat{p}_j^2 = \frac{\gamma}{4}\hat{w}_j$ while maintaining the feasibility requirement, i.e., $\boldsymbol{z} \in \mathcal{Z}_{S,N}$. In particular, consider the following two cases:

1. Suppose $|N| > d-k$, then $\mathbf{1}^T\boldsymbol{z} \leq k$ is automatically satisfied. By letting $z_i = 1$ for all $i \in [\![d]\!] \setminus N$ that includes S as a subset, the resulting tuple $(S, N, [\![d]\!] \setminus (S \cup N))$ is a SCG-tuple with $S \cup C = [\![d]\!] \setminus N$ forms the support of optimal $\boldsymbol{z}$.
2. Suppose $|N| \leq d-k$, then the remaining index set $R := [\![d]\!] \setminus (S \cup N)$ includes at least $k - |S|$ elements. Thus, to maximize $f_2(\boldsymbol{z})$, the optimal $\boldsymbol{z}$ selects top-$(k-|S|)$ largest components in $\{\frac{\gamma}{4}\hat{w}_j\}_{j\in R}$. Still, the resulting (S, N, C) is a SCG-tuple with respect to $\hat{\boldsymbol{w}}$, and set $S \cup C$ forms the support of optimal $\boldsymbol{z}$.

Combining the above two cases ensures that the optimal $\boldsymbol{z}$ always takes $z_i = 1$ for $i \in S \cup C$. Then, we have

$$\begin{aligned} v_{S,N} \geq \tilde{v}_{S,N} \geq v_{\texttt{res}} &= \mathcal{L}(\hat{\boldsymbol{\beta}}) + \gamma \sum_{j=1}^{d} (\hat{p}_j \hat{\beta}_j) - \gamma \sum_{j \in (S\cup C)} \frac{1}{4}\hat{w}_j \\ &= \underbrace{\left[\mathcal{L}(\hat{\boldsymbol{\beta}}) + \gamma \sum_{j=1}^{d} (\hat{p}_j \hat{\beta}_j) - \gamma \sum_{j=1}^{k} \frac{1}{4}\hat{w}_{[j]} \right]}_{=v_{\texttt{conic}}} + \frac{\gamma}{4}\left[\sum_{i=1}^{k} \hat{w}_{[i]} - \sum_{i\in S}\hat{w}_i - \sum_{j \in N} \hat{w}_j \right] \end{aligned}$$

If inequality (3) holds, we have $\frac{\gamma}{4}[\sum_{i=1}^{k}\hat{w}_{[i]} - \sum_{i\in S}\hat{w}_i - \sum_{j\in N}\hat{w}_j] > v_{\texttt{ub}} - v_{\texttt{conic}}$, which implies $v_{S,N} \geq v_{\texttt{res}} > v_{\texttt{ub}} \geq v^*$. That is to say, any feasible solution of $(\mathcal{P})$ with $z_i = 1$ for $i \in S$ and $z_j = 0$ for $j \in N$ cannot be optimal, and we can add screening cut $\sum_{i\in S} z_i + \sum_{j \in Z}(1-z_j) \leq |S| + |Z| - 1$ to remove all points that satisfy $\boldsymbol{z} \in \mathcal{Z}_{S,N}$.

Notice that if there exists some $i \in C$ with $\frac{\gamma}{4}\hat{w}_i = 0$, any $z_i \in [0,1]$ leads to an optimal solution for (12). Without loss of generality, we always choose $z_i = 1$ for our analysis. □

A.2 Proof of Remark 1

Proof. let $S = \{i\}$ for some $i \in [\![d]\!]$ and $N = \emptyset$. If $i \in \texttt{Top}_k(\hat{\boldsymbol{w}})$, we have $S \cup C = \texttt{Top}_k(\hat{\boldsymbol{w}})$. The inequality (3) is not satisfied with $0 > \frac{4}{\gamma}\texttt{gap}$ and no cut can be generated. If $i \notin \texttt{Top}_k(\hat{\boldsymbol{w}})$, and the inequality (3) is satisfied, which is equivalent to $\hat{w}_{[k]} - \hat{w}_i > \frac{4}{\gamma}\texttt{gap}$, then $(\{i\}, \emptyset, \texttt{Top}_{k-1}(\hat{\boldsymbol{w}}))$ is a valid tuple. The resulting cut is $z_i \leq 0$.

Similarly, let $S = \emptyset$ and $Z = \{j\}$ for some $j \in [\![d]\!]$. If $j \notin \texttt{Top}_k(\hat{\boldsymbol{w}})$, we have $S \cup C = \texttt{Top}_k(\hat{\boldsymbol{w}})$. The inequality (3) is not satisfied with $0 > \frac{4}{\gamma}\texttt{gap}$ and no cut can be generated. If $j \in \texttt{Top}_k(\hat{\boldsymbol{w}})$, and the inequality (3) is satisfied, which is equivalent to $\hat{w}_j - \hat{w}_{[k+1]} > \frac{4}{\gamma}\texttt{gap}$, then $(\emptyset, \{i\}, \texttt{Top}_{k+1}(\hat{\boldsymbol{w}}) \setminus \{j\})$ is a valid tuple. The resulting cut is $z_j \geq 1$. □

A.3 Proof of Corollary 1

Here, we propose a corollary of Theorem 1, which will be used in the proof of Proposition 3.

Corollary 1. *Suppose components in $\hat{\boldsymbol{w}} = \hat{\boldsymbol{p}} \circ \hat{\boldsymbol{p}}$ are distinct. Given a SCG-tuple (S, N, C), we say it is valid if the following three conditions hold simultaneously:*

(1) $S, N, C \subseteq [\![d]\!]$ are pair-wisely disjoint with $|C| = \min\{k - |S|, d - |N| - |S|\}$;

(2) If $C \neq \emptyset$, then for all i such that $\hat{w}_i \geq \underline{\hat{w}_C}$, we have $i \in (S \cup N \cup C)$;

(3) $\sum_{i=1}^{k} \hat{w}_{[i]} - \sum_{i\in S} \hat{w}_i - \sum_{j\in C} \hat{w}_j > \frac{4}{\gamma}\texttt{gap}$.

Proof. Denote R as the set of remaining indices, i.e. $R = [\![d]\!] \setminus (S \cup N)$.

$\Rightarrow$: Define $C := \{i \in R \mid \hat{w}_i \geq (\hat{w}_R)_{[c]}\}$, where $c := \min\{k - |S|, d - |N| - |S|\}$. If $|S| = k$, or $d = |N| + |S|$, then $C = \emptyset$ and we are done. For other cases, $c > 0$ and $C \neq \emptyset$ since $c \leq d - |N| - |S| = |R|$. Clearly $|C| = c$ and $\underline{\hat{w}_C} = (\hat{w}_R)_{[c]}$. We only need to show point (ii). Prove by contradiction. Suppose there exists some $j \notin (S \cup N \cup C)$ such that $\hat{w}_j \geq \underline{\hat{w}_C}$. First, we have $j \in R$. We also have $j \in C$ since $\hat{w}_j \geq \underline{\hat{w}_C} = (\hat{w}_R)_{[c]}$. Contradiction.

