Fast Presolving Framework For Sparsity Constrained Convex Quadratic Programming: Screening-Based Cut Generation and Selection

Haozhe Tan[1], Guanyi Wang[1]

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

February 14, 2026

Abstract

Screening is widely utilized for Mixed-Integer Programming (MIP) presolving. It aims to certify a *priori* whether one or multiple specific binary variables can be fixed to optimal values based on solutions from convex relaxations. This paper studies the challenge of solving Sparsity-constrained (strongly) Convex Quadratic Programming (SCQP) and proposes the Screening-based Cut Presolving Framework (SCPF). SCPF contains two parts: a Screening-based Cut Generation (SCG) rule and a Screening-based Cut Selection (SCS) method. We show that the SCG provides superior screening ability compared to existing screening methods, and achieves a finer balance between screening effectiveness and computational overhead. We then provide theoretical guarantees for the SCS method to ensure the selection of generated cuts with high screening ability. Extensive numerical experiments validate the theoretical findings and demonstrate that the proposed framework significantly outperforms state-of-the-art screening methods. Notably, our SCPF achieves a 1.7× to 3.0× acceleration in total running time, especially in challenging phases, across high-dimensional synthetic datasets, complex real-world instances, and simulation libraries from sparse identification of nonlinear dynamics.

1 Introduction & Preliminary

Screening is a prevalently used presolving method for modern Mixed-Integer Programming (MIP) solvers (Achterberg et al., 2020; Bestuzheva et al., 2021). It aims to certify a *priori*, before implementing the exact Branch-and-Bound (BnB) algorithm, whether specific subsets of binary variables can be fixed to their optimal values based on solutions obtained from convex relaxations, thereby reducing the search space.

This paper studies screening methods for *Sparsity-constrained (strongly) Convex Quadratic Programming (SCQP)*. SCQP covers diverse applications in modern operations research (Gao and Li, 2013; Hazimeh and Mazumder, 2020), machine learning (Pilanci et al., 2015; Bertsimas et al., 2021), and data analysis (which includes the widely studied Sparse Ridge Regression (SRR) (Bertsimas and Van Parys, 2020; Askari et al., 2022) as a special case). SCQP is formulated as:

$$v^* := \min_{\boldsymbol{\beta}\in\mathbb{R}^d} \ \mathcal{L}(\boldsymbol{\beta}) + \gamma\|\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \le k \ , \tag{1}$$

where $\mathcal{L}(\boldsymbol{\beta}) := \frac{1}{2}\boldsymbol{\beta}^\top \boldsymbol{Q}\boldsymbol{\beta} + \boldsymbol{q}^\top\boldsymbol{\beta}$ is a convex quadratic function, $\gamma > 0$ is a parameter that describes the level of strong convexity of objective. If the minimum eigenvalue of $\boldsymbol{Q}$ is strictly positive, one can increase γ to $\gamma + \lambda_{\min}(\boldsymbol{Q})$ while updating $\boldsymbol{Q}$ by a semidefinite matrix $\boldsymbol{Q} - \lambda_{\min}(\boldsymbol{Q})\boldsymbol{I}$.

The sparsity constraint $\|\boldsymbol{\beta}\|_0 \leq k$ restricts the support size $\texttt{supp}(\boldsymbol{\beta}) := \{i \in [\![d]\!] \mid \beta_i \neq 0\}$ to at most k. In the MIP community, this sparsity constraint is typically modeled by introducing "on-off" binary variables $\mathbf{z} \in Z^k := \{\mathbf{z} \in \{0,1\}^d \mid \mathbf{1}^\top \mathbf{z} \leq k\}$ and defining the feasible set as $\mathcal{S}^k := \{(\boldsymbol{\beta}, \mathbf{z}) \in \mathbb{R}^d \times Z^k \mid \beta_i(1 - z_i) = 0 \ \forall\, i \in [\![d]\!]\}$.

Screening methods rely on tractable convex relaxations of (1). To simplify the analysis, this paper focuses on the widely adopted (Xie and Deng, 2020; Atamturk and Gomez, 2020; Liu et al., 2023) perspective relaxation:

$$\begin{array}{rll} v_{\texttt{relax}} := \min_{\boldsymbol{\beta},\mathbf{z}} & \mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{i=1}^d \beta_i^2/z_i & \\ \text{s.t.} & \mathbf{z} \in Z^k_{\texttt{relax}} := \{\mathbf{z} \in [0,1]^d \mid \mathbf{1}^\top \mathbf{z} \leq k\} & , \end{array} \tag{2}$$

where the term β_i^2/z_i is defined as 0 if $\beta_i = z_i = 0$, and $+\infty$ if $\beta_i \neq 0,\ z_i = 0$. While tighter convex relaxations based on decompositions of semidefinite matrices, polyhedral outer approximations, and prospective reformulations exist (Frangioni et al., 2020; Han et al., 2022; Gómez and Xie, 2024; Atamturk and Gomez, 2025), establishing these decompositions by optimizing an additional semidefinite programming is often computationally intractable, especially in presolving. Plus, finding tighter relaxations is beyond the scope of this paper, and we emphasize that our proposed framework can be directly applied to different convex relaxations.

For preliminary, variable screening certifies whether a single binary variable z_i can be fixed to a specific value $\overline{z}_i \in \{0,1\}$ by checking the *optimality-based criterion:* $v_{\texttt{relax}}(\overline{z}_i) > v_{\texttt{ub}}$. Here, $v_{\texttt{relax}}(\overline{z}_i)$ represents the (optimal) value function of the relaxation (2) restricted by an additional constraint $z_i = \overline{z}_i$, and $v_{\texttt{ub}}$ is a valid upper bound for (1). If the criterion holds, z_i is safely fixed to $1 - \overline{z}_i$ in (1). However, directly evaluating the optimality-based criterion is computationally expensive, as it requires re-optimizing up to $2d$ restricted convex relaxations. To address this bottleneck, Atamturk and Gomez (2020); Deza and Atamturk (2022) propose *Safe Screening Rules (SSR)* based on the Fenchel dual reformulation of (2):

$$\begin{array}{rl} & \min_{(\boldsymbol{\beta},\mathbf{z}) \in \mathbb{R}^d \times Z^k_{\texttt{relax}}} \ \mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{i=1}^d \max_{p_i} \left(p_i \beta_i - \frac{p_i^2}{4} z_i \right) \\ = & \max_{\mathbf{p}} \min_{(\boldsymbol{\beta},\mathbf{z}) \in \mathbb{R}^d \times Z^k_{\texttt{relax}}} \ \mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{i=1}^d \left(p_i \beta_i - \frac{p_i^2}{4} z_i \right) \quad . \end{array} \tag{3}$$

Let its optimal primal-dual pair be $(\widehat{\boldsymbol{\beta}}, \widehat{\mathbf{z}}), \widehat{\mathbf{p}}$. Atamturk and Gomez (2020); Deza and Atamturk (2022) derive computationally cheaper lower bounds on $v_{\texttt{relax}}(\overline{z}_i)$ for screening, as summarized in Proposition 1.

Proposition 1 (Restatement from Atamturk and Gomez (2020); Deza and Atamturk (2022); Safe Screening Rules)**.** [1]

Given $(\widehat{\boldsymbol{\beta}}, \widehat{\mathbf{z}}), \widehat{\mathbf{p}}$ *an optimal primal-dual pair of* (3)*, assume there is no tie among components in* $\widehat{\mathbf{w}}$ *with* $\widehat{\mathbf{w}} := \widehat{\mathbf{p}} \circ \widehat{\mathbf{p}}$*. For any index* $j \in [\![d]\!]$*, the following SSR*

$$z_j = \begin{cases} 0, & \text{if} \quad \widehat{w}_j \leq \widehat{w}_{[k+1]} \quad \text{and} \quad \gamma \frac{\hat{w}_{[k]}}{4} - \gamma \frac{\hat{w}_j}{4} > v_{\texttt{ub}} - v_{\texttt{relax}} \\ 1, & \text{if} \quad \widehat{w}_j \geq \widehat{w}_{[k]} \quad \text{and} \quad -\gamma \frac{\hat{w}_{[k+1]}}{4} + \gamma \frac{\hat{w}_j}{4} > v_{\texttt{ub}} - v_{\texttt{relax}} \end{cases}$$

[1] We acknowledge that, a recent work, "*Logic Rules and Chordal Graphs for Sparse Learning*" by A. Deza, A. Gómez, and A. Atamtürk (ISMP 2024), proposes a refined safe screening rule using chordal graphs. However, no preprint is currently available (Jan/2026). Based on their presentation, our results differ significantly: we focus on a novel Screening-based Cut Generation (SCG) rule that balances screening ability with computational overhead, and a Screening-based Cut Selection (SCS) method with theoretical guarantees to improve computational efficiency.

screens whether a binary z_j is zero or one in optimal solution set to (1). Here, we use $\widehat{w}_{[j]}$ to denote the j-th largest component of $\widehat{\mathbf{w}}$.

As a brief literature review, variable screening and its variants have proven effective across various domains. In Mixed-Binary Linear Programming (MBLP), reduced-cost fixing methods (Schürmann and Mutzel, 2023; Yang, 2025) perform efficiently by developing proper choice mechanisms for Lagrangian dual multipliers. Recent advancements (Li et al., 2024; Jiang and Xie, 2025) improve vanilla screening methods by considering multiple binary variables to derive optimality cuts. Additionally, Dai and Chen (2025) demonstrates that prefixing pairs of binaries yields stronger implications for screening. Beyond linear problems, Atamturk and Gomez (2020); Deza and Atamturk (2022) provide scalable safe screening rules for SRR. Building on these concepts, Liu et al. (2023, 2025) propose a series of tailored BnB algorithms that incorporate screening-based safe lower bound estimations for k-sparse generalized linear models.

However, as illustrated in Figure 1, challenges remain, especially in the relatively weak strong-convexity regime (i.e., with low γ parameter, shaded light red and light yellow part, see subsection 4.2 part (T1) for details), across a wide range of typical SCQP problems. We observe that a significant portion of binary variables remains difficult to certify using existing screening rules in presolving, which motivates our central research question:

How and to what extent can we trade off between the effectiveness of screening and computational cost based on given convex relaxations?

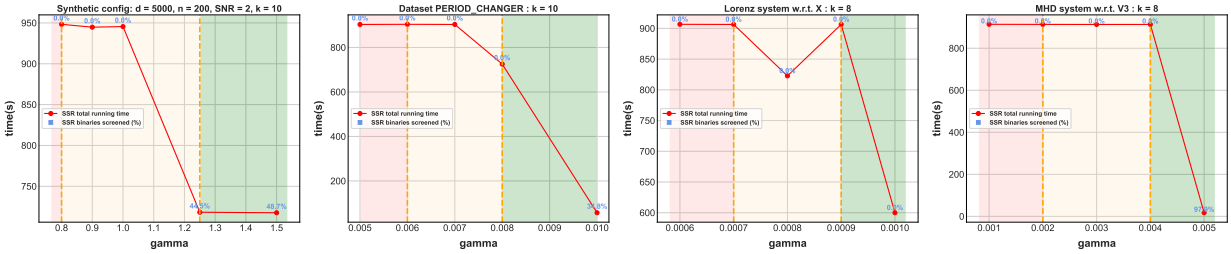


Figure 1. Changes in the percentage of binaries screened by SSR & corresponding total running time under different γ parameters for instances from synthetic dataset, real-world dataset, and simulation library for Sparse Identification of Nonlinear Dynamics (SINDy).

1.1 Main Contributions & Paper Organization

To address the above research question, this paper proposes the *Screening-based Cut Presolving Framework (SCPF)*. This framework contains two key parts: (1) a *Screening-based Cut Generation (SCG)* rule that achieves a finer balance between screening effectiveness and computational efficiency in presolving, and (2) a *Screening-based Cut Selection (SCS)* method, which provides theoretical guarantees for selecting SCG cuts with high screening potential to reduce total running time further. The main contributions are detailed below in correspondence with the paper's organization.

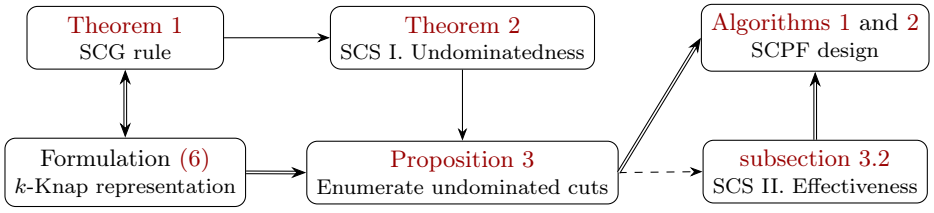
1. *section 2 introduces the SCG rule.* The proposed SCG rule provides three advantages over existing methods. First, the SCG rule balances computational efficiency with *screening ability*, which informally refers to the capacity to generate cuts. This is achieved by (i) simultaneously analyzing any specific support pattern, other than a single binary, and by (ii) designing efficient underestimations of the value function. In contrast to existing methods, our SCG rule eliminates the

need to exactly re-optimize relaxations over all desired patterns for support-set reduction. Second, SCG rule operates independently of the optimal solution $\widehat{\mathbf{z}}$ to relaxations (3). This independence provides the flexibility to generate SCG cuts from multiple supports in presolving. Third, from a conceptual perspective, SCG can be interpreted as a generalized, optimality-based cutting plane algorithm applied to support-space reduction, an application we believe to be novel in the context of presolving. Additionally, section 2 characterizes the *k-cardinality constrained Knapsack polytope (k-Knap)* and analyzes its connections to the enumeration of generated SCG cuts. We believe this analysis will provide an initial step toward assessing the closure of tightness of SCG cuts as an outer approximation of the optimal solution set to (1).

2. *section 3 develops the SCS method to improve computational efficiency with theoretical guarantees.* We present necessary and sufficient conditions for certifying minimal SCG tuples that generate *undominated* screening cuts. Using the properties of the k-Knap polytope, we translate the geometric/set properties of generated SCG cuts into algebraic rules implementable in the SCPF. Furthermore, subsection 3.2 introduces a criterion to quantify the "(potential) screening ability" of the generated cuts, allowing us to focus on specific types of *effective* undominated cuts with algorithmic guarantees. Integrating these two components, subsection 3.3 formalizes the complete SCPF algorithm and establishes its algorithmic soundness.

3. *Lastly, section 4 reports numerical experiments to validate our theoretical findings and demonstrate the practical advantages of the proposed SCPF.* The numerical results demonstrate that: (i) the SCPF outperforms existing baselines, particularly in challenging scenarios, *achieving a* $1.7\times$ *to* $3.0\times$ *acceleration in total runtime*; (ii) the multi-support SCPF variant exhibits strictly stronger screening ability than the single-support SCPF, leading to significant runtime reductions in some specific cases; and (iii) the SCS method is crucial in practice by ensuring that only undominated and effective cuts are generated in our SCPF, which substantially improves the performance of the subsequent BnB step.

The paper roadmap is presented below. Rectangular blocks represent the main contributions. A solid arrow ($A \to B$) indicates mathematical derivation; a double-stemmed arrow ($A \Rightarrow B$) indicates establishment; a double-headed & double-stemmed arrow ($A \Leftrightarrow B$) indicates equivalence; and a dashed arrow ($A \dashrightarrow B$) indicates motivation.



1.2 Notation

We use lowercase letters, e.g., x, boldface lowercase letters, e.g., $\mathbf{x}$, and boldface uppercase letters, e.g., $\mathbf{X}$, to represent scalars, vectors, and matrices, respectively. Let $\mathbf{1}$ denote the all-ones vector. For two integers a, b with $a \leq b$, we write $[\![a, b]\!] := \{a, \ldots, b\}$ and use $[\![b]\!] := \{1, \ldots, b\}$ as shorthand. Given two sets S, T, we use $|S|$ to denote the cardinality of set S and $S \setminus T$ to denote the set difference.