$\Leftarrow$: If $|S| = k$, or $d = |N| + |S|$, then $C = \emptyset$ and we are done. For other cases, we have $|C| > 0$. Define $\mathcal{D} := \{i \in R \mid \hat{w}_i \geq (\hat{w}_R)_{[|C|]}\}$. This is valid since $|C| \leq d - |N| - |S| = |R|$. Now we know $|\mathcal{D}| = |C|$. It is sufficient to prove that pick any $i' \in C$, $i' \in D$. Clearly $i' \in R$. Suppose $i' \notin \mathcal{D}$, we have $\hat{w}_{i'} < (\hat{w}_R)_{[|C|]}$. Since we know $|\mathcal{D}| = |C|$, the existence of i' implies that there exists some $j' \in \mathcal{D}$ but $j' \notin C$. Thus we have $\hat{w}_{j'} \geq (\hat{w}_R)_{[|C|]} > \hat{w}_{i'} \geq \underline{\hat{w}_C}$, and $j' \notin (S \cup N \cup C)$. Contradiction with the point (ii). Thus we have $C = \mathcal{D}$ and we are done. $\square$

A.4 Proof of Proposition 2

Proposition 2. *Consider two valid SCG-tuples $(S_1, N_1, C_1), (S_2, N_2, C_2)$, if $S_1 \subseteq S_2$ and $N_1 \subseteq N_2$, then the screening cut generated by (S_2, N_2, C_2) is dominated by the screening cut generated by (S_1, N_1, C_1).*

Proof. The screening cuts generated by $(S_1, N_1, C_1), (S_2, N_2, C_2)$ are

$$\sum_{i\in S_1} z_i + \sum_{j\in N_1}(1 - z_j) \leq |S_1| + |N_1| - 1 \quad \text{and} \quad \sum_{i\in S_2} z_i + \sum_{j\in N_2}(1 - z_j) \leq |S_2| + |N_2| - 1 \quad ,$$

respectively. Consider any binary vector $\tilde{\boldsymbol{z}} \in \{0,1\}^d$ that is feasible for the screening cuts generated by (S_1, N_1, C_1), to show domination, it is sufficient to have

$$\begin{aligned}
& \sum_{i\in S_2} \tilde{z}_i + \sum_{j\in N_2}(1 - \tilde{z}_j) \leq |S_2| + |N_2| - 1 \\
\Leftrightarrow \quad & \sum_{i\in S_2} \tilde{z}_i \leq |S_2| - 1 + \sum_{j\in N_2} \tilde{z}_j \\
\Leftrightarrow \quad & \sum_{i\in S_1} \tilde{z}_i + \sum_{i\in S_2\setminus S_1} \tilde{z}_i \leq (|S_1| - 1) + (|S_2| - |S_1|) + \sum_{j\in N_1} \tilde{z}_j + \sum_{j\in N_2\setminus N_1} \tilde{z}_j \\
\Leftrightarrow \quad & \sum_{i\in S_1} \tilde{z}_i + \sum_{i\in S_2\setminus S_1} \tilde{z}_i \leq \left(|S_1| - 1 + \sum_{j\in N_1} \tilde{z}_j\right) + (|S_2| - |S_1|) + \sum_{j\in N_2\setminus N_1} \tilde{z}_j \quad ,
\end{aligned}$$

where the last inequality holds due to our assumption on $\tilde{z}$, $\sum_{i\in S_2\setminus S_1} \tilde{z}_i \leq |S_2| - |S_1|$, and $\sum_{j\in N_2\setminus N_1} \tilde{z}_j \geq 0$. Therefore, the domination holds. $\square$

A.5 Proof of Proposition 3

Proposition 3. *Suppose components in $\hat{\boldsymbol{w}} = \hat{\boldsymbol{p}} \circ \hat{\boldsymbol{p}}$ are distinct. For a valid SCG-tuple (S, N, C) with respect to $\hat{\boldsymbol{w}}$, we have:*

- ***Case 1.*** *If $|N| > d - k$, then (S, N, C) is minimal if and only if $S = \emptyset$.*

- ***Case 2.*** *If $|N| \le d-k$ and $C = \emptyset$, then (S,N,C) is minimal if and only if $N = \emptyset$ and* $\max_{i \in S} \ \hat{w}_i < \hat{w}_{[1]}$.
- ***Case 3.*** *If $|N| \le d-k$ and $C \ne \emptyset$, then (S,N,C) is minimal if and only if (1) & (2) hold simultaneously,*
  - *(1) $N = \emptyset$, or $\underline{\hat{w}_N} \ge \underline{\hat{w}_C}$;*
  - *(2) $S = \emptyset$, or there exists some index $j \notin (S \cup N \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \ \hat{w}_i$.*

Proof. Before presenting the proof of Proposition 3, recall the following three conditions of a *valid SCG-tuple* (S,N,C) with respect to $\hat{\boldsymbol{w}}$:

(1) $S, N, C \subseteq [\![d]\!]$ are pair-wisely disjoint with $|C| = \min\{k - |S|, d - |N| - |S|\}$;

(2) If $C \ne \emptyset$, for all i with $\hat{w}_i \ge \underline{\hat{w}_C}$, we have $i \in (S \cup N \cup C)$;

(3) $\sum_{i=1}^{k} \hat{w}_{[i]} - \sum_{i \in S} \hat{w}_i - \sum_{j \in C} \hat{w}_j > \frac{4}{\gamma}\mathtt{gap}$.

In the following proof, given any non-empty set $C \subseteq [\![d]\!]$, we denote $\mathcal{G}(C) := \{i \in [\![d]\!] \mid \hat{w}_i \ge \underline{\hat{w}_C}\}$, and thus, the second point (ii) above can be written as: if $C \ne \emptyset$, $\mathcal{G}(C) \subseteq S \cup N \cup C$.

Proof of Case 1.

Let (S,N,C) is a valid SCG-tuple with $|N| > d-k$. Based on Definition 1, we have $S \cup N \cup C = [\![d]\!]$ and $|S \cup C| < k$.

$\Leftarrow$: If $S = \emptyset$, we have $C = [\![d]\!] \setminus N$ and $|C| = d - |N| < k$. Based on Definition 2, it is sufficient to show that $(\emptyset, N \setminus \{i\}, C)$ cannot form a valid tuple for any $i \in N$. To show invalidity, for any $i \in N$, we claim that the tuple $(\emptyset, N \setminus \{i\}, C)$ is invalid since it violates point (i): $|C| \ne \min\{k - |\emptyset|, d - |N \setminus \{i\}| - |\emptyset|\}$, where $\min\{k - |\emptyset|, d - |N \setminus \{i\}| - |\emptyset|\} = d - |N| + 1$. Therefore, $(\emptyset, N, C)$ is minimal.

$\Rightarrow$: If (S,N,C) is minimal, by Definition 2, we have either $S = \emptyset$ or $(S \setminus \{i\}, N, C \cup \{i\})$ is not valid for any $i \in S$. Suppose C is non-empty. For each $i \in S$, $(S \setminus \{i\}, N, C \cup \{i\})$ is valid, since (i) $|C \cup \{i\}| = |C| + 1$, $\min(k - |S \setminus \{i\}|, d - |N| - |S \setminus \{i\}|) = d - |N| - |S| + 1$ and we know $|S| + |N| + |C| = d$; (ii) it is obvious since $S \cup N \cup C = [\![d]\!]$; (iii) since (S,N,C) is a valid SCG-tuple, the inequality also holds for $(S \setminus \{i\}, N, C \cup \{i\})$. Thus $S = \emptyset$.

Proof of Case 2.

Assume (S,N,C) is a valid SCG-tuple with $|N| \le d-k$ and $C = \emptyset$. Based on Definition 1 for SCG-tuple, we have $|S| = k$. Let i^* be the index with respect to the largest component in $\hat{\boldsymbol{w}}$, i.e., $\hat{w}_{i^*} = \hat{w}_{[1]}$. We have the following condition $\max_{i \in S} \hat{w}_i < \hat{w}_{[1]}$ is equivalent to say $i^* \notin S$.

$\Leftarrow$: If $N = \emptyset$ and $i^* \notin S$, it is sufficient to show that $(S \setminus \{i\}, \emptyset, \{i\})$ is not a valid SCG-tuple for any $i \in S$. This is true because it violates the second point: (ii) for any j such that $\hat{w}_j \ge \hat{w}_i$, we have $j \in (S \setminus \{i\} \cup \emptyset \cup \{i\}) = S$. However, $\hat{w}_{i^*} \ge \hat{w}_i$ and $i^* \notin S$ by assumption. Contradiction.

$\Rightarrow$: If $(S, N, \emptyset)$ is minimal, we show two conditions listed in Case 2, respectively, by contradiction.

- Suppose $N \ne \emptyset$. Then $(S, N \setminus \{i\}, \emptyset)$ is a valid SCG-tuple for any $i \in N$, since (i) $|\emptyset| = 0$ and $\min(k - |S|, d - |N \setminus \{i\}| - |S|) = k - |S| = 0$; (ii) no need to verify; (iii) since $(S, N, \emptyset)$ is a valid tuple, the inequality also holds for $(S, N \setminus \{i\}, \emptyset)$. Contradiction with $(S, N, \emptyset)$ being minimal. Thus $N = \emptyset$.

- Suppose $i^* \in S$. We claim that $(S \setminus \{i^*\}, \emptyset, \{i^*\})$ is also a valid SCG-tuple, since (i) $|\{i^*\}| = 1$ and $\min(k - |S \setminus \{i^*\}|, d - |\emptyset| - |S \setminus \{i^*\}|) = k - |S \setminus \{i^*\}| = k - |S| + 1 = 1$; (ii) $C = \{i^*\}$, and it's obvious; (iii) since $(S, \emptyset, \emptyset)$ is a valid tuple, the inequality also holds for $(S \setminus \{i^*\}, \emptyset, \{i^*\})$. Contradiction with $(S, \emptyset, \emptyset)$ being minimal. Thus we have $i^* \notin S$.

Proof of Case 3.

Assume (S, N, C) is a valid SCG-tuple with $|N| \leq d - k$ and $C \neq \emptyset$. Based on validity, we have $|S \cup C| = k$. Besides, S and N cannot be empty at the same time, in order to generate a valid screening cut. Therefore, we consider the following three scenarios.

1. The first scenario: $S = \emptyset$, $N \neq \emptyset$. It is sufficient to focus on the N conditions.

$\Leftarrow$: Suppose $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$. Then for any $i \in N$, $(\emptyset, N \setminus \{i\}, C)$ is not valid since it violates the second point (ii) $i \notin ((\emptyset \cup N \setminus \{i\}) \cup C)$ and $\hat{w}_i \geq \underline{\hat{w}_C}$.

$\Rightarrow$: Suppose, for any $i \in N$, we have $(\emptyset, N \setminus \{i\}, C)$ is not valid, then we want to show $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$. Prove by contradiction. Suppose there exists some $j \in N$ such that $\hat{w}_j < \underline{\hat{w}_C}$. Then $(\emptyset, N \setminus \{j\}, C)$ is a valid tuple by verifying that (i) $|C| = \min\{k, d - |N \setminus \{j\}|\} = k$; (ii) We have $\mathcal{G}(C) \subseteq (N \cup C)$ based on the validness of SCG-tuple $(\emptyset, N, C)$. Based on the assumption on j, we have $j \notin \mathcal{G}(C)$, and thus $\mathcal{G}(C) \subseteq ((N \setminus \{j\}) \cup C)$; (iii) the inequality holds since SCG-tuple $(\emptyset, N, C)$ is a valid tuple by assumption. Then, we get a contradiction which implies $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$.

2. The second scenario: $S \neq \emptyset$, $N = \emptyset$. It is sufficient to focus on the S conditions.

$\Leftarrow$: If there exists some $j \notin (S \cup \emptyset \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \hat{w}_i$, we want to show that for any $i' \in S$, $(S \setminus \{i'\}, \emptyset, C \cup \{i'\})$ is not valid. This is true, since it violates the second point (ii) we have $\hat{w}_j > \max_{i \in S} \hat{w}_i \geq \hat{w}_{i'} \geq \underline{\hat{w}_{C \cup \{i'\}}}$ and $j \notin (S \cup C)$.

$\Rightarrow$: If $(S \setminus \{i\}, \emptyset, C \cup \{i\})$ is invalid for all $i \in S$, we show conditions listed in Case 3 using contradiction.

- Suppose there exists some $i' \in S$ such that $\hat{w}_{i'} \geq \underline{\hat{w}_C}$. Then $(S \setminus \{i'\}, \emptyset, C \cup \{i'\})$ is valid, by verifying that (i) $|C \cup \{i'\}| = |C| + 1$, and $\min(k - |S \setminus \{i'\}|, d - |S \setminus \{i'\}|) = k - |S| + 1$. We have $|S + |C| = k$ by assumption; (ii) since $(S, \emptyset, C)$ is valid, $\mathcal{G}(C) = \mathcal{G}(C \cup \{i'\}) \subseteq (S \cup C)$; (iii) since $(S, \emptyset, C)$ is valid, the inequality also holds for $(S \setminus \{i'\}, \emptyset, C \cup \{i'\})$. Contradiction. Thus we have $\underline{\hat{w}_C} > \max_{i \in S} \hat{w}_i$.

- Suppose there does not exist $j \notin (S \cup \emptyset \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \hat{w}_i$. Denote $i^* \in S$ such that $\hat{w}_{i^*} = \max_{i \in S} \hat{w}_i$. We have $(S \setminus \{i^*\}, \emptyset, C \cup \{i^*\})$ is a valid tuple, by verifying (i) $|C \cup \{i^*\}| = |C| + 1$, and $\min\{k - |S \setminus \{i^*\}|, d - |S \setminus \{i^*\}|\} = k - |S| + 1$. We have $|S| + |C| = k$ by assumption; (ii) since $(S, \emptyset, C)$ is valid, we have $\mathcal{G}(C) \subseteq (S \cup C)$. Thus $\mathcal{G}(C \cup \{i^*\}) = \mathcal{G}(C) \cup \{i^*\} \subseteq (S \cup C)$; (iii) since $(S, \emptyset, C)$ is a valid tuple, the inequality also holds for $(S \setminus \{i^*\}, \emptyset, C \cup \{i^*\})$. Contradiction. Thus there exist some $j \notin (S \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \hat{w}_i$.

3. The third scenario: $S \neq \emptyset$, $N \neq \emptyset$.

$\Leftarrow$: First we show that for any $i \in N$, $(S, N \setminus \{i\}, C)$ is not valid. This is true, since it violates the second point (ii) given $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$, $i \notin (S \cup (N \setminus \{i\}) \cup C)$ and $\hat{w}_i \geq \underline{\hat{w}_C}$.

Next we show that for any $i' \in S$, $(S \setminus \{i'\}, N, C \cup \{i'\})$ is not valid. Given some $j \notin (S \cup N \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \hat{w}_i$, it violates the second point (ii) $\hat{w}_j > \max_{i \in S} \hat{w}_i \geq \hat{w}_{i'} \geq \underline{\hat{w}_{C \cup \{i'\}}}$ and $j \notin (S \cup N \cup C)$. Then $(S \setminus \{i\}, N, C \cup \{i\})$ is not valid as claimed.