Given a vector $\mathbf{x} \in \mathbb{R}^d$, for any $i \in [\![d]\!]$, we write $(\mathbf{x})_i$ or x_i to denote the i-th component in $\mathbf{x}$ without further explanations. We use square brackets $(\mathbf{x})_{[k]}$ or $x_{[k]}$ to denote the k-th largest

component, and $\underline{\mathbf{x}} := x_{[d]}$ to be the smallest component (i.e., $\min_{i\in[\![d]\!]} x_i$). Let $\texttt{supp}(\mathbf{x})$ be the support set of $\mathbf{x}$, and $\texttt{Top}_p(\mathbf{x})$ be the index set with respect to top-p largest components in $\mathbf{x}$ (ties breaking lexicographically). For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Hadamard product is $\mathbf{x} \circ \mathbf{y}$, defined componentwise by $(\mathbf{x} \circ \mathbf{y})_i := x_i y_i$. For an index subset $S \subseteq [\![d]\!]$, let $\mathbf{x}_S \in \mathbb{R}^{|S|}$ be the subvector of $\mathbf{x}$ with its entries restricted on index subset S. We use $\mathbf{e}_S \in \mathbb{R}^d$ to denote the indicator vector of $S \subseteq [\![d]\!]$, with ones on S and zeros otherwise.

2 SCG Rule

As outlined in subsection 1.1, the SCG rule optimizes the trade-off between screening ability and computational efficiency. Unlike existing methods, our SCG rule (i) enhances screening ability by *jointly evaluating multiple binary variables*, and (ii) employs *efficient value function underestimations*, thereby eliminating the need to re-optimize relaxations for every support pattern. Formally, for disjoint index subsets $S, N \subseteq [\![d]\!]$ with $|S| \leq k$, SCG rule aims to certify the *optimality-based criterion* $v_{\texttt{relax}}(S, N) > v_{\texttt{ub}}$, where the restricted (optimal) value function is defined as: $v_{\texttt{relax}}(S, N) := \min_{(\boldsymbol{\beta},\mathbf{z})\in\mathbb{R}^d\times Z^k_{\texttt{relax}}} \mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{j=1}^d \beta_j^2/z_j$ s.t. $z_i = 1\ \forall i \in S,\ z_i = 0\ \forall i \in N$. Validating this criterion implies that any feasible support with $z_i = 1$ for all $i \in S$ and $z_i = 0$ for all $i \in N$ is strictly non-optimal. However, since the number of potential tuples (S, N) grows exponentially with respect to $|S|$ and $|N|$, *exhaustively re-optimizing $v_{\texttt{relax}}(S, N)$ for all tuples of interest is computationally intractable in presolving.* Thus, the central challenge is to develop a scalable underestimation strategy, which we present in Theorem 1.

We first introduce the definition of an *SCG-tuple*, which serves as the basis for the above underestimations. Let $\overline{\mathbf{z}} \in Z^k_{\texttt{relax}}$ be any primal feasible support. Consider the corresponding restricted Fenchel dual reformulation problem: $v_{\texttt{relax}}(\overline{\mathbf{z}}) := \min_{\boldsymbol{\beta}} \max_{\mathbf{p}}\ \mathcal{L}(\boldsymbol{\beta}) + \gamma \sum_{i=1}^d \left(p_i\beta_i - \frac{p_i^2}{4}\overline{z}_i\right)$. Let $\left(\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}}), \widehat{\mathbf{p}}(\overline{\mathbf{z}})\right)$ be an optimal primal-dual pair to the above problem, and set $\widehat{\mathbf{w}}(\overline{\mathbf{z}}) := \widehat{\mathbf{p}}(\overline{\mathbf{z}}) \circ \widehat{\mathbf{p}}(\overline{\mathbf{z}})$.

Definition 1 (SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$)**.** *Assume there is no tie in $\widehat{\mathbf{w}} = \widehat{\mathbf{w}}(\overline{\mathbf{z}})$. For any two disjoint index subsets $S, N \subseteq [\![d]\!]$ with $|S| \leq k$, we define the quartet $(\overline{\mathbf{z}}, S, N, C)$ as an SCG-tuple, where $C = C(\widehat{\mathbf{w}}(\overline{\mathbf{z}}), S, N) := \left\{i \in R \mid \widehat{w}_i \geq (\widehat{\mathbf{w}}_R)_{[c]}\right\}$ with $R := [\![d]\!] \setminus (S \cup N)$ and $c := \min\{k - |S|,\ |R|\}$. Specifically, when $c = 0$, we set $C := \emptyset$.*

Notably, set C is uniquely determined by vector $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$ and sets S, N as the "best" possible set that, combining with S, forms a support set of size k to (1).

We are now poised to present the SCG rule in Theorem 1.

Theorem 1. *Assume there is no tie in $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$. Given a SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$, if the following optimality-based criterion:*

$$v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}, S, N, C) :=\ \mathcal{L}(\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})) + \gamma\langle\widehat{\mathbf{p}}(\overline{\mathbf{z}}), \widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})\rangle - \frac{\gamma}{4}\sum_{i\in(S\cup C)} (\widehat{\mathbf{w}}(\overline{\mathbf{z}}))_i\ >\ v_{\texttt{ub}} \tag{4}$$

holds, the proposed SCG rule certifies the optimality-based criteria $v_{\texttt{relax}}(S, N) > v_{\texttt{ub}}$ and equivalently, the following screening-based cutting plane $\sum_{i\in S} z_i + \sum_{i\in N}(1 - z_i) \leq |S| + |N| - 1$ eliminates some non-optimal but remains all optimal solutions for (1). Additionally, we call a SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$ a valid SCG-tuple if $(\overline{\mathbf{z}}, S, N, C)$ also ensures the above inequality (4).

Proof. Consider the Fenchel dual reformulation of $v_{\texttt{relax}}(S,N)$:

$$\begin{aligned} v_{\texttt{relax}}(S,N) &= \min_{(\boldsymbol{\beta},\mathbf{z})\in\mathbb{R}^d\times Z^k_{\texttt{relax}}(S,N)} \max_{\mathbf{p}} \ \mathcal{L}(\boldsymbol{\beta}) + \gamma\sum_{i=1}^d (p_i\beta_i - \tfrac{p_i^2}{4}z_i) \\ &= \max_{\mathbf{p}} \min_{(\boldsymbol{\beta},\mathbf{z})\in\mathbb{R}^d\times Z^k_{\texttt{relax}}(S,N)} \ \mathcal{L}(\boldsymbol{\beta}) + \gamma\sum_{i=1}^d (p_i\beta_i - \tfrac{p_i^2}{4}z_i) \end{aligned}$$

where $Z^k_{\texttt{relax}}(S,N) := Z^k_{\texttt{relax}} \cap \{\mathbf{z}\in\mathbb{R}^d \mid z_i = 1\ \forall i\in S,\ z_j = 0\ \forall j\in N\}$ and the second equality holds by the strong duality based on linearity constraint qualification (LCQ). A lower bound of $v_{\texttt{relax}}(S,N)$ can be obtained by plugging in $\widehat{\mathbf{p}}(\overline{\mathbf{z}})$ to the outer maximization $v_{\texttt{relax}}(S,N) \geq v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}},S,N,C)$, where $v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}},S,N,C) := \min_{\boldsymbol{\beta}} f_1(\boldsymbol{\beta};\widehat{\mathbf{p}}(\overline{\mathbf{z}})) - \max_{\mathbf{z}\in Z^k_{\texttt{relax}}(S,N)} f_2(\mathbf{z};\widehat{\mathbf{p}}(\overline{\mathbf{z}}))$ can be represented as the difference of two subproblems with $f_1(\boldsymbol{\beta};\widehat{\mathbf{p}}(\overline{\mathbf{z}})) := \mathcal{L}(\boldsymbol{\beta}) + \gamma\langle\widehat{\mathbf{p}}(\overline{\mathbf{z}}),\boldsymbol{\beta}\rangle$ and $f_2(\mathbf{z};\widehat{\mathbf{p}}(\overline{\mathbf{z}})) := \frac{\gamma}{4}\langle\widehat{\mathbf{w}}(\overline{\mathbf{z}}),\mathbf{z}\rangle$, respectively.

To minimize $f_1(\boldsymbol{\beta};\widehat{\mathbf{p}}(\overline{\mathbf{z}}))$ over $\boldsymbol{\beta}$, it is easy to verify that $\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})$ is an optimal solution due to $\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})$ and $\widehat{\mathbf{p}}(\overline{\mathbf{z}})$ are optimal primal-dual pairs to $v_{\texttt{relax}}(\overline{\mathbf{z}})$. To minimize $f_2(\mathbf{z};\widehat{\mathbf{p}}(\overline{\mathbf{z}}))$, we need to choose k largest components in $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$ while maintaining the feasibility requirement $\mathbf{z}\in Z^k_{\texttt{relax}}(S,N)$. By definition of SCG-tuple $(\overline{\mathbf{z}},S,N,C)$, we have $z_i = 1$ for $i\in(S\cup C)$ at optimality. Therefore, we have $v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}},S,N,C) = \mathcal{L}(\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})) + \gamma\langle\widehat{\mathbf{p}}(\overline{\mathbf{z}}),\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})\rangle - \frac{\gamma}{4}\sum_{i\in(S\cup C)}(\widehat{\mathbf{w}}(\overline{\mathbf{z}}))_i$. If the optimality-based criterion (4) holds, we have $v^{\texttt{lb}}_{\texttt{relax}}(S,N) > v_{\texttt{ub}}$, thus any feasible solution of (1) with $z_i = 1$ for $i\in S$ and $z_j = 0$ for $j\in N$ cannot be optimal, and we can add screening-based cut $\sum_{i\in S} z_i + \sum_{i\in N}(1-z_i) \leq |S| + |N| - 1$ to remove all such feasible points but maintain all optimal solutions. □

One can relax the condition "assume there is no tie in $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$" by breaking ties lexicographically for the set C. Furthermore, *the proof of Theorem 1 extends to general sparsity-constrained convex programming beyond the quadratic setting, thereby broadening the applicability of Theorem 1.*

The SCG rule achieves a favorable trade-off between screening ability and computational efficiency. By simultaneously constraining multiple binary variables, our SCG rule strengthens screening ability compared to single-variable fixing techniques (Atamturk and Gomez, 2020; Deza and Atamturk, 2022; Yang, 2025; Yamagishi et al., 2026). Moreover, unlike methods that require re-optimizing relaxations for every desired subset pair (S,N) (Li et al., 2024; Jiang and Xie, 2025; D'Ambrosio et al., 2026), *Theorem 1 efficiently estimates lower bounds of $v_{\texttt{relax}}(S,N)$ based on a single relaxed support $\overline{\mathbf{z}}\in Z^k_{\texttt{relax}}$*, which significantly reduces the computational burden from all interested pairs of subsets (S,N) to one or several chosen relaxed supports, as discussed below.

Since Theorem 1 does not specify the choice of support $\overline{\mathbf{z}}$, a natural extension is to generate cuts based on *a sequence of deserved "relaxed" supports* $\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^T \in Z^k_{\texttt{relax}}$. If the *multi-support optimality-based criterion*

$$\max_{t=1}^{T}\ v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}^{(t)},S,N,C^{(t)}) \ > \ v_{\texttt{ub}} \tag{5}$$

holds, the corresponding SCG cut $\sum_{i\in S} z_i + \sum_{i\in N}(1-z_i) \leq |S| + |N| - 1$ is valid for (1). Numerical experiments further indicate that, with a proper choice of multiple supports, the resulting SCG rule provides a strictly stronger screening ability on most challenging settings, and leads to significant runtime reductions on some specific instances, see Figure 3 in section 4. Note that a very recent work by Yamagishi et al. (2026) introduces a dual-path fixing method, which performs single-variable fixing using a path of dual feasible solutions generated from the root-node relaxation of MILPs. In contrast, our multi-support SCG method employs a fundamentally different strategy. Rather than relying solely on root-node information, we construct dual feasible solutions based on a sequence of "relaxed" supports $\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^T$ selected for their *diversity* and *representativeness*. This approach essentially incorporates optimal relaxation information from targeted child nodes, thereby ensuring

a greater possibility to satisfy the multi-support optimality-based criterion (5). Further details on multi-support SCG are provided in Appendix B.1.

For researchers of independent interest, the criterion in (4) can be interpreted as a generalized, optimality-based cutting plane algorithm applied to support-space reduction. This is obtained by reformulating (4) as:

$$v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}, S, N, C) = \; v_{\texttt{relax}}(\overline{\mathbf{z}}) + \frac{\gamma}{4}\langle -\widehat{\mathbf{w}}(\overline{\mathbf{z}}), \boldsymbol{e}_{S\cup C} - \overline{\mathbf{z}}\rangle$$

with $v_{\texttt{relax}}(\overline{\mathbf{z}}) = \mathcal{L}(\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})) + \gamma\langle\widehat{\mathbf{p}}(\overline{\mathbf{z}}), \widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})\rangle - \frac{\gamma}{4}\langle\widehat{\mathbf{w}}(\overline{\mathbf{z}}), \overline{\mathbf{z}}\rangle$. While conceptually this framework aligns with optimality-based cutting plane algorithms or outer approximations (Tawarmalani and Sahinidis, 2004; Bertsimas and Cory-Wright, 2022; Wei and Küçükyavuz, 2024; Bui et al., 2025) and Lagrangian cut generation (Zou et al., 2018; Chen and Luedtke, 2022; Deng and Xie, 2024), our method distinguishes itself through its efficiency. Specifically, the SCG rule leverages dual multipliers from a *single* relaxed support $\overline{\mathbf{z}} \in Z^k_{\texttt{relax}}$ to underestimate value functions across numerous subset pairs, without expensive re-optimization. *To the best of our knowledge, it is the first time that an efficient cutting plane algorithm is applied specifically to support-space reduction in presolving.*

We summarize the hierarchy of screening-based methods in the following diagram:

$$\underbrace{\underset{[\text{exact}]}{v^*(S,N)} \;\geq\; \underset{[26;\,24;\,14]}{v_{\texttt{relax}}(S,N)}}_{\text{exact optimize over all } (S,N)} \;\vdots\; \underbrace{\underset{[\text{Theorem 1};\,34^*]}{v^{\texttt{lb}}_{\texttt{relax}}(\{\overline{\mathbf{z}}^{(t)}\}, S, N, \{C^{(t)}\})} \;\geq\; \underset{[3;\,17]}{v^{\texttt{lb}}_{\texttt{relax}}(\widehat{\mathbf{z}}, \overline{z}_i)}}_{\text{underestimate by supports (cutting plane algo)}} \;\overset{\text{opt criteria}}{\geq}\; v_{\texttt{ub}} \geq v^*$$

*: The dual-path fixing proposed in Yamagishi et al. (2026) only applies to single-variable case, i.e., $|S \cup N| = 1$.

Moving from left to right, the tightness of the lower bound decreases while computational efficiency increases. The green dotted line separates methods based on exact re-optimization (left), which are generally intractable for presolving, from our proposed estimation methods (right), which leverage dual multipliers over specific supports to ensure efficiency.

We now give an equivalent condition for valid SCG-tuples, formalized in Corollary 1. These conditions serve as a foundation for Theorem 2. The proof is provided in Appendix A.1.

Corollary 1. *Given some $\overline{\mathbf{z}} \in Z^k_{\texttt{relax}}$, assume there is no tie in $\widehat{\mathbf{w}} = \widehat{\mathbf{w}}(\overline{\mathbf{z}})$. We have $(\overline{\mathbf{z}}, S, N, C)$ is a valid SCG-tuple if and only 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 $\mathcal{I}(C) \subseteq (S \cup N \cup C)$, where $\mathcal{I}(C) := \{i \in [\![d]\!] : \widehat{w}_i \geq \underline{\widehat{\mathbf{w}}_C}\}$;

(3) $v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}, S, N, C) > v_{\texttt{ub}}$.