$\Rightarrow$: First we show that $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$. Prove by contradiction. Suppose there exists some $j \in N$ such that $\hat{w}_j < \underline{\hat{w}_C}$. Then $(S, N \setminus \{j\}, C)$ is a valid tuple by verifying that (i) $|C| = \min\{k - |S|, d - |N \setminus \{j\}| - |S|\} = k - |S|$; (ii) since (S, N, C) is valid, we have $\mathcal{G}(C) \subseteq (S \cup N \cup C)$. Moreover, $j \notin \mathcal{G}(C)$ implies $\mathcal{G}(C) \subseteq (S \cup (N \setminus \{j\}) \cup C)$; (iii) the inequality holds since (S, N, C) is a valid tuple by assumption. Contradiction. Thus we have $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$.

Next, we show that there exists some $j \notin (S \cup N \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \hat{w}_i$. Similarly, we prove by using contradiction twice.

- Suppose there exists some $i' \in S$ such that $\hat{w}_{i'} \geq \underline{\hat{w}_C}$. Then $(S \setminus \{i'\}, N, C \cup \{i'\})$ is valid, by verifying that (i) $|C \cup \{i'\}| = |C| + 1$, and $\min(k - |S \setminus \{i'\}|, d - |N| - |S \setminus \{i'\}|) = k - |S| + 1$. We have $|S + |C| = k$ by assumption; (ii) since (S, N, C) is valid, we have $\mathcal{G}(C) = \mathcal{G}(C \cup \{i'\}) \subseteq (S \cup N \cup C)$; (iii) since (S, N, C) is valid, the inequality also holds for $(S \setminus \{i\}, N, C \cup \{i\})$. Contradiction. Thus we have $\underline{\hat{w}_C} > \max_{i \in S} \hat{w}_i$.

- Since we have proved that $\underline{\hat{w}_N} \geq \underline{\hat{w}_C}$, it is sufficient to show that there exists some $j \notin (S \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \hat{w}_i$. Prove by contradiction. Suppose such j does not exist. Denote $i^* \in S$ such that $\hat{w}_{i^*} = \max_{i \in S} \hat{w}_i$. We have $(S \setminus \{i^*\}, N, C \cup \{i^*\})$ is a valid tuple, by verifying (i) $|C \cup \{i^*\}| = |C| + 1$, and $\min\{k - |S \setminus \{i^*\}|, d - |N| - |S \setminus \{i^*\}|\} = k - |S| + 1$. We have $|S| + |C| = k$ by assumption; (ii) since (S, N, C) is valid, we have $\mathcal{G}(C) \subseteq (S \cup N \cup C)$. Thus $\mathcal{G}(C \cup \{i^*\}) = \mathcal{G}(C) \cup \{i^*\} \subseteq (S \cup N \cup C)$; (iii) since (S, N, C) is a valid tuple, the inequality also holds for $(S \setminus \{i^*\}, N, C \cup \{i^*\})$. Contradiction. Thus there exist some $j \notin (S \cup N \cup C)$ such that $\underline{\hat{w}_C} > \hat{w}_j > \max_{i \in S} \hat{w}_i$.

$\square$

A.6 Proof of Proposition 4

Proposition 4. *Suppose components in $\tilde{\boldsymbol{w}}$ are distinct. Given any non-trivial $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$. Let the minimum consecutive partitions of* $\texttt{supp}(\boldsymbol{y})$ *and* $\texttt{comp}(\texttt{supp}(\boldsymbol{y}))$ *be*

$$\texttt{supp}(\boldsymbol{y}) = \cup_{j=1}^{m} A_j(\boldsymbol{y}) \quad \textit{and} \quad \texttt{comp}(\texttt{supp}(\boldsymbol{y})) = \cup_{j=1}^{m} B_j(\boldsymbol{y})$$

with some $m \leq k$. Then, one can construct minimal SCG-tuple (S, N, C) with $S \cup C = \texttt{supp}(\boldsymbol{y})$ as follows:

(1) If $|\texttt{supp}(\boldsymbol{y})| < k$, then

$$(S, N, C) = (\emptyset,\ [\![d]\!] \setminus \texttt{supp}(\boldsymbol{y}),\ \texttt{supp}(\boldsymbol{y}))$$

is a minimal SCG-tuple.

(2) If $|\texttt{supp}(\boldsymbol{y})| = k$, for $\ell = 1, \ldots, m-1$, we have the following SCG-tuple

$$(S, N, C) = \left(\cup_{j=\ell+1}^{m} A_j(\boldsymbol{y}),\ \cup_{j=1}^{\ell} B_j(\boldsymbol{y}),\ \cup_{j=1}^{\ell} A_j(\boldsymbol{y})\right)$$

is a minimal SCG-tuple. For $\ell = 0$ or $\ell = m$, define

$$(S_1, N_1, C_1) = (\texttt{supp}(\boldsymbol{y}),\ \emptyset,\ \emptyset) \quad \textit{or} \quad (S_2, N_2, C_2) = (\emptyset,\ \texttt{comp}(\texttt{supp}(\boldsymbol{y})),\ \texttt{supp}(\boldsymbol{y}))\ \ ,$$

respectively. Then (S_2, N_2, C_2) is minimal. Moreover, if $B_1(\boldsymbol{y}) \neq \emptyset$, (S_1, N_1, C_1) is minimal as well.

Proof. We separately give proofs on the two scenarios presented in Proposition (4).
1. The first scenario: $|\texttt{supp}(\boldsymbol{y})| < k$

In this case, we have $|S \cup C| = |\texttt{supp}(\boldsymbol{y})| < k$, and $N = [\![d]\!] \setminus (S \cup C) = [\![d]\!] \setminus \texttt{supp}(\boldsymbol{y})$ based on validity. From Proposition (3), we know that if $|N| > d - k$, then (S, N, C) is minimal if and only if $S = \emptyset$. Thus we can construct the minimal tuple as $(S, N, C) = (\emptyset, [\![d]\!] \setminus \texttt{supp}(\boldsymbol{y}), \texttt{supp}(\boldsymbol{y}))$.
2. The second scenario: $|\texttt{supp}(\boldsymbol{y})| = k$

In this case, we have $|S \cup C| = |\texttt{supp}(\boldsymbol{y})| = k$, and $|N| \le d - k$ based on validity.

1. First, we construct a minimal tuple with $C = \emptyset$. Thus we have $S = \texttt{supp}(\boldsymbol{y})$. From Proposition (3), we know that $(\texttt{supp}(\boldsymbol{y}), N, \emptyset)$ is minimal if and only if $N = \emptyset$, and $\max_{i \in \texttt{supp}(\boldsymbol{y})} \tilde{w}_i < \tilde{w}_{[1]}$. Therefore, $(\texttt{supp}(\boldsymbol{y}), \emptyset, \emptyset)$ is minimal if and only $\max_{i \in \texttt{supp}(\boldsymbol{y})} \tilde{w}_i < \tilde{w}_{[1]}$, which is equivalent to $B_1(\boldsymbol{y}) \neq \emptyset$ since $\tilde{\boldsymbol{w}}$ is in decreasing order.