To facilitate the enumeration and selection of SCG cuts (as detailed in section 3), we reformulate the validity condition (4) as a packing constraint $\sum_{i\in(S\cup C)}(\widehat{\mathbf{w}}(\overline{\mathbf{z}}))_i < q(\overline{\mathbf{z}})$ with $q(\overline{\mathbf{z}}) := \frac{4}{\gamma}\left(\mathcal{L}(\widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})) + \gamma\langle\widehat{\mathbf{p}}(\overline{\mathbf{z}}), \widehat{\boldsymbol{\beta}}(\overline{\mathbf{z}})\rangle - v_{\texttt{ub}}\right)$. Then, define *k-cardinality-constrained binary knapsack polytope* $K(\overline{\mathbf{z}})$ as

$$K(\overline{\mathbf{z}}) := \left\{\mathbf{x} \in \{0,1\}^d \;\middle|\; \textstyle\sum_{i=1}^d (\widehat{\mathbf{w}}(\overline{\mathbf{z}}))_i \cdot x_i < q(\overline{\mathbf{z}}),\; \|\mathbf{x}\|_0 \leq k\right\}. \tag{6}$$

There is a direct connection between valid SCG-tuples and the set $K(\overline{\mathbf{z}})$. Specifically, every valid SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$ maps to a feasible point $\mathbf{x} \in K(\overline{\mathbf{z}})$ with $\texttt{supp}(\mathbf{x}) = S \cup C$. Conversely,

assuming $K(\overline{\mathbf{z}}) \neq \emptyset$ and $q(\overline{\mathbf{z}}) > 0$, every $\mathbf{x} \in K(\overline{\mathbf{z}})$ generates at least $2^{|\mathtt{supp}(\mathbf{x})|}$ valid SCG-tuples by setting $N = [\![d]\!] \setminus \mathtt{supp}(\mathbf{x})$ and partitioning $\mathtt{supp}(\mathbf{x})$ into disjoint sets S and C. Based on the above connection, incorporating all corresponding SCG cuts to the original problem (1) is computationally impractical. Consequently, section 3 utilizes the structure of $K(\overline{\mathbf{z}})$ to systematically enumerate and select only those cuts that come with guarantees on their effectiveness.

3 SCS Method

Building upon the SCG rule, this section studies the enumeration and selection methods of screening cuts (see subsection 3.1 and subsection 3.2) to enhance SCG performance and reduce total running time. These two parts are then unified to establish our Screening-based Cut Presolving Framework (SCPF), as detailed in subsection 3.3. To streamline the exposition and without loss of generality, we adopt the following assumption throughout this section.

Assumption 1. *For any fixed $\overline{\mathbf{z}} \in Z^k_{\mathtt{relax}}$, (i) there is no tie in $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$; and (ii) components in $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$ are in a decreasing order, i.e., $\widehat{\mathbf{w}}(\overline{\mathbf{z}})_1 > \cdots > \widehat{\mathbf{w}}(\overline{\mathbf{z}})_d$.*

3.1 Selection by Undominatedness

We begin by establishing a proposition regarding cut domination.

Proposition 2. *Given any $\overline{\mathbf{z}} \in Z^k_{\text{relax}}$, suppose $(\overline{\mathbf{z}}, S, N, C)$ is a valid SCG-tuple. We have any SCG-tuple $(\overline{\mathbf{z}}, S', N', C')$ with $S \subseteq S'$ and $N \subseteq N'$ is also valid. Moreover, the cut generated by $(\overline{\mathbf{z}}, S', N', C')$ is dominated by the cut generated by $(\overline{\mathbf{z}}, S, N, C)$.*

The proof is given in Appendix A.2. As an illustrative example, consider an index $i \in [\![d]\!]$ for which the SCG rule generates $z_i = 0$ based on the valid SCG-tuple $\big(\overline{\mathbf{z}}, \{i\}, \emptyset, \mathtt{Top}_{k-1}(\widehat{\mathbf{w}}(\overline{\mathbf{z}}))\big)$. Then all cuts of the form $z_i + z_j \leq 1$ with $j \in [\![d]\!] \setminus \{i\}$ are also technically valid. However, Proposition 2 ensures that these cuts of form $z_i + z_j \leq 1,\ \forall\, j \in [\![d]\!] \setminus \{i\}$ are redundant and need not be added to (1). Motivated by Proposition 2, we introduce the definition of a *minimal SCG-tuple* to avoid dominated cuts and enhance computational efficiency.

Definition 2. *We call a valid SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$ a* minimal SCG-tuple *if the following two conditions hold simultaneously:*

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

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

We now establish necessary and sufficient algebraic conditions for the minimality of a valid SCG-tuple.

Theorem 2. *Suppose Assumption 1 holds. Given any $\overline{\mathbf{z}} \in Z^k_{\mathtt{relax}}$, for a valid SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$, we have:*

- ***Case 1.*** *If $|N| > d - k$, then $(\overline{\mathbf{z}}, S, N, C)$ is minimal if and only if $S = \emptyset$.*
- ***Case 2.*** *If $|N| \leq d - k$ and $C = \emptyset$, then $(\overline{\mathbf{z}}, S, N, C)$ is minimal if and only if $N = \emptyset$ and $\max_{i \in S} (\widehat{\mathbf{w}})_i < (\widehat{\mathbf{w}})_{[1]}$.*

- ***Case 3.*** *If $|N| \le d-k$ and $C \neq \emptyset$, then $(\overline{\mathbf{z}}, S, N, C)$ is minimal if and only if the following two conditions hold simultaneously:*

  (1) $N = \emptyset$, or $\underline{\widehat{\mathbf{w}}_N} \ge \underline{\widehat{\mathbf{w}}_C}$;

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

Proof. To establish minimality of a valid SCG-tuple, we verify that every single-element modification breaks validity by checking the conditions in Corollary 1. Our approach primarily relies on proof by contradiction.

Proof of Case 1. Assume $|N| > d-k$. Thus $S \cup N \cup C = [\![d]\!]$ and $|S \cup C| < k$.

($\Leftarrow$): If $S = \emptyset$, it suffices to show that $(\overline{\mathbf{z}}, \emptyset, N \setminus \{i\}, C)$ is not valid for any $i \in N$. It violates (1) in Corollary 1 where $|C| = d - |N| \neq \min\{k, d-|N|+1\}$.

($\Rightarrow$): Prove by contradiction. Suppose $S \neq \emptyset$. Then for any $i \in S$, $(\overline{\mathbf{z}}, S \setminus \{i\}, N, C \cup \{i\})$ is valid SCG-tuple by verifying Corollary 1. This contradicts minimality, so $S = \emptyset$.

Proof of Case 2. Assume $|N| \le d-k$ and $C = \emptyset$. Thus $|S| = k$. Denote $i^* = \arg\max_i (\widehat{\mathbf{w}})_i$.

($\Leftarrow$): If $N = \emptyset$ and $i^* \notin S$, it suffices to show that for any $j \in S$, $(\overline{\mathbf{z}}, S \setminus \{j\}, \emptyset, \{j\})$ is not valid. It violates (2) in Corollary 1 where $\widehat{w}_{i^*} > \widehat{w}_j$ but $i^* \notin S$.

($\Rightarrow$): If $(\overline{\mathbf{z}}, S, N, \emptyset)$ is minimal, we first show $N = \emptyset$ and then $i^* \notin S$.

- Prove by contradiction. If $N \neq \emptyset$, for any $i \in N$, $(\overline{\mathbf{z}}, S, N \setminus \{i\}, \emptyset)$ is valid by verifying Corollary 1, contradicting minimality. Thus $N = \emptyset$.

- Now we have $(\overline{\mathbf{z}}, S, \emptyset, \emptyset)$ is minimal. Prove by contradiction. If $i^* \in S$, consider SCG-tuple $(\overline{\mathbf{z}}, S \setminus \{i^*\}, \emptyset, \{i^*\})$, which is valid by verifying Corollary 1, contradicting minimality. Thus $i^* \notin S$.

Proof of Case 3. Assume $|N| \le d-k$ and $C \neq \emptyset$. Thus $|S| + |C| = k$. We prove the case in two parts: first establishing the equivalence on minimality conditions for N, then conclude the proof.

Minimality on set N. We first establish the equivalence of condition (1) in Definition 2 with condition (1) in Theorem 2. Without loss of generality, assume $N \neq \emptyset$.

($\Leftarrow$): If $\underline{\widehat{\mathbf{w}}_N} \ge \underline{\widehat{\mathbf{w}}_C}$, for any $i \in N$, the SCG-tuple $(\overline{\mathbf{z}}, S, N \setminus \{i\}, C)$ is not valid since $i \in \mathcal{I}(C)$ but $i \notin (S \cup (N \setminus \{i\}) \cup C)$, violating (2) in Corollary 1.

($\Rightarrow$): Prove by contradiction. Suppose there exists $j \in N$ such that $\widehat{w}_j < \underline{\widehat{\mathbf{w}}_C}$. Thus $j \notin \mathcal{I}(C)$. Consider SCG-tuple $(\overline{\mathbf{z}}, S, N \setminus \{j\}, C)$, which is valid by verifying Corollary 1. Contradiction with minimality on N.

Minimality on set S. To complete the proof, it is sufficient to show that the minimality conditions on S is equivalent between Definition 2 and Theorem 2, with $N = \emptyset$, or $\underline{\widehat{\mathbf{w}}_N} \ge \underline{\widehat{\mathbf{w}}_C}$ holds. For the nontrivial case, assume $S \neq \emptyset$.

($\Leftarrow$): If such j exists, for any $i' \in S$, consider SCG-tuple $(\overline{\mathbf{z}}, S \setminus \{i'\}, N, C \cup \{i'\})$. It is not valid since we have $\widehat{w}_j > \max_{i \in S} (\widehat{\mathbf{w}})_i \ge \widehat{w}_{i'}$, and thus $\widehat{w}_j > \underline{\widehat{\mathbf{w}}_{C \cup \{i'\}}}$, but $j \notin (S \setminus \{i'\}) \cup N \cup (C \cup \{i'\})$, violating (2) in Corollary 1.

($\Rightarrow$): There are two steps to prove such j exists.

- Suppose there exists some $i' \in S$ such that $\widehat{w}_{i'} \ge \underline{\widehat{\mathbf{w}}_C}$. Consider SCG-tuple $(\overline{\mathbf{z}}, S \setminus \{i'\}, N, C \cup \{i'\})$. Then $\mathcal{I}(C \cup \{i'\}) = \mathcal{I}(C) \subseteq S \cup N \cup C$, thus it is valid by verifying Corollary 1. Contradiction, and we have $\max_{i \in S} \widehat{w}_i < \underline{\widehat{\mathbf{w}}_C}$.

- Since we know $N = \emptyset$, or $\underline{\widehat{\mathbf{w}}_N} \geq \underline{\widehat{\mathbf{w}}_C}$, it suffices to prove that there exists some $j \notin (S \cup C)$ such that $\underline{\widehat{\mathbf{w}}_C} > \widehat{w}_j > \max_{i \in S} \widehat{w}_i$. Suppose not. Denote $i' = \operatorname{argmax}_{i \in S} \widehat{w}_i$. Consider SCG-tuple $(\overline{\mathbf{z}}, S \setminus \{i'\}, N, C \cup \{i'\})$. Then $\mathcal{I}(C \cup \{i'\}) = \mathcal{I}(C) \cup \{i'\} \subseteq S \cup N \cup C$. It is valid by verifying Corollary 1. Contradiction, and we are done.

□

Note that the validity and minimality of an SCG-tuple depend on the chosen support $\overline{\mathbf{z}} \in Z^k_{\texttt{relax}}$. As illustrated in Appendix D.2, the multi-support SCG rule can generate minimal screening cuts that are inaccessible to the single-support SCG rule.

To analyze the connection between any point $\mathbf{x} \in K(\overline{\mathbf{z}})$ and corresponding minimal SCG-tuples (based on Theorem 2 and (6)), we introduce the following definitions regarding index partitioning.

Definition 3 (Consecutive index set & Minimum consecutive partition)**.** *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 $i - 1 \in A$.*

The minimum consecutive partition *of an index set is the unique partition that decomposes the set into the minimum number of disjoint consecutive index sets.*

For example, there are several ways to partition an index set $I = \{2, 3, 5, 6, 9\}$ into disjoint consecutive index sets, such as $\{2, 3\} \cup \{5, 6\} \cup \{9\}$ or $\{2\} \cup \{3\} \cup \{5, 6\} \cup \{9\}$ or $\{2\} \cup \{3\} \cup \{5\} \cup \{6\} \cup \{9\}$, respectively. Among all above partitions, the minimum consecutive partition of I is $\{2, 3\} \cup \{5, 6\} \cup \{9\}$. Recall that under Assumption 1, the components of $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$ are strictly decreasing. We define the complementary index set as follows.

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

Using these definitions, consider any non-trivial point $\mathbf{0} \neq \mathbf{x} \in K(\overline{\mathbf{z}})$. Let the minimum consecutive partition of its support be $\texttt{supp}(\mathbf{x}) = \bigcup_{j=1}^{m} A_j(\mathbf{x})$ for some $m \leq k$, where each consecutive set is written as $A_j(\mathbf{x}) := \{i_1^{(j)}, \ldots, i_{|A_j|}^{(j)}\}$. Then the minimum consecutive partition of the complementary set $\texttt{comp}(\texttt{supp}(\mathbf{x}))$ can be expressed as $\cup_{j=1}^{m} B_j(\mathbf{x})$, where $B_j(\mathbf{x}) := \{i_{|A_{j-1}|}^{(j-1)} + 1, \ldots, i_1^{(j)} - 1\} \quad \forall j = 1, \ldots, m$, with $i_{|A_0|}^{(0)} = 0$. Note that $B_1(\mathbf{x})$ may be empty. This structure allows us to characterize minimal SCG-tuples based on feasible points in $K(\overline{\mathbf{z}})$.

Proposition 3. *Suppose Assumption 1 holds. For any non-trivial $\mathbf{x} \in K(\overline{\mathbf{z}})$ with $\overline{\mathbf{z}} \in Z^k_{\texttt{relax}}$, let the minimum consecutive partitions of $\texttt{supp}(\mathbf{x})$ and $\texttt{comp}(\texttt{supp}(\mathbf{x}))$ be $\texttt{supp}(\mathbf{x}) = \cup_{j=1}^{m} A_j(\mathbf{x})$ and $\texttt{comp}(\texttt{supp}(\mathbf{x})) = \cup_{j=1}^{m} B_j(\mathbf{x})$ with some $m \leq k$. Then, one can construct minimal SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$ with $S \cup C = \texttt{supp}(\mathbf{x})$ as follows:*

(1) If $|\texttt{supp}(\mathbf{x})| < k$, then

$$(\overline{\mathbf{z}}, S, N, C) = (\overline{\mathbf{z}}, \emptyset,\ [\![d]\!] \setminus \texttt{supp}(\mathbf{x}),\ \texttt{supp}(\mathbf{x}))$$

is a minimal SCG-tuple.

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

$$(\overline{\mathbf{z}}, S, N, C) = \left(\overline{\mathbf{z}}, \cup_{j=\ell+1}^{m} A_j(\mathbf{x}),\ \cup_{j=1}^{\ell} B_j(\mathbf{x}),\ \cup_{j=1}^{\ell} A_j(\mathbf{x})\right)$$

is minimal. For boundary cases $\ell = 0$ and $\ell = m$, define

$$(\overline{\mathbf{z}}, S_1, N_1, C_1) = (\overline{\mathbf{z}}, \mathit{supp}(\mathbf{x}), \emptyset, \emptyset), \quad \text{and}$$
$$(\overline{\mathbf{z}}, S_2, N_2, C_2) = (\overline{\mathbf{z}}, \emptyset, \mathit{comp}(\mathit{supp}(\mathbf{x})), \mathit{supp}(\mathbf{x})),$$

respectively. The SCG-tuple $(\overline{\mathbf{z}}, S_2, N_2, C_2)$ is always minimal. Moreover, if $B_1(\mathbf{x}) \neq \emptyset$, the SCG-tuple $(\overline{\mathbf{z}}, S_1, N_1, C_1)$ is minimal as well.

The proof is presented in Appendix A.3. Based on this construction, one can generate at most $m+1$ minimal SCG-tuples (and corresponding screening cuts) from any non-trivial point $\mathbf{x} \in K(\overline{\mathbf{z}})$.

3.2 Selection by Effectiveness

However, undominatedness is a necessary but insufficient condition to ensure computational effectiveness for cut selection (see Appendix D.1 for details). Consider a minimal SCG-tuple $(\overline{\mathbf{z}}, \mathtt{supp}(\mathbf{x}), \emptyset, \emptyset)$ characterized in Proposition 3, where $B_1(\mathbf{x}) \neq \emptyset$. The resulting cut, $\sum_{i \in \mathtt{supp}(\mathbf{x})} z_i \leq |\mathtt{supp}(\mathbf{x})| - 1$, eliminates only a single feasible support from (1), which offers negligible reduction in support space. To address this issue, we introduce the concept of *(potential) screening ability*, which quantifies the maximum number of feasible supports eliminated by a cut.

Definition 5. *Given a valid SCG-tuple $(\overline{\mathbf{z}}, S, N, C)$ 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 *refers to the number of feasible solutions eliminated by incorporating the above cut into the original problem (1), which is $\sum_{i=0}^{|C|} \binom{d - |S \cup N|}{i}$.*

The proof of this bound is given in Appendix A.4. We distinguish Definition 5 from the *screening ability* of an algorithm, which informally refers to the algorithm's capacity to generate cuts.

It follows from Definition 5 that minimizing $|S \cup N|$ while maximizing $|C|$ enhances potential screening ability. In practice, however, cuts are incorporated sequentially and collectively. The number of additional non-optimal supports removed by a *specific* cut, based on the set of previously selected cuts, is termed by *(marginal) screening ability*. Since marginal screening ability is state-dependent and difficult to estimate, we use potential screening ability as a tractable performance metric in section 4.

To simplify the analysis, we focus on two specific types of cuts[2]: *inclusive cuts* and *exclusive cuts*, which enforce that at least one candidate binary variable is included in or excluded from the optimal support, respectively. Recall that we proceed under Assumption 1.

Inclusive cuts. The first type of screening cuts takes the form: $\sum_{i \in N} z_i \geq 1$. These inclusive cuts ensure that any optimal solution to (1) must contain at least one index from the set N. Applying Proposition 3 with $S = \emptyset$ and $C = \mathtt{supp}(\mathbf{x})$ for some $\mathbf{x} \in K(\overline{\mathbf{z}})$, we derive the corresponding minimal SCG-tuple:

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

When $|\mathtt{supp}(\mathbf{x})| < k$, the cardinality of N becomes $d - |\mathtt{supp}(\mathbf{x})|$, which is typically large. As noted in Definition 5, a larger $|N|$ significantly weakens the potential screening ability. Therefore, we restrict

[2]Preliminary experiments also considered *one-dominate-another cuts* of the form $z_i \leq z_j$, which, however, were less effective in practice: they offer weaker potential screening ability than inclusive cuts and are more difficult to generate than exclusive cuts in challenging phases. Thus, this type of cut is omitted in the manuscript.

our focus to the subset of full-cardinality supports, defined as $\mathcal{T}(\overline{\mathbf{z}}) := \{\mathbf{x} \in K(\overline{\mathbf{z}}) \mid |\mathtt{supp}(\mathbf{x})| = k\}$. The corresponding minimal SCG-tuples are $(\overline{\mathbf{z}}, \emptyset, \mathtt{comp}(\mathtt{supp}(\mathbf{x})), \mathtt{supp}(\mathbf{x}))$.

To analyze these inclusive cuts efficiently, we partition $\mathcal{T}(\overline{\mathbf{z}})$ based on the largest index in support $\mathtt{supp}(\mathbf{x})$ as $\mathcal{T}(\overline{\mathbf{z}}) = \cup_{i=1}^{d} \mathcal{T}_i(\overline{\mathbf{z}})$, where

$$\mathcal{T}_i(\overline{\mathbf{z}}) := \{\mathbf{x} \in \mathcal{T}(\overline{\mathbf{z}}) \;\big|\; \mathtt{supp}(\mathbf{x}) \subseteq [\![i]\!], i \in \mathtt{supp}(\mathbf{x})\} \quad . \tag{7}$$

Under Assumption 1 (descending weights), it follows that $\mathcal{T}_i(\overline{\mathbf{z}}) = \emptyset$ for all $i \leq k$. Specifically, for $i = k$, the only candidate is $\mathtt{supp}(\mathbf{x}) = [\![k]\!]$. However, the optimality-based criterion (4) leads $\langle \widehat{\mathbf{w}}(\overline{\mathbf{z}}), \mathbf{x} \rangle \geq q(\overline{\mathbf{z}})$, which contradicts $\mathbf{x} \in K(\overline{\mathbf{z}})$. Thus, we consider only the range $i \in \{k+1, \ldots, d\}$ with $\mathcal{T}(\overline{\mathbf{z}}) = \cup_{i=k+1}^{d} \mathcal{T}_i(\overline{\mathbf{z}})$. The following Claim 1 characterizes the non-emptiness of these subsets.

Claim 1. *Given $\mathcal{T}_i(\overline{\mathbf{z}})$ with $i \in [\![k+1, d]\!]$, $\mathcal{T}_i(\overline{\mathbf{z}}) \neq \emptyset$ if and only if there exists some $\mathbf{x}' \in K(\overline{\mathbf{z}})$ with $\mathtt{supp}(\mathbf{x}') = [\![i-(k-1), i]\!]$. Moreover, if $\mathcal{T}_i(\overline{\mathbf{z}}) \neq \emptyset$, we have $\mathcal{T}_j(\overline{\mathbf{z}}) \neq \emptyset$ for all $j = i+1, \ldots, d$.*

The proof is provided in Appendix A.5. This monotonicity property implies that identifying all non-empty sets $\mathcal{T}_i(\overline{\mathbf{z}})$ is equivalent to finding the smallest index $s \in [\![k+1, d]\!]$ such that $\mathcal{T}_s(\overline{\mathbf{z}}) \neq \emptyset$. This equivalence is applied in designing Algorithm 1.

Exclusive cuts. The second type of screening cuts takes the form $\sum_{i \in S} z_i \leq |S| - 1$. These exclusive cuts ensure that the indices in S cannot simultaneously constitute a subset of the support in any optimal solution to (1).

Recall the minimal consecutive partitions of $\mathtt{supp}(\mathbf{x})$ and $\mathtt{comp}(\mathtt{supp}(\mathbf{x}))$ are $\mathtt{supp}(\mathbf{x}) = \cup_{j=1}^{m} A_j(\mathbf{x})$ and $\mathtt{comp}(\mathtt{supp}(\mathbf{x})) = \cup_{j=1}^{m} B_j(\mathbf{x})$. According to Proposition 3, for any $\mathbf{x} \in K(\overline{\mathbf{z}})$ with $|\mathtt{supp}(\mathbf{x})| = k$, the resulting minimal SCG-tuples are

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

When $B_1(\mathbf{x}) \neq \emptyset$, the corresponding exclusive cut eliminates only a single feasible solution, which gives negligible screening ability. Therefore, the exclusive cut generation method restricts attention to the subset with $B_1(\mathbf{x}) = \emptyset$ (implying $1 \in \mathtt{supp}(\mathbf{x})$), which is defined as

$$\mathcal{T}'(\overline{\mathbf{z}}) := \{\mathbf{x} \in \mathcal{T}(\overline{\mathbf{z}}) \;\big|\; B_1(\mathbf{x}) = \emptyset\} = \{\mathbf{x} \in \mathcal{T}(\overline{\mathbf{z}}) \;\big|\; 1 \in \mathtt{supp}(\mathbf{x})\} \quad .$$

The minimal SCG-tuples for the above set are $(\overline{\mathbf{z}}, \cup_{j=2}^{m} A_j(\mathbf{x}),\ \emptyset,\ A_1(\mathbf{x}))$, which yields minimal cuts with $|\sum_{j=2}^{m} A_j(\mathbf{x})| = k - |A_1(\mathbf{x})|$ binaries. Similarly, we partition $\mathcal{T}'(\overline{\mathbf{z}})$ based on the first consecutive index set $A_1(\mathbf{x})$ as $\mathcal{T}'(\overline{\mathbf{z}}) = \cup_{i=1}^{d} \mathcal{T}'_i(\overline{\mathbf{z}})$, where

$$\mathcal{T}'_i(\overline{\mathbf{z}}) := \{\mathbf{x} \in \mathcal{T}(\overline{\mathbf{z}}) \;\big|\; A_1(\mathbf{x}) = [\![i]\!]\} \tag{8}$$

Similarly, under Assumption 1, it follows that $\mathcal{T}'_i(\overline{\mathbf{z}}) = \emptyset$ for all $i \in \{k, \ldots, d\}$. Thus, we consider only the range $i \in \{1, \ldots, k-1\}$. Claim 2 provides a necessary and sufficient condition to certify the non-emptiness of $\mathcal{T}'_i(\overline{\mathbf{z}})$.

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

Algorithm 1: SCPF($K(\bar{\mathbf{z}}), \sharp_{\texttt{max}}, \sharp_{\texttt{len}}$)

Input : knapsack polytope $K(\bar{\mathbf{z}})$, max number of cuts $\sharp_{\texttt{max}}$, max length of cuts $\sharp_{\texttt{len}}$
Output : Generated screening cuts $S_{\texttt{cuts}}$
Initialize $S_{\texttt{cuts}} = \emptyset$;
Set $\texttt{SR} = \texttt{SR}_{\texttt{inc}}$ or $\texttt{SR} = \texttt{SR}_{\texttt{exc}}$ from (9) or (10), set $\texttt{stopflag} =$ false;
Initialize starting search index $s_{\texttt{inc}}$ or $s_{\texttt{exc}}$ from (11); // STEP 1
while *stopflag* $\neq$ *true* **do**
  Enumerate supports in $\mathcal{T}_s(\bar{\mathbf{z}})$ or $\mathcal{T}'_s(\bar{\mathbf{z}})$, see e.g. Algorithm 3; // STEP 2
  foreach $\mathbf{x} \in \mathcal{T}_s(\bar{\mathbf{z}})$ *or* $\mathcal{T}'_s(\bar{\mathbf{z}})$ **do**
    Add inclusive/exclusive cut into $S_{\texttt{cuts}}$;
    Update $\sharp_{\texttt{max}} = \sharp_{\texttt{max}} - 1$;
    if $\sharp_{max} \leq 0$ **then**
      set $\texttt{stopflag} =$ true;
      break;
  Update search index: $s \leftarrow s + 1$ (inclusive) or $s \leftarrow s - 1$ (exclusive);
  if $s \in$ *SR* **then**
    set $\texttt{stopflag} =$ false;
  else
    set $\texttt{stopflag} =$ true;
return S_{cuts};

The proof of Claim 2 is given in Appendix A.6. This claim is then used in the design of Algorithm 1.

In summary, the SCPF selects undominated cuts with greater potential screening ability, which corresponds to minimizing the number of variables involved in the cut (i.e., minimizing $|S \cup N|$). For inclusive cut generation, given $\mathbf{x} \in \mathcal{T}_i(\bar{\mathbf{z}}) \neq \emptyset$, the cut involves $i - k$ binary variables. To maximize screening ability, our SCPF searches for the *smallest* index $i \in \{k+1, \ldots, d\}$ such that $\mathcal{T}_i(\bar{\mathbf{z}}) \neq \emptyset$. In contrast, for exclusive cut generation, given $\mathbf{x} \in \mathcal{T}'_i(\bar{\mathbf{z}}) \neq \emptyset$, the cut involves $k - i$ binary variables. To maximize screening ability, the SCPF searches for the *largest* index $i \in \{1, \ldots, k-1\}$ such that $\mathcal{T}'_i(\bar{\mathbf{z}}) \neq \emptyset$.

3.3 SCPF Design

Combining the results from subsection 3.1 and subsection 3.2, we present the SCPF in Algorithm 1. The SCPF balances the effectiveness of the generated cuts against the computational cost of incorporating these cuts into the original problem (1). The cut selection process follows a hierarchical criteria structure (in descending order of priority): (1) ensuring undominatedness; (2) maximizing potential screening ability; and (3) maximizing the left-hand side value of the inequality in (6).

Algorithm 1 contains two hyperparameters: $\sharp_{\texttt{max}}$, which controls the total number of generated cuts, and $\sharp_{\texttt{len}}$, which restricts the maximum number of binary variables involved in a cut (thereby directly ensuring potential screening ability). We give brief discussions on two key steps of Algorithm 1 as follows:

STEP 1. The hyperparameter $\sharp_{\texttt{len}}$ restricts the number of binary variables involved in a cut. Thus,

it suffices to consider indices i such that $i \leq k + \sharp_{\texttt{len}}$ for inclusive cuts, or $i \geq k - \sharp_{\texttt{len}}$ for exclusive cuts. Therefore, the SCPF algorithm restricts the search to feasible points in $\mathcal{T}_i(\overline{\mathbf{z}})$ or $\mathcal{T}'_i(\overline{\mathbf{z}})$ within the following ranges:

$$i \in \texttt{SR}_{\texttt{inc}} := [\![k+1, \min\{k + \sharp_{\texttt{len}}, d\}]\!] \quad \text{for inclusive cuts,} \tag{9}$$

$$i \in \texttt{SR}_{\texttt{exc}} := [\![\max\{1, k - \sharp_{\texttt{len}}\}, k-1]\!] \quad \text{for exclusive cuts.} \tag{10}$$

Using the monotonicity properties established in Claim 1 and Claim 2, the starting search index s is initialized by:

$$\begin{aligned} s_{\texttt{inc}} &:= \operatorname{argmin} \left\{ i \in \texttt{SR}_{\texttt{inc}} \;\middle|\; \exists\, \mathbf{x}' \in K(\overline{\mathbf{z}}) \text{ s.t. } \texttt{supp}(\mathbf{x}') = [\![i - (k-1), i]\!] \right\} \\ s_{\texttt{exc}} &:= \operatorname{argmax} \left\{ i \in \texttt{SR}_{\texttt{exc}} \;\middle|\; \exists\, \mathbf{x}' \in K(\overline{\mathbf{z}}) \text{ s.t. } \texttt{supp}(\mathbf{x}') = [\![i]\!] \cup [\![d - (k-i-1), d]\!] \right\} \end{aligned} \tag{11}$$

If $\mathcal{T}_i(\overline{\mathbf{z}}) = \emptyset$ or $\mathcal{T}'_i(\overline{\mathbf{z}}) = \emptyset$ for all i within the respective searching ranges $\texttt{SR}_{\texttt{inc}}, \texttt{SR}_{\texttt{exc}}$, no feasible support exists in $K(\overline{\mathbf{z}})$ that satisfies the criteria, and the algorithm terminates.

STEP 2. If the current search index s lies within $\texttt{SR}_{\texttt{inc}}$ or $\texttt{SR}_{\texttt{exc}}$, we enumerate feasible supports in $\mathcal{T}_s(\overline{\mathbf{z}})$ or $\mathcal{T}'_s(\overline{\mathbf{z}})$ using Algorithm 3. Note that exhaustive enumeration (i.e., $\sharp_{\texttt{max}} = \infty$) is generally computationally intractable to incorporate all possible cuts in presolving, then $\sharp_{\texttt{max}}$ is usually set to some relatively large number based on problem size. In addition, our numerical experiments (see Figure 6 in Appendix D.1) suggest that screening cuts involving more than two binary variables offer diminishing returns for tested SCQP problems. Thus, we give a small limit (e.g., $\sharp_{\texttt{len}} = 2$) to ensure computational efficiency in practice. Furthermore, to be consistent with the prioritized criteria structure, the recursive algorithm (Algorithm 3) in Step 2 first enumerates solutions yielding larger left-hand-side values in (6).