2. Second, we construct minimal tuple with $C \neq \emptyset$. Denote $i^* \in C \subseteq \texttt{supp}(\boldsymbol{y})$ the index with respect to the smallest component in $\tilde{w}_C$, i.e., $\tilde{w}_{i^*} = \underline{\tilde{w}_C}$. We have two subcases:

   - If $i^* \in A_m(\boldsymbol{y})$, then $N \supseteq \texttt{comp}(\texttt{supp}(\boldsymbol{y}))$ to ensure validity. Based on Proposition (3), we have minimal tuple as $(S, N, C) = (\emptyset, \texttt{comp}(\texttt{supp}(\boldsymbol{y})), \texttt{supp}(\boldsymbol{y}))$.
   - If $i^* \in A_\ell(\boldsymbol{y})$ for some $\ell = 1, \ldots, m-1$, similarly, we can construct minimal SCG-tuple as $(S, N, C) = (\cup_{j=\ell+1}^m A_j(\boldsymbol{y}),\ \cup_{j=1}^l B_j(\boldsymbol{y}),\ \cup_{j=1}^l A_j(\boldsymbol{y}))$.

□

A.7 Proof of bounds for potential screening ability

Proof. For any screening cut, it removes all the feasible points that satisfy $z_i = 1$, for all $i \in S$ and $z_j = 0$, for all $j \in N$. If $|N| > d - k$, then based on validity, we have $|S| + |N| + |C| = d$ and thus $(S \cup N \cup C) = [\![d]\!]$. The remaining index set is C and we can choose up to $|C|$ one's within C. We have at most $\sum_{i=0}^{|C|} \binom{|C|}{i} = \sum_{i=0}^{|C|} \binom{d-|S \cup N|}{i}$ eliminated points. Similarly, if $|N| \le d - k$, we have $|S| + |C| = k$ and $(S \cup N \cup C) \subseteq [\![d]\!]$. The remaining index set is $[\![d]\!] \setminus (S \cup N)$ and we can choose up to $|C|$ one's. Thus we have at most $\sum_{i=0}^{|C|} \binom{d-|S \cup N|}{i}$ eliminated points. □

A.8 Proofs in Section 3.4

A.8.1 Inclusive Cuts Generation

Claim 1. *Given $\mathcal{T}_i$ with $i \in [\![k+1, d]\!]$, $\mathcal{T}_i \neq \emptyset$ if and only if there exists some $\boldsymbol{v} \in K(\tilde{\boldsymbol{w}})$ such that* $\texttt{supp}(\boldsymbol{v}) = [\![i - (k-1), i]\!]$. *Moreover, if $\mathcal{T}_i \neq \emptyset$, we have $\mathcal{T}_j \neq \emptyset$ for all $j = i+1, \ldots, d$.*

Proof. We show two results in Claim 1 separately.
Proof on the first result. One direction is obvious and we suppose $\mathcal{T}_i \neq \emptyset$. Pick any $\boldsymbol{y}' \in \mathcal{T}_i$. Given that $\tilde{\boldsymbol{w}}$ is in decreasing order and $\boldsymbol{y}' \in K(\tilde{\boldsymbol{w}})$, we have $\tilde{\boldsymbol{w}}^T \boldsymbol{v} \le \tilde{\boldsymbol{w}}^T \boldsymbol{y}' < c(\tilde{\boldsymbol{w}})$, since $\boldsymbol{v} \in \{0,1\}^d$ and $\texttt{supp}(\boldsymbol{v}) = [\![i - (k-1), i]\!]$ as assumed. Thus $\boldsymbol{v} \in K(\tilde{\boldsymbol{w}})$.
Proof on the second result. Given $\mathcal{T}_{i'} \neq \emptyset$, it is sufficient to show that $\mathcal{T}_{i'+1} \neq \emptyset$. Based on the first part, we have $\boldsymbol{v}_{i'} \in K(\tilde{\boldsymbol{w}})$ and $\texttt{supp}(\boldsymbol{v}_{i'}) = [\![i' - (k-1), i']\!]$. We construct $\boldsymbol{v}_{i'+1} \in \{0,1\}^d$ and $\texttt{supp}(\boldsymbol{v}_{i'+1}) = [\![i'+1-(k-1), i'+1]\!]$. It is obvious that $\boldsymbol{v}_{i'+1}^T \tilde{\boldsymbol{w}} < \boldsymbol{v}_{i'}^T \tilde{\boldsymbol{w}} < c(\tilde{\boldsymbol{w}})$. Then $\boldsymbol{v}_{i'+1} \in K(\tilde{\boldsymbol{w}})$ and we are done. □

A.8.2 Exclusive Cuts Generation

Claim 2. *Given $\mathcal{T}'_i$ with $i \in [\![1, k-1]\!]$, we have $\mathcal{T}'_i \neq \emptyset$ if and only if there exists some $\boldsymbol{v} \in K(\tilde{\boldsymbol{w}})$ such that $\texttt{supp}(\boldsymbol{v}) = [\![i]\!] \cup [\![d-(k-i-1), d]\!]$; Furthermore, if $\mathcal{T}'_i \neq \emptyset$, we have $\mathcal{T}'_j \neq \emptyset$, for all $j = 1, \ldots, i-1$.*

Proof. We show two results in Claim 2 separately.
Proof on the first result. One direction is obvious and we suppose $\mathcal{J}_i \neq \emptyset$. Pick any $\boldsymbol{y}' \in \mathcal{J}_i$. Given that $\tilde{\boldsymbol{w}}$ is in decreasing order and $\boldsymbol{y}' \in K(\tilde{\boldsymbol{w}})$, we have $\tilde{\boldsymbol{w}}^T\boldsymbol{v} \leq \tilde{\boldsymbol{w}}^T\boldsymbol{y}' < c(\tilde{\boldsymbol{w}})$, since $\boldsymbol{v} \in \{0,1\}^d$ and $\texttt{supp}(\boldsymbol{v}) = [\![i]\!] \cup [\![d-(k-i-1), d]\!]$ as assumed. Thus $\boldsymbol{v} \in K(\tilde{\boldsymbol{w}})$.
Proof on the second result. Given $\mathcal{J}_{i'} \neq \emptyset$, it is sufficient to show that $\mathcal{J}_{i'-1} \neq \emptyset$. Based on the first part, we have $\boldsymbol{v}_{i'} \in K(\tilde{\boldsymbol{w}})$ and $\texttt{supp}(\boldsymbol{v}_{i'}) = [\![i']\!] \cup [\![d-(k-i'-1), d]\!]$. We construct $\boldsymbol{v}_{i'-1} \in \{0,1\}^d$ and $\texttt{supp}(\boldsymbol{v}_{i'-1}) = [\![i'-1]\!] \cup [\![d-(k-i'), d]\!]$. It is obvious that $\boldsymbol{v}_{i'-1}^T\tilde{\boldsymbol{w}} < \boldsymbol{v}_{i'}^T\tilde{\boldsymbol{w}} < c(\tilde{\boldsymbol{w}})$. Then $\boldsymbol{v}_{i'-1} \in K(\tilde{\boldsymbol{w}})$ and we are done. □

A.8.3 SCG Algorithm Design

Claim 3. *By setting the maximal number of inclusive or exclusive cuts $\sharp_{\texttt{max}} = +\infty$, Algorithm 1 outputs all minimal inclusive cuts with $\sharp_{\texttt{len}} \leq d-k$, or exclusive cuts with maximum length $\sharp_{\texttt{len}} \leq k-1$, respectively.*

Proof. We show results on inclusive cuts generation and exclusive cuts generation separately.
Inclusive cuts. Based on previous analysis, for each $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$ with $|\texttt{supp}(\boldsymbol{y})| < k$, the minimal inclusive cut has length $|[\![d]\!] \setminus \texttt{supp}(\boldsymbol{y})| > d-k \geq \sharp_{\texttt{len}}$. Therefore, we have $\boldsymbol{y} \in \mathcal{O}$ to generate such inclusive cuts.

For any $\boldsymbol{y} \in \mathcal{O}$ that can generate minimal inclusive cut with length at most $\sharp_{\texttt{len}}$, it is sufficient to prove that $\boldsymbol{y} \in \mathcal{T}_{i'}$ for some $i' \in [\![k+1, d]\!]$ and $\mathcal{T}_{i'}$ shall be found in STEP 2 of Algorithm 1.

Given a maximum length $\sharp_{\texttt{len}}$, we have $|\texttt{comp}(\texttt{supp}(\boldsymbol{y}))| = i'-k \leq \sharp_{\texttt{len}}$, which is equivalent to $i' \leq k + \sharp_1$. Therefore, we have the searching index s initialized in STEP 1 satisfies $s \leq i'$. Since $\sharp_{\texttt{max}} = +\infty$, the Algorithm 1 will not stop until $s = \min\{k+\sharp_1, d\}$. Thus we have $s = i'$ before termination.
Exclusive cuts. Based on previous analysis, for each $\boldsymbol{y} \in K(\tilde{\boldsymbol{w}})$ with $|\texttt{supp}(\boldsymbol{y})| = k$ and $B_1(\boldsymbol{y}) \neq \emptyset$, the minimal exclusive cut has length $|\texttt{supp}(\boldsymbol{y})| = k > \sharp_{\texttt{len}}$. Therefore, we have $\boldsymbol{y} \in \mathcal{O}'$ to generate such exclusive cuts.

For any $\boldsymbol{y} \in \mathcal{O}'$ that can generate minimal exclusive cut with length at most $\sharp_{\texttt{len}}$, it is sufficient to prove that $\boldsymbol{y} \in \mathcal{T}'_{i'}$ for some $i' \in [\![1, k-1]\!]$ and $\mathcal{T}'_{i'}$ shall be found in STEP 2 of Algorithm 1.

Given a maximum length $\sharp_1$, we have $|\cup_{j=2}^m A_j(\boldsymbol{y}')| = k-i' \leq \sharp_{\texttt{len}}$, which is equivalent to $i' \geq k - \sharp_1$. Therefore, in STEP 1, we have the initial searching index $s \geq i'$. Since $\sharp_{\texttt{max}} = +\infty$, the Algorithm 1 will not stop until $s = \max\{k-\sharp_1, 1\}$. Thus we have $s = i'$ before termination. □

Here, we present the pseudo-code (Algorithm 2) for the recursion used in STEP 2 of Algorithm 1. Given a search index $s \in \texttt{SR}_{\texttt{inc}}$ or $\texttt{SR}_{\texttt{exc}}$, the computational complexity of enumerating $\mathcal{T}_s$ or $\mathcal{T}'_s$ is $O\big((k-1)^2 \cdot \binom{s-1}{k-1}\big)$ or $O\big((k-s)^2 \cdot \binom{d-s-1}{k-s}\big)$ for inclusive or exclusive cuts generation.

Remark 2. *The Algorithm 2 generates all possible supports of $\texttt{SN}$ elements from index set $\texttt{SR}$ that satisfy outer if condition presented in Algorithm 2. If outer if-condition is satisfied for every possible support, Algorithm 2 is equivalent to brute-force enumeration. Specifically, there are $\binom{|\texttt{SR}|}{\texttt{SN}}$*

Algorithm 2: Recursion in STEP 2 of Algorithm 1

```
Input: Sparsity level $k$, sorted $\tilde{\boldsymbol{w}}$, current search index $s$ in STEP 2.
Set weight limit $\texttt{WL} = c(\tilde{\boldsymbol{w}}) - \tilde{\boldsymbol{w}}_s$ for inclusive cuts, $\texttt{WL} = c(\tilde{\boldsymbol{w}}) - \sum_{j=1}^{s} \tilde{w}_j$ for exclusive cuts.
Set searching range $\texttt{SR} = [\![s-1]\!]$ for inclusive cuts, $\texttt{SR} = [\![s+2, d]\!]$ for exclusive cuts.
Set select number $\texttt{SN} = k-1$ for inclusive cuts, $\texttt{SN} = k-s$ for exclusive cuts.
Initialized target partition subset $\mathcal{T}_s = \emptyset$ and temporary enumerating support $\texttt{T}_{\texttt{supp}} = \emptyset$.
Function RI(WL, $\tilde{\boldsymbol{w}}$, SR, SN, $\mathcal{T}_s$, $T_{supp}$):
    for index $i \in [\![|\texttt{SR}| - \texttt{SN} + 1]\!]$ do
        Select $i$th smallest index in SR, i.e., index $j = \texttt{SR}[i]$;
        Compute $\texttt{min-weight} = \tilde{w}_j + \sum_{t=|\texttt{SR}|-\texttt{SN}+2}^{|\texttt{SR}|} \tilde{w}_{\texttt{SR}[t]}$;
        if min-weight $< \texttt{WL}$ then
            Add index $j$ to current enumerating support, i.e., $\texttt{T}_{\texttt{supp}} = \texttt{T}_{\texttt{supp}} \cup \{j\}$;
            if $SN = 1$ then
                Add current temporary enumerating support $\texttt{T}_{\texttt{supp}}$ into $\mathcal{T}_s$, i.e.,
                  $\mathcal{T}_s = \mathcal{T}_s \cup \{\texttt{T}_{\texttt{supp}}\}$ ;
            else
                Update weight limit $\texttt{WL} = \texttt{WL} - \tilde{w}_j$;
                Remove all the indices that are smaller than or equal to $j = \texttt{SR}[i]$ from SR, i.e.,
                  $\texttt{SR} = \{\texttt{SR}[i+1], \ldots, \texttt{SR}[|\texttt{SR}|]\}$;
                Update current $\texttt{SN} = \texttt{SN} - 1$;
                RI(WL, $\tilde{\boldsymbol{w}}$, SR, SN, $\mathcal{T}_s$, $T_{supp}$);
            Remove index $j$ from temporary support set $\texttt{T}_{\texttt{supp}}$, i.e., $\texttt{T}_{\texttt{supp}} = \texttt{T}_{\texttt{supp}} \setminus \{j\}$ ;
    return target partition set $\mathcal{T}_s$;
```

possible support sets, and each support requires at most $O(SN^2)$ elementary algebraic operations in our recursion Algorithm 2. Therefore, the total computational complexity is $O\big(SN^2 \cdot \binom{|SR|}{SN}\big)$. The main difference between our recursion Algorithm 2 and brute-force enumeration lies in identifying whether index j presented in Algorithm 2 can be added in any possible support.