Then, the search index s is updated (adding one for inclusive cuts, deleting one for exclusive cuts). Based on previous claims (Claim 1 and Claim 2), the updated search index ensures the non-emptiness of the target sets. The algorithm terminates when either the max number of cuts $\sharp_{\texttt{max}}$ is achieved, or the searching 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 to $\sharp_{max} = +\infty$, Algorithm 1 outputs, for a given $K(\overline{\mathbf{z}})$, all minimal inclusive cuts satisfying $\sharp_{len} \leq d - k$, or all minimal exclusive cuts satisfying $\sharp_{len} \leq k - 1$, respectively.*

The proof of Claim 3 is given in Appendix A.7, which ensures that, *Algorithm 1 theoretically captures all minimal screening cuts of the specified length.*

4 Numerical Experiment

This section conducts numerical experiments to address the following three key tasks: (**T1**) *To what extent does the proposed SCPF improve upon existing screening methods in terms of both screening ability and computational efficiency?* (**T2**) *How many gains in screening ability and total runtime are achieved by the multi-support SCPF compared to the single-support SCPF?* (**T3**) *How does the SCS method contribute to the overall efficiency of the framework?*

We compare the proposed SCPF (Algorithm 1) and its multi-support variant, SCPF-m (Algorithm 2), with the existing baseline, Safe Screening Rules (SSR, Atamturk and Gomez (2020)), in

presolving. The comparison works on three distinct data sources: synthetic datasets, real-world datasets, and the simulation library for Sparse Identification of Nonlinear Dynamics (SINDy). Detailed descriptions of data generation, background, and parameter settings are provided in Appendix C.

All experiments are conducted on a Dell Precision 7920 workstation equipped with a 3GHz 48-Core Intel Xeon CPU and 128GB of 2934MHz DDR4 RAM. The proposed methods and baselines are implemented in Python 3.12.8 using the Gurobi 12.0.0 solver. The source code for the experiments will be released on GitHub later.

4.1 Experimental Setup and Implementation

Numerical experiments are conducted based on the following sparse linear ridge regression (SLRR):

$$
\begin{aligned}
v^* = \min_{\boldsymbol{\beta},\mathbf{z}} \quad & \frac{1}{n}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\gamma\|\boldsymbol{\beta}\|_2^2 && \texttt{(comp)}\\
\text{s.t.} \quad & \boldsymbol{\beta}\circ(\mathbf{1}-\mathbf{z})=\mathbf{0},\ \mathbf{1}^\top\mathbf{z}\le k,\ \mathbf{z}\in\{0,1\}^d,\\
= \min_{\boldsymbol{\beta},\mathbf{z},\mathbf{t}\ge\mathbf{0}} \quad & \frac{1}{n}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\gamma\mathbf{1}^\top\mathbf{t} && \texttt{(SOC)}\\
\text{s.t.} \quad & \mathbf{t}\circ\mathbf{z}\ge\boldsymbol{\beta}\circ\boldsymbol{\beta},\ \mathbf{1}^\top\mathbf{z}\le k,\ \mathbf{z}\in\{0,1\}^d,
\end{aligned}
$$

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

Remark 1. *While* (comp) *and* (SOC) *are equivalent in mathematics, their computational performances in Gurobi are different. Among all choices of dimensions in our experiments,* (SOC) *ensures smaller total running times compared with* (comp)*, based on our preliminary numerical experiments*[3]*. Furthermore, we observe that in high-dimensional settings, the substantial memory requirements of the BnB step can cause the Gurobi solver to stall at a relatively high MIPGap in our platform. Such stalling behavior will significantly impact the recorded performance metrics.*

For the SCPF and SCPF-m, we set the max number of cuts to $\sharp_{\texttt{max}}=k$ for inclusive cuts and $\sharp_{\texttt{max}}=d$ for exclusive cuts. While higher $\sharp_{\texttt{max}}$ values tighten the outer approximation to the optimal set, they may impede overall computational efficiency; thus, these limits are chosen to balance support-space tightness with computational efficiency. To ensure the effectiveness of individual cuts, the maximum cut length is fixed at $\sharp_{\texttt{len}}=2$ (see Figure 6 in Appendix D.1). Regarding support selection, SCPF utilizes the optimal relaxation support $\widehat{\mathbf{z}}$, whereas SCPF-m employs the multi-support sequence detailed in Appendix B.1. For all three algorithms, the upper bound $v_{\texttt{ub}}$ is computed using the greedy algorithm proposed by Xie and Deng (2020), as summarized in Appendix B.3.

Numerical performance is measured based on the following two metrics:

[3]To ensure consistency, we only report numerical results corresponding to (SOC) in the main paper. Preliminary comparisons for $d<6000$ showed that (SOC) significantly outperforms (comp); and these results are omitted for brevity. However, comparative results for relatively high-dimensional cases ($d=6000$) are provided in Appendix D.3.

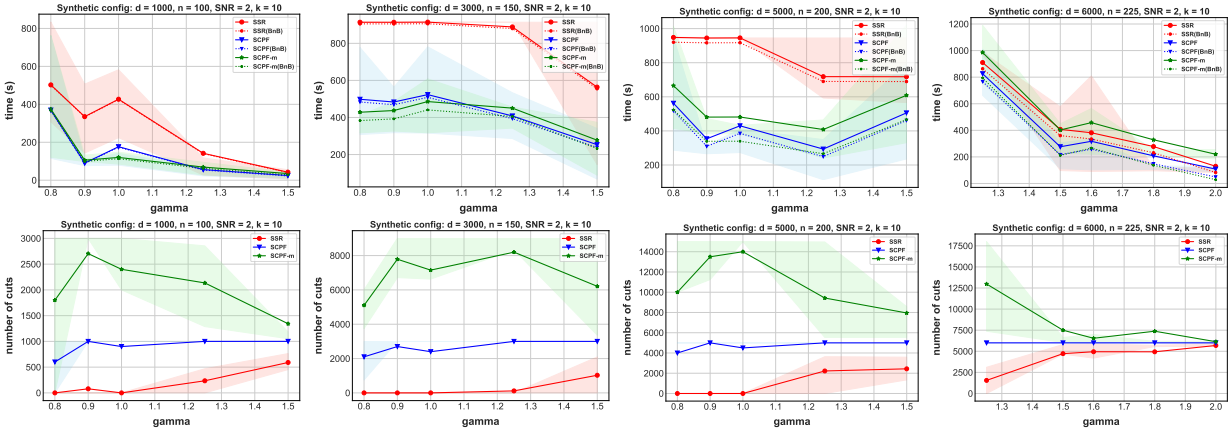


Figure 2. Synthetic datasets. Parameter configurations are listed in the title of each panel, with further details provided in Appendix C. The first row shows the TRT and BnB solving time ($t_{\mathcal{A}}^{\texttt{sol}}$), and the second row reports the SA of the three methods under different γ values. Each dot represents the averaged value over 10 independent instances. The shaded area denotes the inter-quartile range (25th–75th percentiles) of TRT and SA, respectively.

1. *Screening Ability (SA)*: This metric quantifies the reduction in the support space. For the SSR baseline, SA corresponds to the number of binary variables fixed in presolving. For SCPF and SCPF-m (with $\sharp_{\texttt{len}} = 2$), SA is reported as the number of inclusive and exclusive cuts incorporated.

2. *Total Running Time (TRT)*: For any screening method $\mathcal{A}$, this metric is defined as $t_{\mathcal{A}} := t_{\mathcal{A}}^{\texttt{pre}} + t_{\mathcal{A}}^{\texttt{sol}}$, where $t_{\mathcal{A}}^{\texttt{pre}}$ is the presolving time (cut generation) for $\mathcal{A}$, and $t_{\mathcal{A}}^{\texttt{sol}}$ is the *BnB solving time* required to reach a specific MIPGap for SLRR with incorporated cuts. Here, the term MIPGap is defined as MIPGap $:= |v_p - v_d|/|v_p|$ with v_p and v_d the primal and dual objective bounds, respectively.

For each instance $(\mathbf{X}, \mathbf{Y})$ with regularization parameter γ and sparsity level k, we take a two-stage performance evaluation procedure:

1. *Baseline SSR:* We first solve SLRR using the SSR method via Gurobi. The solver terminates if one of the following two stopping criteria (SC) is satisfied: (SSR-SC1) the time limit reaches 15 minutes, or (SSR-SC2) the MIPGap falls below 1%. We record the final solving time $t_{\texttt{SSR}}^{\texttt{sol}}$ and the final MIPGap $\texttt{Gap}_{\texttt{SSR}}$.

2. *Proposed Frameworks (SCPF/SCPF-m):* We subsequently solve SLRR using the proposed frameworks (SCPF/SCPF-m). To ensure a rigorous comparison, the solver for framework $\mathcal{A} \in \{\texttt{SCPF}, \texttt{SCPF-m}\}$ terminates if: ($\mathcal{A}$-SC1) the solving time reaches 15 minutes, or ($\mathcal{A}$-SC2) the MIPGap improves upon the baseline, specifically MIPGap $< \max\{\texttt{Gap}_{\texttt{SSR}} - \varepsilon, 1\%\}$ with tolerance $\varepsilon = 0.1\%$. This criterion is included to address situations where the baseline method is stuck at a specific MIPGap level.

4.2 Discussions on Numerical Results

Based on the numerical results, we address the three key tasks proposed at the beginning of section 4.

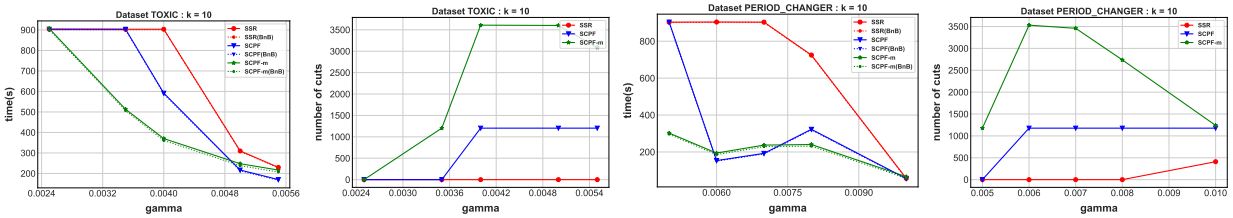


Figure 3. Real datasets. Instance name and parameter configurations are listed in the title of every panel, and details for the real datasets are provided in Appendix C. From left to right, the 1st and 3rd panels show the TRT and BnB solving time ($t_{\mathcal{A}}^{\texttt{sol}}$) over the Toxicity and Period Changer instances under different γ values, respectively. The 2nd and 4th panels report the corresponding SA.

(**T1**) *To what extent does the proposed SCPF improve upon existing screening methods in terms of both screening ability and computational efficiency?*

We first evaluate the extent to which SCPF improves upon the SSR baseline in terms of Screening Ability (SA) and Total Running Time (TRT). In general, the results presented in Figures 2 to 5 show that the performance of both methods heavily depends on the parameter γ to the level of strong convexity. *As γ decreases, the metric SA exhibits a distinct "three-phase" trend, which also significantly reflects on metric TRT.* We illustrate this trend using the synthetic dataset with dimensions $(d, n) = (5000, 250)$ (see Figure 2) as an example.

1. *Easy Phase (High γ): Strong SSR vs. Strong SCPF.* When γ is relatively large (e.g., $\gamma \geq 1.25$ in the example), both SSR and SCPF successfully fix the majority of binary variables with a similar performance on SA. Thus, SCPF cannot generate sufficiently many additional cuts, resulting in a TRT comparable to that of SSR.

2. *Challenging Phase (Intermediate γ): Weak SSR vs. Strong SCPF.* As γ decreases, the SA of SSR weakens significantly. In contrast, SCPF maintains relatively robust performance in SA by generating effective inclusive and exclusive cuts in presolving. This difference creates a substantial performance gap to TRT; in our example ($\gamma \in [0.8, 1.25]$), SCPF achieves a roughly 2.0× acceleration compared to SSR on average.

3. *Hard Phase (Low γ): Weak SSR vs. Weak SCPF.* At sufficiently low γ regime (e.g., $\gamma < 0.8$), the capacity of both methods to generate screening cuts becomes negligible. Therefore, the whole solving procedure reduces to a pure BnB procedure without presolving benefits, i.e., no effective screening cuts generated in presolving.

Overall, computational efficiency is strongly driven by SA. The "three-phase" trend is consistent across most instances, with SCPF demonstrating superior performance primarily in the Challenging Phase, typically yielding 1.7× to 3.0× improvements in TRT. We note that the specific γ-intervals defining these phases are instance-dependent, as illustrated in Figures 1 to 5.

(**T2**) *How many gains in screening ability and total runtime are achieved by the multi-support SCPF compared to the single-support SCPF?*

We next quantify the gains in SA and TRT achieved by the SCPF-m compared to the SCPF. The numerical results across all datasets confirm that SCPF-m consistently exhibits superior SA. As evidenced in Figures 2 to 5, SCPF-m generates a strictly larger set of screening cuts, including cuts inaccessible to the SCPF even when $\sharp_{\texttt{max}} = \infty$ (see Appendix D.2). Note that this enhanced screening ability incurs a higher computational cost in the presolving step.

Regarding the TRT, SCPF-m does not consistently outperform SCPF, suggesting a complex trade-off between the effectiveness of cut generation and the overall solver efficiency. Specifically, for

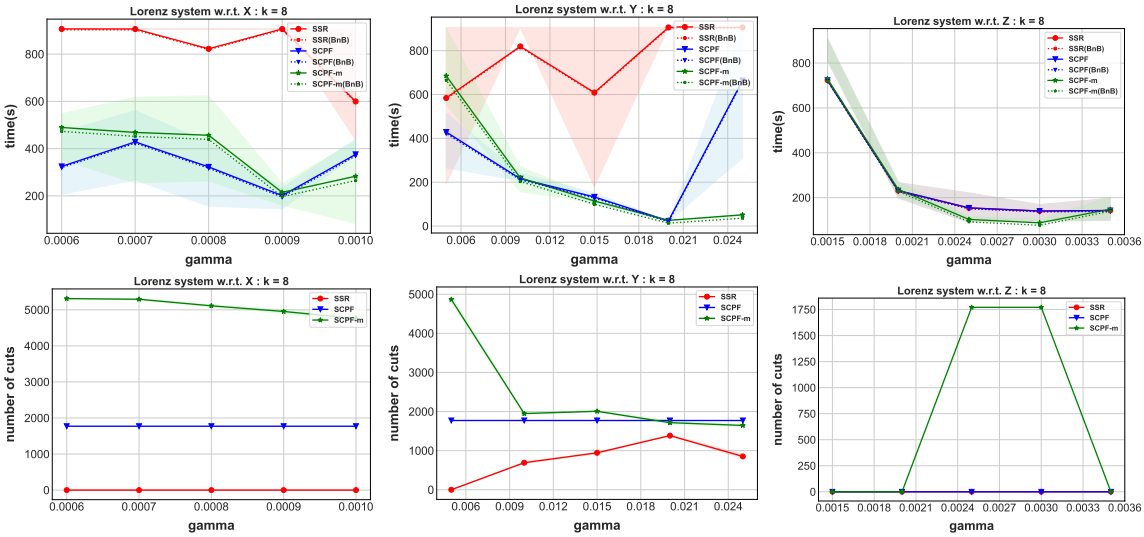


Figure 4. Lorenz system from SINDy. The parameter configurations are listed in the title of every panel, and details are given in Appendix C. For each dimension, the panels are arranged to show TRT and BnB solving time ($t^{\texttt{sol}}_{\mathcal{A}}$) in the first 3 panels, followed by the corresponding SA in the last 3 panels under different γ values. Each dot represents the averaged value over 10 independent trajectories. The shaded area denotes the inter-quartile range (25th–75th percentiles) of the TRT and SA metric, respectively.

synthetic datasets, performance is comparable in low-dimensional instances (e.g., $(d, n) = (1000, 100)$ and $(3000, 150)$). However, in high-dimensional settings (e.g., $(d, n) = (5000, 200)$ and $(6000, 225)$), SCPF achieves a lower TRT; while the BnB solving times are similar, SCPF-m suffers from significant presolving overhead. For *SINDy datasets*, SCPF outperforms SCPF-m across all tested dimensions (see Figures 4 and 5). For *real-world datasets*, results are mixed: performance is comparable on the Period Changer dataset, whereas SCPF-m proves superior on the Toxicity dataset (see Figure 3).