B Tables in Section 4.3

Summary of pre-solving time. We report the pre-solving time for selecting inclusive and exclusive cuts by SCG and SSR in Table 1. The first column indicates the employed methods. The second to fourth columns report the pre-solving time on synthetic datasets with dimension $d \in \{100, 300, 500\}$, respectively. Each cell contains a pair of format $(\leq t^{\mathtt{ub}}_{0.5},\ \leq t^{\mathtt{ub}}_{3.5})$. Here, given a fixed parameter setting/configuration and a fixed sequence of conducted ridge parameters $\{\gamma_i\}_{i\in I}$, $t^{\mathtt{ub}}_{0.5}$ and $t^{\mathtt{ub}}_{3.5}$ denote the upper bounds of pre-solving time among all distinct γs for SNR $\in \{0.5,\ 3.5\}$, respectively, i.e.,

$$t^{\mathtt{ub}}_{0.5} := \max_{i\in I}\ t^{\mathtt{pre}}_{\star}(\gamma_i; d, \mathrm{SNR} = 0.5) \quad \text{and} \quad t^{\mathtt{ub}}_{3.5} := \max_{i\in I}\ t^{\mathtt{pre}}_{\star}(\gamma_i; d, \mathrm{SNR} = 3.5)$$

with $\star \in \{\text{SCG, SSR}\}$ and $t^{\mathtt{pre}}_{\star}(\gamma_i; d, \mathrm{SNR})$ the pre-processing time for variable screening method $\star$ with dataset parameter configuration (d, SNR). The last two columns provide the pre-solving time on two real datasets: UJIndoorLoc and CNAE. Each cell contains the pair $(\leq t^{\mathtt{ub}}_{\mathtt{reduced}},\ \leq t^{\mathtt{ub}}_{\mathtt{all}})$, where $t^{\mathtt{ub}}_{\mathtt{reduced}}$ and $t^{\mathtt{ub}}_{\mathtt{all}}$ represents the upper bounds of pre-solving time among all distinct γs on the reduced sample dataset and whole dataset, respectively.

Table 1: Pre-solving time for SCG and SSR(s)

Methods	$d = 100$	$d = 300$	$d = 500$	UJIndoorLoc	CNAE
SCG	$(\leq 0.21,\ \leq 0.24)$	$(\leq 0.49,\ \leq 0.69)$	$(\leq 0.76,\ \leq 1.22)$	$(\leq 4.73,\ \leq 27.6)$	$(\leq 0.7,\ \leq 3.2)$
SSR	$(\leq 0.39,\ \leq 0.16)$	$(\leq 1.17,\ \leq 0.72)$	$(\leq 2.76,\ \leq 1.54)$	$(\leq 5.12,\ \leq 28.4)$	$(\leq 2.1,\ \leq 4.3)$

Notation used in cut characteristic tables. Before reporting numerical results on cut characteristics (Table 2, 3, 4, 5, 6, 7), let us first introduce notation that will be used in these tables. We use $n_{\mathtt{inc}}$ and $n_{\mathtt{exc}}$ to denote the average number of inclusive and exclusive cuts generated, respectively. The average number of binary variables involved in each inclusive and exclusive cut is represented as $l_{\mathtt{inc}}$ and $l_{\mathtt{exc}}$. We use $(\mathtt{NA}_{\mathtt{inc}}, \mathtt{NA}_{\mathtt{exc}})$ (a shorthand for "None Added") to denote the number of datasets where no inclusive cut or no exclusive cut is added.

Synthetic datasets - $d = 100$

Table 2: $d = 100,\ \mathrm{SNR} = 0.5$

γ	Methods	$(n_{\texttt{INC}}, n_{\texttt{EXC}})$	$(l_{\texttt{INC}}, l_{\texttt{EXC}})$	$(\texttt{NA}_{\texttt{INC}}, \texttt{NA}_{\texttt{EXC}})$
2.2	SCG	(3.9, 89)	(1.27, 1.19)	(1, 0)
	SSR	(2.9, 71.9)	-	(1, 0)
2	SCG	(3, 89.5)	(1.09, 1.21)	(1, 0)
	SSR	(2.6, 70.7)	-	(1, 0)
1.8	SCG	(3.3, 89.2)	(1.4, 1.34)	(2, 0)
	SSR	(2.1, 58.3)	-	(4, 1)
1.6	SCG	(2.3, 89.1)	(1.15, 1.32)	(0, 0)
	SSR	(1.9, 60.3)	-	(1, 0)
1.4	SCG	(1.8, 89.2)	(1.28, 1.4)	(4, 0)
	SSR	(1, 52.9)	-	(4, 0)

Table 3: $d = 100,\ \mathrm{SNR} = 3.5$

γ	Methods	$(n_{\texttt{INC}}, n_{\texttt{EXC}})$	$(l_{\texttt{INC}}, l_{\texttt{EXC}})$	$(\texttt{NA}_{\texttt{INC}}, \texttt{NA}_{\texttt{EXC}})$
2.2	SCG	(5.1, 86.5)	(1.08, 1.08)	(0, 0)
	SSR	(4.8, 78.9)	-	(0, 0)
2	SCG	(4.9, 88.7)	(1.25, 1.16)	(0, 0)
	SSR	(3.7, 77.5)	-	(0, 0)
1.8	SCG	(5, 87.3)	(1.35, 1.15)	(0, 0)
	SSR	(2.8, 73.8)	-	(0, 0)
1.6	SCG	(4.5, 88.5)	(1.26, 1.2)	(0, 0)
	SSR	(3.1, 70.3)	-	(1, 0)
1.4	SCG	(3.6, 88.7)	(1.13, 1.24)	(0, 0)
	SSR	(3.2, 67.7)	-	(1, 0)

Synthetic datasets - $d = 300$

Table 4: $d = 300,\ \text{SNR} = 0.5$

γ	Methods	$(n_{\texttt{INC}}, n_{\texttt{EXC}})$	$(l_{\texttt{INC}}, l_{\texttt{EXC}})$	$(\texttt{NA}_{\texttt{INC}}, \texttt{NA}_{\texttt{EXC}})$
2.2	SCG	(2.6, 290)	(1.55, 1.21)	(2, 0)
	SSR	(0.9, 228)	-	(4, 0)
2	SCG	(1.1, 290)	(1.5, 1.25)	(4, 0)
	SSR	(0.6, 217.1)	-	(6, 0)
1.8	SCG	(1.6, 290)	(1.52, 1.25)	(3, 0)
	SSR	(0.6, 218)	-	(5, 0)
1.6	SCG	(0.7, 290)	(1.7, 1.52)	(5, 0)
	SSR	(0.2, 138.4)	-	(8, 2)
1.4	SCG	(0.6, 290)	(1.67, 1.52)	(7, 0)
	SSR	(0.2, 140.5)	-	(9, 1)

Table 5: $d = 300,\ \text{SNR} = 3.5$

γ	Methods	$(n_{\texttt{INC}}, n_{\texttt{EXC}})$	$(l_{\texttt{INC}}, l_{\texttt{EXC}})$	$(\texttt{NA}_{\texttt{INC}}, \texttt{NA}_{\texttt{EXC}})$
2.2	SCG	(2.3, 290)	(1.15, 1.12)	(0, 0)
	SSR	(1.9, 255.7)	-	(1, 0)
2	SCG	(3.2, 290)	(1.48, 1.11)	(0, 0)
	SSR	(1.5, 258.6)	-	(2, 0)
1.8	SCG	(3.5, 290)	(1.36, 1.20)	(1, 0)
	SSR	(2, 232.7)	-	(1, 0)
1.6	SCG	(2.9, 290)	(1.31, 1.22)	(0, 0)
	SSR	(2.1, 225.6)	-	(2, 0)
1.4	SCG	(1.2, 290)	(1.52, 1.39)	(3, 0)
	SSR	(0.5, 177.9)	-	(6, 2)

Synthetic datasets - $d = 500$

Table 6: $d = 500,\ \text{SNR} = 0.5$

γ	Methods	$(n_{\texttt{INC}}, n_{\texttt{EXC}})$	$(l_{\texttt{INC}}, l_{\texttt{EXC}})$	$(\texttt{NA}_{\texttt{INC}}, \texttt{NA}_{\texttt{EXC}})$
2.2	SCG	(1.1, 490)	(1, 1.15)	(5, 0)
	SSR	(1.1, 418.2)	-	(5, 0)
2	SCG	(1.6, 490)	(1.38, 1.2)	(4, 0)
	SSR	(0.6, 390.3)	-	(5, 0)
1.8	SCG	(1, 490)	(1.92, 1.41)	(7, 0)
	SSR	(0.1, 288.1)	-	(9, 1)
1.6	SCG	(0.3, 490)	(1.33, 1.5)	(7, 0)
	SSR	(0.2, 245.9)	-	(8, 2)
1.4	SCG	(0.3, 490)	(2, 1.74)	(8, 0)
	SSR	(0, 127.5)	-	(10, 6)

Table 7: $d = 500,\ \text{SNR} = 3.5$

γ	Methods	$(n_{\texttt{INC}}, n_{\texttt{EXC}})$	$(l_{\texttt{INC}}, l_{\texttt{EXC}})$	$(\texttt{NA}_{\texttt{INC}}, \texttt{NA}_{\texttt{EXC}})$
2.2	SCG	(2, 490)	(1.28, 1.1)	(1, 0)
	SSR	(1.3, 440.1)	-	(2, 0)
2	SCG	(2.1, 489.9)	(1.32, 1.1)	(2, 0)
	SSR	(1.5, 441.9)	-	(3, 0)
1.8	SCG	(2.3, 490)	(1.34, 1.22)	(2, 0)
	SSR	(1.4, 383.3)	-	(3, 1)
1.6	SCG	(1.2, 490)	(1.4, 1.23)	(3, 0)
	SSR	(0.7, 378.1)	-	(5, 0)
1.4	SCG	(1, 490)	(1.67, 1.53)	(4, 0)
	SSR	(0.3, 230.7)	-	(8, 3)

Real dataset - UJIndoorLoc

Table 8: UJIndoorLoc

γ	Methods	$(n^{\texttt{ALL}}_{\texttt{INC}}, n^{\texttt{ALL}}_{\texttt{EXC}})$	$(l^{\texttt{ALL}}_{\texttt{INC}}, l^{\texttt{ALL}}_{\texttt{EXC}})$	$(n^{\texttt{RED}}_{\texttt{INC}}, n^{\texttt{RED}}_{\texttt{EXC}})$	$(l^{\texttt{RED}}_{\texttt{INC}}, l^{\texttt{RED}}_{\texttt{EXC}})$
14000	SCG	(0, 510)	(NA, 1.14)	(0, 510)	(NA, 1.1)
	SSR	(0, 439)	-	(0, 459)	-
13000	SCG	(0, 510)	(NA, 1.14)	(0, 510)	(NA, 1.86)
	SSR	(0, 439)	-	(0, 69)	-
12000	SCG	(0, 510)	(NA, 1.15)	(0, 510)	(NA, 1.9)
	SSR	(0, 429)	-	(0, 51)	-
11000	SCG	(0, 510)	(NA, 1.80)	(0, 510)	(NA, 1.10)
	SSR	(0, 98)	-	(0, 456)	-
10000	SCG	(0, 510)	(NA, 1.83)	(0, 510)	(NA, 1.1)
	SSR	(0, 83)	-	(0, 459)	-
9000	SCG	(0, 510)	(NA, 1.29)	(0, 510)	(NA, 1.09)
	SSR	(0, 358)	-	(0, 461)	-
8000	SCG	(0, 510)	(NA, 1.29)	(0, 510)	(NA, 1.1)
	SSR	(0, 359)	-	(0, 459)	-
7000	SCG	(0, 510)	(NA, 1.77)	(0, 510)	(NA, 1.11)
	SSR	(0, 114)	-	(0, 452)	-
6000	SCG	(0, 510)	(NA, 1.87)	(0, 510)	(NA, 1.76)
	SSR	(0, 65)	-	(0, 120)	-
5000	SCG	(0, 510)	(NA, 2)	(0, 510)	(NA, 1.84)
	SSR	(0, 0)	-	(0, 80)	-
4000	SCG	(0, 510)	(NA, 2)	(0, 510)	(NA, 1.92)
	SSR	(0, 0)	-	(0, 38)	-
3000	SCG	(0, 510)	(NA, 2)	(0, 510)	(NA, 1.98)
	SSR	(0, 0)	-	(0, 9)	-

Real dataset - CNAE

Table 9: CNAE

γ	Methods	$(n^{\texttt{ALL}}_{\texttt{INC}}, n^{\texttt{ALL}}_{\texttt{EXC}})$	$(l^{\texttt{ALL}}_{\texttt{INC}}, l^{\texttt{ALL}}_{\texttt{EXC}})$	$(n^{\texttt{RED}}_{\texttt{INC}}, n^{\texttt{RED}}_{\texttt{EXC}})$	$(l^{\texttt{RED}}_{\texttt{INC}}, l^{\texttt{RED}}_{\texttt{EXC}})$
0.1	SCG	(3, 846)	(1, 1.01)	(4, 846)	(1, 1.02)
	SSR	(3, 831)	-	(4, 830)	-
0.09	SCG	(4, 846)	(1.5, 1.02)	(4, 846)	(1, 1.03)
	SSR	(2, 828)	-	(4, 821)	-
0.08	SCG	(4, 846)	(1.75, 1.07)	(4, 846)	(1.5, 1.06)
	SSR	(1, 780)	-	(2, 796)	-
0.077	SCG	(3, 846)	(1.67, 1.17)	(4, 846)	(1.5, 1.06)
	SSR	(1, 699)	-	(2, 792)	-
0.075	SCG	(2, 846)	(1.5, 2)	(4, 846)	(1.5, 1.09)
	SSR	(1, 0)	-	(2, 771)	-
0.074	SCG	(2, 846)	(1.5, 2)	(4, 846)	(1.5, 1.1)
	SSR	(1, 0)	-	(2, 754)	-
0.073	SCG	(2, 846)	(1.5, 2)	(3, 846)	(1.33, 1.1)
	SSR	(1, 0)	-	(2, 753)	-
0.07	SCG	(1, 846)	(1, 2)	(3, 846)	(1.33, 1.18)
	SSR	(1, 0)	-	(2, 692)	-
0.067	SCG	(1, 846)	(1, 2)	(3 846)	(1.33, 2)
	SSR	(1, 0)	-	(2, 0)	-
0.063	SCG	(1, 846)	(1, 2)	(3, 846)	(1.33, 2)
	SSR	(1, 0)	-	(2, 0)	-
0.06	SCG	(1, 846)	(1, 2)	(4, 846)	(1.75, 2)
	SSR	(1, 0)	-	(1, 0)	-
0.055	SCG	(1, 846)	(1, 2)	(2, 846)	(1.5, 2)
	SSR	(1, 0)	-	(1, 0)	-
0.05	SCG	(4, 846)	(2, 2)	(1, 846)	(1, 2)
	SSR	(0, 0)	-	(1, 0)	-
0.04	SCG	(0, 846)	(NA, 2)	(0, 846)	(NA, 2)
	SSR	(0, 0)	-	(0, 0)	-