These findings suggest that incorporating a larger set of undominated and effective screening cuts does not monotonically reduce TRT. This phenomenon likely results from diminishing returns in support-space reduction relative to the increased presolving overhead and the computational cost of managing newly incorporated constraints.

(**T3**) *How does the SCS method contribute to the overall efficiency of the framework?*

This part validates the effectiveness and necessity of the proposed SCS method, which selects cuts based on dual criteria: undominatedness and effectiveness. To clearly identify the contribution of each criterion, we compare the standard SCPF against variants lacking complete SCS method on synthetic datasets (details in Appendix D.1). The numerical results (see Figure 6) ensure that satisfying both conditions is critical. Specifically, incorporating *undominated but ineffective* cuts (i.e., those involving three or more binary variables) degrades BnB efficiency by expanding the constraint pool with cuts of negligible (potential) screening ability. Conversely, relying on *effective but dominated* cuts (e.g., generating $z_i + z_j \leq 1$ when the tighter cut $z_i = 0$ is valid) significantly increases TRT, as their dominatedness weakens solver performance relative to the standard SCS. These experiments demonstrate that excluding either criterion leads to notable performance regression, whereas applying both criterion proposed in standard SCS is essential for reducing TRT.

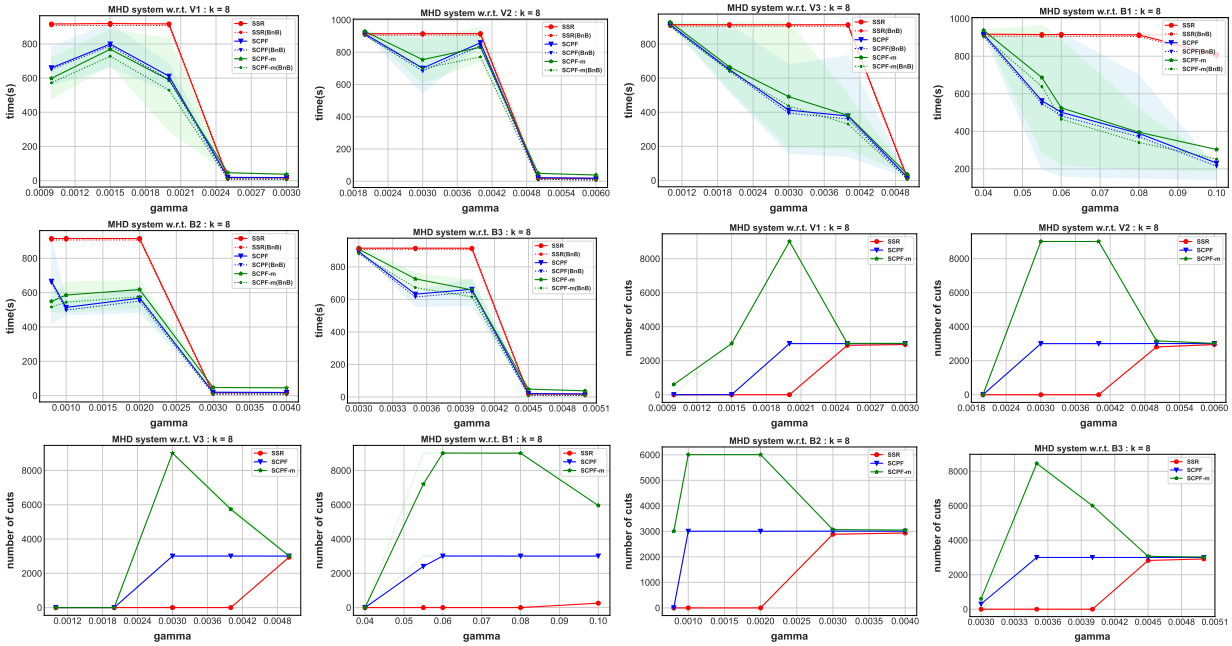


Figure 5. MHD system from SINDy. The parameter configurations are listed in the title of every panel, and details are given in Appendix C. For each dimension, the panels are arranged to show TRT and BnB solving time ($t_{\mathcal{A}}^{\texttt{sol}}$) in the first 6 panels, followed by the corresponding SA in the last 6 panels under different γ values. Each dot represents the averaged value over 10 independent trajectories. The shaded area denotes the inter-quartile range (25th–75th percentiles) of the TRT and SA metric, respectively.

4.3 Summary of Numerical Experiments

Based on these numerical experiments, the proposed SCPF and SCPF-m (i) offer clear advantages in SA, which validates the theoretical results presented in section 2 and section 3; (ii) yield significant improvements in TRT across a majority of instances, especially for these in challenging phase.

1. The SCPF achieves a finer balance between enhanced screening ability and computational overhead in presolving. In particular, in challenging phase, SCPF achieves accelerations of approximately 1.7× to 3.0× on TRT.

2. The multi-support variant, SCPF-m, consistently strengthens SA by generating a strictly larger set of cuts, including those inaccessible to the single-support SCPF (even with $\sharp_{\texttt{max}} = \infty$). While this enhancement requests additional presolving costs, its impact on the BnB process is instance-dependent. Empirically, SCPF-m maintains comparable runtime to SCPF on most datasets and achieves substantial speedups on specific real-world instances (e.g., Toxicity instance in Figure 3).

3. The SCS method is essential to SCPF's success. By selecting cuts that are *both undominated and effective*, SCS ensures that newly added constraints would improve the solving process. Numerical results show that omitting either criterion significantly degrades the computational performance.

5 Conclusion & Future Directions

In conclusion, this paper proposes a novel *Screening-based Cut Presolving Framework* (SCPF) designed to enhance existing screening approaches used in the presolving step of solving SCQP.

The framework contains two parts. First, the proposed *Screening-based Cut Generation* (SCG) rule leverages convex relaxations to identify and eliminate non-optimal support patterns. By doing so, our SCG rule achieves a finer balance between computational overhead and enhanced screening ability in presolving. Second, the *Screening-based Cut Selection* (SCS) method establishes necessary and sufficient conditions for certifying minimal SCG-tuples that yield undominated cuts. Furthermore, the SCS incorporates a "(potential) screening ability" criterion to prioritize the generation of *effective & undominated cuts*. Together, the two components presented in SCS method form the foundation for our SCPF. Numerical experiments in Section 4 further validate our theoretical results and demonstrate the computational effectiveness of the proposed method.

We close with some potential extensions and research questions for future investigation. First, there is significant potential to sharpen existing screening techniques through stronger optimality-based criteria, particularly in non-smooth, highly degenerate, or non-convex settings. Second, characterizing the closure of minimal screening cuts derived from a specific relaxation remains an open problem; understanding the theoretical limits of outer approximations established by these relaxations is of critical research interest. Finally, motivated by recent advancements Liu et al. (2023, 2025), it is worth investigating whether the proposed framework can be extended beyond presolving to develop tailored BnB algorithms.

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, and Tier-2 grant T2EP20125-0030.

References

T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020.

A. Askari, A. d'Aspremont, and L. E. Ghaoui. Approximation bounds for sparse programs. *SIAM Journal on Mathematics of Data Science*, 4(2):514–530, 2022.

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.

A. Atamturk and A. Gomez. Rank-one convexification for sparse regression. *Journal of Machine Learning Research*, 26(35):1–50, 2025.

D. Bertsimas and R. Cory-Wright. A scalable algorithm for sparse portfolio selection. *INFORMS Journal on Computing*, 34(3):1489–1511, May 2022. ISSN 1526-5528. doi: 10.1287/ijoc.2021.1127. URL http://dx.doi.org/10.1287/ijoc.2021.1127.

D. Bertsimas and W. Gurnee. Learning sparse nonlinear dynamics via mixed-integer optimization. *Nonlinear Dynamics*, 111(7):6585–6604, Jan. 2023. ISSN 1573-269X. doi: 10.1007/s11071-022-08178-9. URL http://dx.doi.org/10.1007/s11071-022-08178-9.

D. Bertsimas and B. Van Parys. Sparse high-dimensional regression. *The Annals of Statistics*, 48(1): 300–323, 2020.

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.

D. Bertsimas, J. Pauphilet, and B. Van Parys. Sparse classification: a scalable discrete optimization perspective. *Machine Learning*, 110(11):3177–3209, 2021.

K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. Van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, et al. The scip optimization suite 8.0. *arXiv preprint arXiv:2112.08872*, 2021.

H. T. Bui, R. Loxton, and Q. Lin. On cutting plane algorithms for nonlinear binary optimization. *SIAM Journal on Optimization*, 35(2):1364–1392, 2025.

R. Chen and J. Luedtke. On generating lagrangian cuts for two-stage stochastic integer programs. *INFORMS Journal on Computing*, 34(4):2332–2349, July 2022. ISSN 1526-5528. doi: 10.1287/ijoc.2022.1185. URL http://dx.doi.org/10.1287/ijoc.2022.1185.

Y. Dai and C. Chen. Serial and parallel two-column probing for mixed-integer programming, 2025. URL https://arxiv.org/abs/2408.16927.

C. D'Ambrosio, M. Fampa, J. Lee, and F. Sinnecker. On a geometric graph-covering problem related to optimal safety-landing-site location. *Discrete Applied Mathematics*, 379:613–634, 2026. doi: 10.1016/j.dam.2025.09.036.

B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton. Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, 2020. doi: 10.21105/joss.02104. URL https://doi.org/10.21105/joss.02104.

H. Deng and W. Xie. On the relu lagrangian cuts for stochastic mixed integer programming, 2024. URL https://arxiv.org/abs/2411.01229.

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.

A. Frangioni, C. Gentile, and J. Hungerford. Decompositions of semidefinite matrices and the perspective reformulation of nonseparable quadratic programs. *Mathematics of Operations Research*, 45(1):15–33, 2020.

J. Gao and D. Li. Optimal cardinality constrained portfolio selection. *Operations research*, 61(3): 745–761, 2013.

A. Gómez and W. Xie. A note on quadratic constraints with indicator variables: Convex hull description and perspective relaxation. *Operations Research Letters*, 52:107059, 2024.

S. Han, A. Gómez, and A. Atamtürk. The equivalence of optimal perspective formulation and shor's sdp for quadratic programs with indicator variables. *Operations Research Letters*, 50(2):195–198, 2022.

H. Hazimeh and R. Mazumder. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Operations Research*, 68(5):1517–1537, 2020.

N. Jiang and W. Xie. The terminator: An integration of inner and outer approximations for solving wasserstein distributionally robust chance constrained programs via variable fixing. *INFORMS Journal on Computing*, 37(2):381–412, Mar. 2025. ISSN 1526-5528. doi: 10.1287/ijoc.2023.0299. URL http://dx.doi.org/10.1287/ijoc.2023.0299.

A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton. Pysindy: A comprehensive python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, 2022. doi: 10.21105/joss.03994. URL https://doi.org/10.21105/joss.03994.

Y. Li, M. Fampa, J. Lee, F. Qiu, W. Xie, and R. Yao. D-optimal data fusion: Exact and approximation algorithms. *INFORMS Journal on Computing*, 36(1):97–120, 2024.

J. Liu, S. Rosen, C. Zhong, and C. Rudin. Okridge: Scalable optimal k-sparse ridge regression. *Advances in neural information processing systems*, 36:41076–41258, 2023.

J. Liu, S. Shafiee, and A. Lodi. Scalable first-order method for certifying optimal k-sparse GLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 39455–39481. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/liu25bk.html.

M. Pilanci, M. J. Wainwright, and L. El Ghaoui. Sparse learning via boolean relaxations. *Mathematical Programming*, 151(1):63–87, 2015.

L. Schürmann and P. Mutzel. *A Reduced Cost-based Model Strengthening Method*, page 75–86. Society for Industrial and Applied Mathematics, Jan. 2023. ISBN 9781611977714. doi: 10.1137/1.9781611977714.7. URL http://dx.doi.org/10.1137/1.9781611977714.7.

M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical programming*, 99(3):563–591, 2004.

L. Wei and S. Küçükyavuz. An outer approximation method for solving mixed-integer convex quadratic programs with indicators. *Preprint*, 2024.

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.

P. M. F. Yamagishi, M. Fampa, and J. Lee. The dual-path fixing strategy and its application to the set-covering problem, 2026. URL https://arxiv.org/abs/2601.20977.

Y. Yang. Deluxing: Deep lagrangian underestimate fixing for column-generation-based exact methods. *Operations Research*, 73(3):1184–1207, May 2025. ISSN 1526-5463. doi: 10.1287/opre.2023.0398. URL http://dx.doi.org/10.1287/opre.2023.0398.

J. Zou, S. Ahmed, and X. A. Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1–2):461–502, Mar. 2018. ISSN 1436-4646. doi: 10.1007/s10107-018-1249-5. URL http://dx.doi.org/10.1007/s10107-018-1249-5.

A Proofs in section 2 and section 3

A.1 Proof of Corollary 1

Proof. Let $R := [\![d]\!] \setminus (S \cup N)$ denote the set of remaining indices. Assume the non-trivial case $C \neq \emptyset$.

($\Rightarrow$) By definition of a valid SCG-tuple, we have $v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}, S, N, C) > v_{\texttt{ub}}$, and $C = \{i \in R \mid \widehat{w}_i \geq (\widehat{\mathbf{w}}_R)_{[c]}\}$ with $c = \min\{k - |S|, |R|\}$. Since $C \neq \emptyset$, we have $|C| = c$. It suffices to show condition (2) holds. Prove by contradiction. Suppose there exists $j \notin (S \cup N \cup C)$ such that $\widehat{w}_j \geq \underline{\widehat{\mathbf{w}}_C}$. Then $j \in R$ and $\widehat{w}_j \geq (\widehat{\mathbf{w}}_R)_{[c]}$, implying $j \in C$ by definition. Contradiction.

($\Leftarrow$) Conversely, assume tuple $(\overline{\mathbf{z}}, S, N, C)$ satisfies the three conditions. Define $\mathcal{D} := \{i \in R \mid \widehat{w}_i \geq (\widehat{\mathbf{w}}_R)_{[|C|]}\}$. Since $|C| \leq |R|$, this set is well-defined and $|\mathcal{D}| = |C|$. It suffices to show that $C = \mathcal{D}$. Take any $i' \in C$ and suppose, for contradiction, that $i' \notin \mathcal{D}$. Then $\widehat{w}_{i'} < (\widehat{\mathbf{w}}_R)_{[|C|]}$. Since $|\mathcal{D}| = |C|$, there exists some $j' \in \mathcal{D} \setminus C$ with $\widehat{w}_{j'} \geq (\widehat{\mathbf{w}}_R)_{[|C|]} > \widehat{w}_{i'} \geq \underline{\widehat{\mathbf{w}}_C}$. Then we have $j' \notin S \cup N \cup C$ but $\widehat{w}_{j'} \geq \underline{\widehat{\mathbf{w}}_C}$, contradicting condition (2). Hence every $i' \in C$ belongs to $\mathcal{D}$, and therefore $C = \mathcal{D}$. □

A.2 Proof of Proposition 2

Proof. We prove both parts separately.

Part 1: Validity of the SCG-tuple. It suffices to show that inequality (4) holds for the SCG-tuple $(\overline{\mathbf{z}}, S', N', C')$. From the proof of Theorem 1, we know $v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}, S', N', C') = \min_{\boldsymbol{\beta}} f_1(\boldsymbol{\beta}; \widehat{\mathbf{p}}(\overline{\mathbf{z}})) - \max_{\mathbf{z} \in Z^k_{\texttt{relax}}(S', N')} f_2(\mathbf{z}; \widehat{\mathbf{p}}(\overline{\mathbf{z}}))$. Since $S' \supseteq S$ and $N' \supseteq N$, it follows that $v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}, S', N', C') \geq v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}, S, N, C)$. Clearly, if $(\overline{\mathbf{z}}, S, N, C)$ is valid, then $(\overline{\mathbf{z}}, S', N', C')$ is also valid.

Part 2: Cut dominance. We show that the cut $\sum_{i \in S'} z_i + \sum_{i \in N'} (1 - z_i) \leq |S'| + |N'| - 1$ is dominated by $\sum_{i \in S} z_i + \sum_{i \in N} (1 - z_i) \leq |S| + |N| - 1$. Take any $\mathbf{z} \in Z^k$ satisfying $\sum_{i \in S} z_i + \sum_{i \in N} (1 - z_i) \leq |S| + |N| - 1$, we have $\sum_{i \in S'} z_i = \sum_{i \in S} z_i + \sum_{i \in (S' \setminus S)} z_i \leq |S| + \sum_{i \in N} z_i - 1 + |S'| - |S| \leq |S'| + \sum_{i \in N'} z_i - 1$, which shows that the cut $\sum_{i \in S'} z_i + \sum_{i \in N'} (1 - z_i) \leq |S'| + |N'| - 1$ is satisfied by $\mathbf{z}$. □

A.3 Proof of Proposition 3

Proof. We separately give proofs on the two scenarios presented in the proposition.

The first scenario. Assume $|\texttt{supp}(\mathbf{x})| < k$. Thus we have $|S \cup C| < k$ and $N = [\![d]\!] \setminus \texttt{supp}(\mathbf{x})$ based on validity. Since $|N| > d - k$, based on the first case in Theorem 2, the SCG-tuple $(\overline{\mathbf{z}}, S, [\![d]\!] \setminus \texttt{supp}(\mathbf{x}), C)$ is minimal if and only if $S = \emptyset$ and $C = \texttt{supp}(\mathbf{x})$.

The second scenario. Assume $|\texttt{supp}(\mathbf{x})| = k$. Thus we have $|S \cup C| = |\texttt{supp}(\mathbf{x})| = k$, and $|N| \leq d - k$ based on validity.

1. First, we construct minimal SCG-tuple with $C = \emptyset$. It is easy to verify that $(\overline{\mathbf{z}}, \texttt{supp}(\mathbf{x}), N, \emptyset)$ is a valid SCG-tuple for any $N \subseteq ([\![d]\!] \setminus \texttt{supp}(\mathbf{x}))$. In order to ensure minimality, by the second case in Theorem 2, $(\overline{\mathbf{z}}, \texttt{supp}(\mathbf{x}), N, \emptyset)$ is minimal if and only if $N = \emptyset$ and $\max_{i \in \texttt{supp}(\mathbf{x})} \widehat{w}_i < \widehat{w}_{[1]}$, which is equivalent to $B_1(\mathbf{x}) \neq \emptyset$ since $\widehat{\mathbf{w}}$ is in decreasing order. This corresponds to the boundary case of $l = 0$.

2. Second, we construct minimal SCG-tuple with $C \neq \emptyset$. Denote i' as $\widehat{w}_{i'} = \underline{\widehat{\mathbf{w}}_C}$. We have two subcases to consider on i'.

Case 1. If $i' \in A_m(\mathbf{x})$, then $N \supseteq \texttt{comp}(\texttt{supp}(\mathbf{x}))$ to ensure validity. Based on the third case in Theorem 2, the SCG-tuple $(\overline{\mathbf{z}}, \emptyset, \texttt{comp}(\texttt{supp}(\mathbf{x})), \texttt{supp}(\mathbf{x}))$ is minimal, which corresponds to the boundary case of $l = m$.

Case 2. If $i' \in A_\ell(\mathbf{x})$ for some $\ell = 1, \ldots, m-1$, similarly, based on the third case in Theorem 2, the SCG-tuple $(\overline{\mathbf{z}}, \cup_{j=\ell+1}^{m} A_j(\mathbf{x}), \cup_{j=1}^{l} B_j(\mathbf{x}), \cup_{j=1}^{l} A_j(\mathbf{x}))$ is minimal.

□

A.4 Proof of bounds in Definition 5

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 $|N| > d-k$, based on validity, we have $(S \cup N \cup C) = [\![d]\!]$. 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 $|N| \leq d-k$, we have $|S| + |C| = k$ and $(S \cup N \cup C) \subseteq [\![d]\!]$. The remaining index set is $[\![d]\!] \backslash (S \cup N)$ 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.5 Proof of Claim 1

Proof. We show two results in the Claim 1 separately.

- One direction is obvious and we suppose $\mathcal{T}_i(\overline{\mathbf{z}}) \neq \emptyset$. Pick any $\mathbf{x} \in \mathcal{T}_i(\overline{\mathbf{z}})$. Let $\mathbf{x}' \in \{0,1\}^d$ and $\texttt{supp}(\mathbf{x}') = [\![i-(k-1), i]\!]$. Given that $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$ is in decreasing order, we have $\langle \widehat{\mathbf{w}}(\overline{\mathbf{z}}), \mathbf{x}' \rangle \leq \langle \widehat{\mathbf{w}}(\overline{\mathbf{z}}), \mathbf{x} \rangle < q(\overline{\mathbf{z}})$. Thus $\mathbf{x}' \in K(\overline{\mathbf{z}})$.

- Given $\mathcal{T}_i(\overline{\mathbf{z}}) \neq \emptyset$, it is sufficient to show that $\mathcal{T}_{i+1}(\overline{\mathbf{z}}) \neq \emptyset$. Based on the first result, we have $\mathbf{x}'_i \in K(\overline{\mathbf{z}})$ and $\texttt{supp}(\mathbf{x}'_i) = [\![i-(k-1), i]\!]$. Let $\mathbf{x}'_{i+1} \in \{0,1\}^d$ and $\texttt{supp}(\mathbf{x}'_{i+1}) = [\![i+1-(k-1), i+1]\!]$. It is obvious that $\langle \mathbf{x}'_{i+1}, \widehat{\mathbf{w}}(\overline{\mathbf{z}}) \rangle \leq \langle \mathbf{x}'_i, \widehat{\mathbf{w}}(\overline{\mathbf{z}}) \rangle < q(\overline{\mathbf{z}})$. Thus $\mathbf{x}'_{i+1} \in K(\overline{\mathbf{z}})$ and we are done.

□

A.6 Proof of Claim 2

Proof. We show two results in the Claim 2 separately.

- One direction is obvious and we suppose $\mathcal{T}'_i(\overline{\mathbf{z}}) \neq \emptyset$. Pick any $\mathbf{x} \in \mathcal{T}'_i(\overline{\mathbf{z}})$. Let $\mathbf{x}' \in \{0,1\}^d$ and $\texttt{supp}(\mathbf{x}') = [\![i]\!] \cup [\![d-(k-i-1), d]\!]$. Given that $\widehat{\mathbf{w}}(\overline{\mathbf{z}})$ is in decreasing order, we have $\langle \widehat{\mathbf{w}}(\overline{\mathbf{z}}), \mathbf{x}' \rangle \leq \langle \widehat{\mathbf{w}}(\overline{\mathbf{z}}), \mathbf{x} \rangle < q(\overline{\mathbf{z}})$. Thus we have $\mathbf{x}' \in K(\overline{\mathbf{z}})$.

- Given $\mathcal{T}'_i(\overline{\mathbf{z}}) \neq \emptyset$, it is sufficient to show that $\mathcal{T}'_{i-1}(\overline{\mathbf{z}}) \neq \emptyset$. Based on the first result, we have $\mathbf{x}'_i \in K(\overline{\mathbf{z}})$ and $\texttt{supp}(\mathbf{x}'_i) = [\![i]\!] \cup [\![d-(k-i-1), d]\!]$. We construct $\mathbf{x}'_{i-1} \in \{0,1\}^d$ and $\texttt{supp}(\mathbf{x}'_{i-1}) = [\![i-1]\!] \cup [\![d-(k-i), d]\!]$. It is obvious that $\langle \mathbf{x}'_{i-1}, \widehat{\mathbf{w}}(\overline{\mathbf{z}}) \rangle \leq \langle \mathbf{x}'_i, \widehat{\mathbf{w}}(\overline{\mathbf{z}}) \rangle < q(\overline{\mathbf{z}})$. Thus $\mathbf{x}'_{i-1} \in K(\overline{\mathbf{z}})$ and we are done.

□

A.7 Proof of Claim 3

Proof. We only prove the results on inclusive cuts, as the argument for exclusive cuts follows similarly.

Inclusive cuts. For any minimal inclusive cut, the associated minimal SCG tuple corresponds to a unique $\mathbf{x} \in K(\overline{\mathbf{z}})$, as discussed under (6) in section 2. Given the cut length satisfies $\sharp_{\texttt{len}} \leq d-k$, we first claim $\mathbf{x} \in \mathcal{T}(\overline{\mathbf{z}})$. Suppose not. We have $|\operatorname{supp}(\mathbf{x})| < k$, and the corresponding minimal inclusive cut has length $|[\![d]\!] \setminus \texttt{supp}(\mathbf{x})| > d-k \geq \sharp_{\texttt{len}}$ from Proposition 3. Contradiction.

Next, $\mathbf{x} \in \mathcal{T}(\overline{\mathbf{z}})$ implies that $\mathbf{x} \in \mathcal{T}_{i'}(\overline{\mathbf{z}})$ for some $i' \in [\![k+1, d]\!]$. It suffices to show that $\mathcal{T}_{i'}(\overline{\mathbf{z}})$ is reached before termination of Algorithm 1. First, we have the cut length $|\texttt{comp}(\texttt{supp}(\mathbf{x}))| = i'-k \leq \sharp_{\texttt{len}}$, which is equivalent to $i' \leq k + \sharp_{\texttt{len}}$. Thus $i' \in \texttt{SR}_{\texttt{inc}}$, and the searching index s initialized in STEP 1 satisfies $s \leq i'$. Since $\sharp_{\texttt{max}} = +\infty$, Algorithm 1 continues until $s = \min\{k + \sharp_{\texttt{len}}, d\}$, and therefore reaches $s = i'$ before termination.

□

B Algorithm Design

B.1 Multi-support SCG rule & SCPF-m Algorithm

This section details the design of the Multi-support Screening Cut Presolving Framework (SCPF-m).

Recall that the validity of an SCG-tuple for a desired pair of subsets (S, N) depends on the chosen relaxed support $\overline{\mathbf{z}} \in Z^k_{\texttt{relax}}$. To satisfy the multi-support optimality-based criterion $\max_{t=1}^{T} v^{\texttt{lb}}_{\texttt{relax}}(\overline{\mathbf{z}}^{(t)}, S, N, C^{(t)}) > v_{\texttt{ub}}$, preliminary experiments indicate that the support sequence $\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^{T}$ should exhibit *diversity*; that is, supports chosen in $\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^{T}$ should reflect different support selection preferences. Without such a diversity property, the sequence induces nearly identical k-Knap sets $K(\overline{\mathbf{z}}^{(t)})$, impeding the generation of effective screening cuts. Finally, our SCPF-m aggregates cuts from multiple supports and implements a final removing step to ensure undominatedness.

Algorithm 2: SCPF-m$(\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^{T}, \sharp_{\texttt{max}}, \sharp_{\texttt{len}})$

Input : multi-support $\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^{T}$, max number of cuts $\sharp_{\texttt{max}}$, max length of cuts $\sharp_{\texttt{len}}$
Output : Generated screening cuts $S^{(T)}_{\texttt{cuts}}$
Initialize $S^{(0)}_{\texttt{cuts}} = \emptyset$;
foreach $\overline{\mathbf{z}}^{(t)}$ **do**
  $S^{(t)}_{\texttt{cuts}} \leftarrow S^{(t-1)}_{\texttt{cuts}} \cup \texttt{SCPF}\left(K(\overline{\mathbf{z}}^{(t)}), \sharp_{\texttt{max}}, \sharp_{\texttt{len}}\right)$;
Remove dominated cuts in $S^{(T)}_{\texttt{cuts}}$;

We present the SCPF-m in Algorithm 2. Its core strategy involves executing Algorithm 1 across the support sequence $\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^{T}$ to collect a comprehensive set of undominated and effective cuts.

In our numerical experiments, we employ the following iterative heuristic to construct the sequence of relaxed supports $\{\overline{\mathbf{z}}^{(t)}\}_{t=1}^{T}$.

Initialization: Set $\overline{\mathbf{z}}^{(1)} = \widehat{\mathbf{z}}$, where $\widehat{\mathbf{z}}$ is an optimal solution to the original relaxation (2).

Iterative Diversity Exploration: Subsequent relaxed supports are generated by perturbing the non-support indices of $\widehat{\mathbf{z}}$, i.e., $[\![d]\!] \setminus \texttt{Top}_k(\widehat{\mathbf{z}})$:

1. *Perturbation of Non-Support Top Components:* Let $I^{\texttt{Top}}$ be the index set corresponding to the

largest components within the non-support indices of $\widehat{\mathbf{z}}$. Generate relaxed supports by solving $\overline{\mathbf{z}} \in \operatorname{argmin} v_{\texttt{relax}}(I^{\texttt{Top}}, \emptyset)$, or $\overline{\mathbf{z}} \in \operatorname{argmin} v_{\texttt{relax}}(\emptyset, I^{\texttt{Top}})$.

2. *Perturbation of Non-Support Bottom Components:* Let $I^{\texttt{Bottom}}$ be the index set corresponding to the smallest components within the non-support indices of $\widehat{\mathbf{z}}$. Generate relaxed supports by solving $\overline{\mathbf{z}} \in \operatorname{argmin} v_{\texttt{relax}}(I^{\texttt{Bottom}}, \emptyset)$, or $\overline{\mathbf{z}} \in \operatorname{argmin} v_{\texttt{relax}}(\emptyset, I^{\texttt{Bottom}})$.

This heuristic enhances diversity by forcing specific non-support indices of $\widehat{\mathbf{z}}$ to extreme values (0 or 1). Empirically, for both strategies, we prioritize the formulation $\overline{\mathbf{z}} \in \operatorname{argmin} v_{\texttt{relax}}(S, \emptyset)$ (fixing indices to 1). But their underlying motivations are different. In the first strategy, the empirical challenges of generating effective inclusive cuts motivate us to focus on exclusive cuts by forcing components in $I^{\texttt{Top}}$ to 1. In the second strategy, fixing components in $I^{\texttt{Bottom}}$ to 1 (rather than 0) maximizes the divergence between $\widehat{\mathbf{z}}$ and the newly generated support. Therefore, by activating non-support indices, we explore the boundary of the support space and stress-test weak signal components.

In our SCPF-m implementation, we set $T = 3$ and $\overline{\mathbf{z}}^{(1)} = \widehat{\mathbf{z}}$ by default. We then select $\overline{\mathbf{z}}^{(2)} \in \operatorname{argmin} v_{\texttt{relax}}(\{i\}, \emptyset)$ with $\widehat{z}_i = \widehat{z}_{[k+1]}$, and $\overline{\mathbf{z}}^{(3)} \in \operatorname{argmin} v_{\texttt{relax}}(\{j, j'\}, \emptyset)$ with $\widehat{z}_j = \widehat{z}_{[d-1]}$ and $\widehat{z}_{j'} = \widehat{z}_{[d]}$.

B.2 Recursive Enumeration

This part presents the pseudo-code for recursive enumeration used in STEP 2 of Algorithm 1.

Algorithm 3: Recursive Enumeration in Step 2 of Algorithm 1

Input : sorted $\widehat{\mathbf{w}} = \widehat{\mathbf{w}}(\overline{\mathbf{z}})$, weight limit WL, search range SR, selection number SN
Output : enumeration set Total

Function RE($\widehat{\mathbf{w}}$, *WL*, *SR*, *SN*, *Total*, T_{supp}):
  for $i = 1$ **to** $|SR| - SN + 1$ **do**
    $j \leftarrow \texttt{SR}[i]$;
    $\texttt{MinWeight} \leftarrow \widehat{w}_j + \mathbb{I}(\texttt{SN} > 1) \left[\sum_{t=|\texttt{SR}|-\texttt{SN}+2}^{|\texttt{SR}|} (\widehat{\mathbf{w}}_{\texttt{SR}})_{[t]}\right]$;
    if *MinWeight* $<$ *WL* **then**
      $\texttt{T}_{\texttt{supp}} \leftarrow \texttt{T}_{\texttt{supp}} \cup \{j\}$;
      if $SN = 1$ **then**
        $\texttt{Total} \leftarrow \texttt{Total} \cup \{\texttt{T}_{\texttt{supp}}\}$;
      else
        RE($\widehat{\mathbf{w}}$, $WL - \widehat{w}_j$, $SR[i+1:]$, $SN - 1$, *Total*, T_{supp});
      $\texttt{T}_{\texttt{supp}} \leftarrow \texttt{T}_{\texttt{supp}} \setminus \{j\}$;
  return *Total*;

Call RE($\widehat{\mathbf{w}}$, *WL*, *SR*, *SN*, $\emptyset$, $\emptyset$);

Given a search index s from Algorithm 1, we configure the parameters for the recursive enumeration algorithm: Weight Limit (WL), Search Range (SR), and Selection Number (SN) as follows. For inclusive cut generation, we define the weight limit $\texttt{WL} := q(\overline{\mathbf{z}}) - (\widehat{\mathbf{w}}(\overline{\mathbf{z}}))_{[s]}$, the search range

$\texttt{SR} := [\![s-1]\!]$, and the selection number $\texttt{SN} := k-1$. On the other hand, for exclusive cut generation, the parameters are set to $\texttt{WL} := q(\overline{\mathbf{z}}) - \sum_{i=1}^{s} (\widehat{\mathbf{w}}(\overline{\mathbf{z}}))_{[i]}$, the search range $\texttt{SR} := [\![s+2, d]\!]$, and the selection number $\texttt{SN} := k-s$.

Remark 2. *The Algorithm 3 generates all possible supports of $\texttt{SN}$ elements from index set $\texttt{SR}$ that satisfy outer-if-condition presented in Algorithm 3. If outer-if-condition is satisfied for every possible support, Algorithm 3 is equivalent to brute-force enumeration. Specifically, there are $\binom{|\texttt{SR}|}{\texttt{SN}}$ possible support sets, and each support requires at most $\mathcal{O}(\texttt{SN}^2)$ elementary algebraic operations in Algorithm 3. Therefore, the total computational complexity is $\mathcal{O}\big(\texttt{SN}^2 \cdot \binom{|\texttt{SR}|}{\texttt{SN}}\big)$. The main difference between Algorithm 3 and brute-force enumeration lies in identifying whether index j presented in Algorithm 3 can be added in any possible support.*

B.3 Greedy Algorithm for Computing $v_{\texttt{ub}}$

In this subsection, we present the following greedy algorithm (Algorithm 4) for finding a tighter upper bound to SLRR for all our numerical experiments conducted in section 4. This algorithm was originally proposed by Xie and Deng (2020).

Algorithm 4: Greedy Algorithm for $v_{\texttt{ub}}$ on SLRR (Xie and Deng, 2020)

Input: coefficient matrix $\mathbf{X}$, response vector $\mathbf{y}$, sparsity level k, regularization parameter γ
Output: Upper bound $v_{\texttt{ub}}$ on SLRR
Initialize $S \leftarrow \emptyset$ and $A_S \leftarrow n\gamma I_n$
for $i = 1, \ldots, k$ **do**
  Select $j^* \in \arg\min_{j \in [\![d]\!] \setminus S} \left\{ -\frac{\gamma\left(\mathbf{y}^\top A_S^{-1}\mathbf{x}_j\right)^2}{1+\mathbf{x}_j^\top A_S^{-1}\mathbf{x}_j} \right\}$
  Update $S \leftarrow S \cup \{j^*\}$ and $A_S \leftarrow A_S + \mathbf{x}_{j^*}\mathbf{x}_{j^*}^\top$, $A_S^{-1} \leftarrow A_S^{-1} - \frac{A_S^{-1}\mathbf{x}_{j^*}\mathbf{x}_{j^*}^\top A_S^{-1}}{1+\mathbf{x}_{j^*}^\top A_S^{-1}\mathbf{x}_{j^*}}$
return $v_{\texttt{ub}} \leftarrow \gamma\, \mathbf{y}^\top A_S^{-1}\mathbf{y}$

C Details on Experimental Dataset

Numerical experiments are conducted on three types of datasets, i.e., synthetic dataset, real dataset, and simulation library for Sparse Identification of Non-linear Dynamics (SINDy).

I. Synthetic datasets. Instances from synthetic datasets are set as follows.

Data generation procedure. We follow the generation procedure described in Bertsimas et al. (2020). Given an instance $(\mathbf{X}, \mathbf{Y})$, its design matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ is generated by drawing each row $\mathbf{X}_{i,:}$ independently from a d-dimensional Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, with covariance structure $\mathbf{\Sigma}_{ij} = \rho^{|i-j|}$ for a correlation parameter $\rho \in [0, 1]$. The response vector $\mathbf{Y}$ is therefore generated by $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}$ for some ground-truth $\boldsymbol{\beta}^*$, where $\boldsymbol{\beta}^* \in \{-1, 0, 1\}^d$ is randomly sampled with exactly k non-zero entries, and every component of noise vector $\boldsymbol{\epsilon}$ is i.i.d. generated from a Gaussian distribution $\mathcal{N}(0, \frac{1}{n}\frac{\|\mathbf{X}\boldsymbol{\beta}^*\|_2^2}{\mathrm{SNR}^2})$ with SNR denotes the signal-to-noise ratio.

Parameter settings. We fix the parameters at $k = 10$, $\rho = 0.5$, and $\mathrm{SNR} = 2$. We evaluate four high-dimensional configurations: $(d, n) \in \{(1000, 100), (3000, 150), (5000, 200), (6000, 225)\}$. For

each configuration $(d, n, \gamma, k = 10, \rho = 0.5, \mathrm{SNR} = 2)$, we generate 10 independent instances, denoted by $(\mathbf{X}^{(1)}, \mathbf{Y}^{(1)}), \ldots, (\mathbf{X}^{(10)}, \mathbf{Y}^{(10)})$.

II. Real datasets. We conducted numerical experiments on two real datasets: Toxicity (1203 features, 171 samples) and Period Changer (1177 features, 90 samples) from the UCI Machine Learning Repository Dua and Graff (2017), both of which satisfy the high-dimensional setting. For each dataset, all features are normalized to have unit ℓ_2 norm, and the sparsity level is fixed at $k = 10$.

III. Simulation library for SINDy. Following Bertsimas and Gurnee (2023); Liu et al. (2023), we simulate two dynamic systems from the PySINDy library (de Silva et al., 2020; Kaptanoglu et al., 2022): Lorenz System and Magneto-Hydro-Dynamic (MHD) model. For both systems, the sparsity level is fixed at $k = 8$. This choice reflects a realistic system identification setting where the true dynamics are usually unknown *a priori*. By assuming structural sparsity, we select $k = 8$ as a conservative upper bound on the number of active terms; this ensures the model remains sufficiently expressive to capture the underlying dynamics while maintaining practical identifiability.

Lorenz System. The Lorenz system is a 3-dimensional system governed by the following nonlinear differential equations:

$$dX/dt = -\sigma X + \sigma Y, \quad dY/dt = \rho X - Y - XZ, \quad dZ/dt = XY - \beta Z$$

where we choose the parameters $\sigma = 10, \rho = 2, \beta = 7/3$ to ensure that the generated trajectory does not exhibit excessive chaotic behavior.

Ten trajectories were initialized within the domain $[-1, 1]^3$ and simulated for 5 seconds at a sampling interval of 0.025 seconds, resulting in $n = 200$ samples per trajectory. Each was perturbed with 0.01% Gaussian noise. The candidate library consists of fractional monomials $x^{\alpha_1/3} y^{\alpha_2/3} z^{\alpha_3/3}$ with a maximum total degree of 20/3 (i.e., $\sum_{i=1}^{3} \alpha_i \leq 20$). This yields a feature dimension of $d = 1771$.

MHD model. The 6-dimensional Magneto-Hydro-Dynamic (MHD) model is defined based on the following dynamics:

$$dV_1/dt = 4V_2V_3 - 4B_2B_3, \; dV_2/dt = -7V_1V_3 + 7B_1B_2, \; dV_3/dt = 3V_1V_2 - 3B_1B_2,$$
$$dB_1/dt = 2B_3V_2 - 2V_3B_2, \; dB_2/dt = 5V_3B_1 - 5B_3V_1, \; dB_3/dt = 9V_1B_2 - 9B_1V_2.$$

Trajectories are initialized within the domain $[-0.5, 0.5]^6$ and simulated for a duration of 5 seconds with a sampling interval of every 0.025 seconds, yielding $n = 200$ samples in total. We generate 10 trajectories, each perturbed by 0.01% Gaussian noise. The candidate feature library is constructed using fractional monomials of the form $x^{1/3}$ with a maximum total degree of 8/3. Specifically, we include all terms: $V_1^{\alpha_1/3} V_2^{\alpha_2/3} V_3^{\alpha_3/3} B_1^{\alpha_4/3} B_2^{\alpha_5/3} B_3^{\alpha_6/3}$, subject to $\sum_{i=1}^{6} \alpha_i \leq 8$, resulting in a feature dimension of $d = 3003$.

D Additional Numerical Results

D.1 Assessing the Screening Cut Selection (SCS)

This section evaluates the impact of SCS method through controlled experiments on synthetic dataset. We demonstrate that selecting screening cuts that are simultaneously *effective* and *undominated* is prerequisite for SCPF performance. Using the optimal relaxed support $\widehat{\mathbf{z}} \in \operatorname{argmin} v_{\texttt{relax}}$, we

implement the numerical experiments detailed below. The corresponding results are reported in Figure 6.

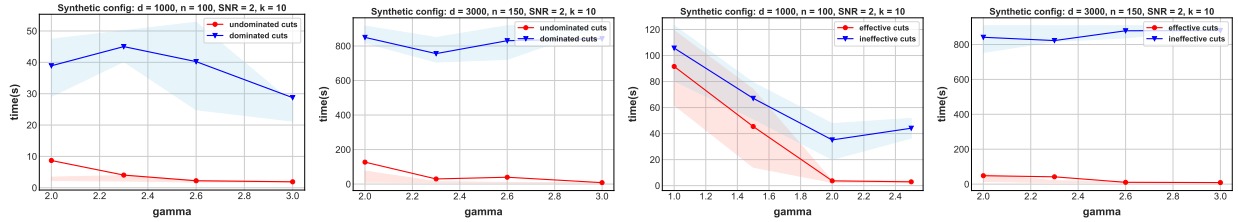


Figure 6. Impact of SCS method on synthetic datasets (details in Appendix C). **Left two panels:** BnB solving times comparing undominated versus dominated cuts. **Right two panels:** BnB solving times comparing effective (length ≤ 2) versus ineffective (length $= 3$) cuts.

Testing undominatedness. To identify the performance impact of undominatedness, we compare the *standard SCPF* (which generates cuts of length $= 1$, equivalent to SSR) against a variant generating *dominated* cuts with length $= 2$. To be more specific, for every index i fixed to 0 ($z_i = 0$) by the standard SCPF, the variant instead adds five randomized exclusive cuts of the form $z_i + z_j \leq 1$ (which are valid but dominated, per Proposition 2). An analogous procedure is applied to inclusive cuts. We report the BnB time required to reach a 1% MIPGap within a 15-minute limit.

Testing effectiveness. To assess the performance impact of effectiveness, we compare the *standard SCPF* (generating undominated cuts with length ≤ 2) against a variant restricted to generating *undominated cuts of length* $= 3$. The latter are theoretically valid but possess significantly lower potential screening ability. Performance is measured by the BnB solving time required to reach a 1% MIPGap within a 15-minute limit.

D.2 Additional SCG cuts

In this section, we report the number of additional SCG cuts generated by Algorithm 2 that are inaccessible to Algorithm 1 with $\sharp_{\texttt{max}} = \infty$, and present the results in Figure 7 over all numerical experiments conducted in section 4.

D.3 Additional experiments on (comp) formulation

This section reports the numerical experiments using (comp) on high-dimensional synthetic datasets with $(d, n) = (6000, 225)$. A comparison of results (see the last column in Figure 2 and Figure 8) reveals that SA exhibits a similar pattern, as different SLRR formulations do not impact the algorithms' screening ability. However, for TRT, the (comp) formulation generally results in higher running time than (SOC). Notably, at $\gamma = 1.5$, nearly every instance with (SOC) formulation achieves optimality, whereas (comp) does not.

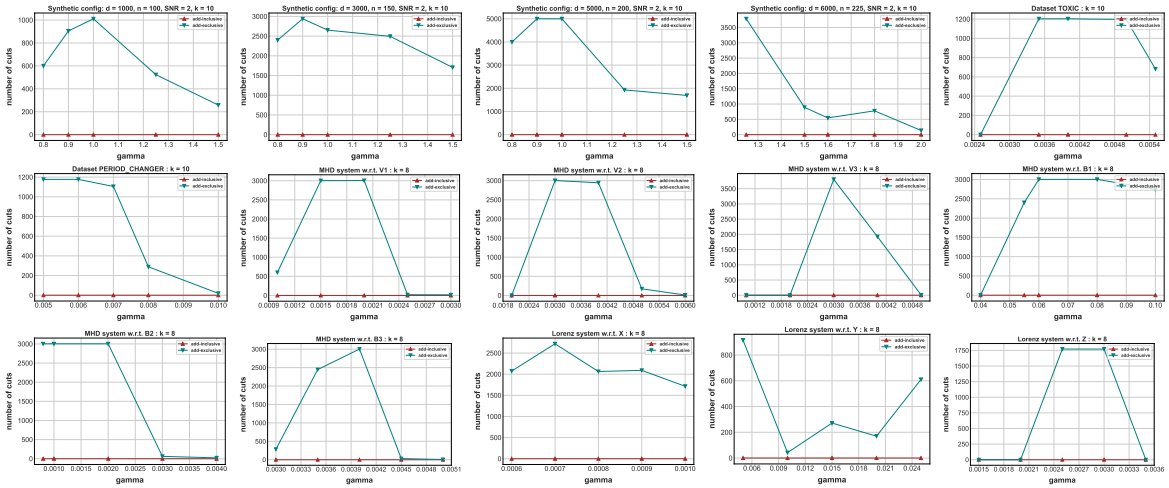


Figure 7. Number of additional SCG cuts generated over all numerical experiments conducted in section 4. Every panel corresponds to the dataset indicated by its label, and each dot represents the average across all tested instances.

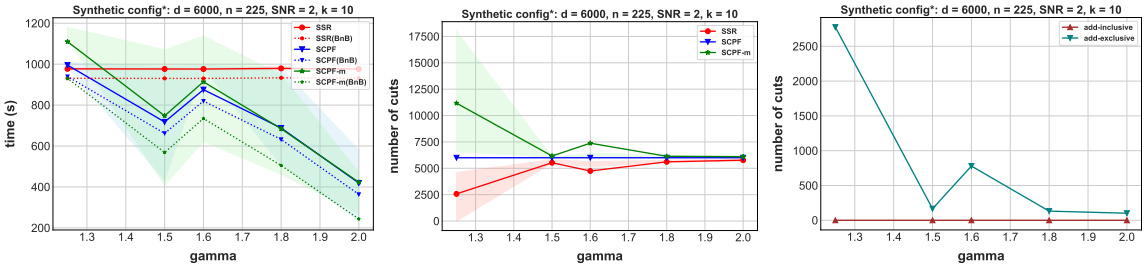


Figure 8. Results of (comp) on synthetic dataset (see Appendix C) with $(d, n) = (6000, 225)$. From left to right: TRT and BnB solving times, SA and the number of additional SCG cuts. The shaded area denotes the inter-quartile range (25th–75th percentiles) of TRT and SA, respectively.