THE DISCRETE DANTZIG SELECTOR: ESTIMATING SPARSE LINEAR MODELS VIA MIXED INTEGER LINEAR OPTIMIZATION

BY RAHUL MAZUMDER* AND PETER RADCHENKO†

Massachusetts Institute of Technology* and University of Southern California†

We propose a new high-dimensional linear regression estimator: the Discrete Dantzig Selector, which minimizes the number of nonzero regression coefficients, subject to a budget on the maximal absolute correlation between the features and residuals. We show that the estimator can be expressed as a solution to a Mixed Integer Linear Optimization (MILO) problem, a computationally tractable framework that delivers provably optimal global solutions. The current state of algorithmics in integer optimization makes our proposal substantially more scalable than the least squares subset selection framework based on integer quadratic optimization, recently proposed in [7] and the continuous nonconvex quadratic optimization framework of [34]. We propose new discrete first-order methods, which, when paired with state-of-the-art MILO solvers, lead to superior upper bounds for the Discrete Dantzig Selector problem for a given computational budget. We demonstrate that the integrated approach, proposed herein, also provides globally optimal solutions in significantly shorter computation times, when compared to off-the-shelf MILO solvers. We demonstrate, both theoretically and empirically, that, in a wide range of regimes, the statistical properties of the Discrete Dantzig Selector are superior to those of popular $\ell_1$-based approaches. Our approach gracefully scales to problem instances up to $p = 10,000$ features with provable optimality, making it, to the best of our knowledge, the most scalable exact variable selection approach in sparse linear modeling at the moment.

1. Introduction. We consider the familiar linear regression framework, with response vector $y \in \mathbb{R}^{n \times 1}$, model matrix $X = [x_1, \ldots, x_p] \in \mathbb{R}^{n \times p}$, regression coefficients $\beta \in \mathbb{R}^{p \times 1}$ and errors $\epsilon \in \mathbb{R}^{n \times 1}$: $y = X\beta + \epsilon$. We assume, unless otherwise mentioned, that the columns of $X$ have been standardized to have zero means and unit $\ell_2$-norm. In many modern statistical applications, the number of variables, $p$, is larger than the number of observations, $n$. In such cases, to carry out statistically meaningful estimation, it is often assumed that the number of nonzero elements in $\beta$ is quite small [27]. The task is to obtain a good estimate, $\hat{\beta}$, which is sparse and serves as a good approximation to the underlying true regression coefficient. Of course, the basic problem of obtaining a sparse model with good data-fidelity is also of interest when the number of observations is comparable to or larger than $p$. In the sparse high-dimensional setting described above, two

---

MSC 2010 subject classifications: Primary 62J05, 62J07; secondary 90C11, 90C26, 90C27

Keywords and phrases: Sparse Linear Regression, $\ell_0$ minimization, Dantzig Selector, Variable Selection, Mixed Integer Optimization, Discrete Optimization, First-order Methods
estimation approaches that have been very popular among statisticians and researchers in related fields are the Lasso [42] and the Dantzig Selector [16]. Both estimators can be expressed as solutions to convex optimization problems, which can be solved using computationally attractive procedures [11, 2, 31, 23], and come with strong theoretical guarantees: see, for example, [16, 8, 13] and the references therein. For reasons that are explained later in this section, the primary motivation for our investigation in this paper is the Dantzig Selector, which is defined as the solution to the following linear optimization problem:

\[
\min_{\beta} \|\beta\|_1 \quad \text{subject to} \quad \|X^T(y - X\beta)\|_\infty \leq \delta.
\]

To distinguish this estimator from our proposed approach, we refer to it as the \(\ell_1\)-Dantzig Selector. This estimator seeks to minimize the \(\ell_1\)-complexity of the coefficient vector, subject to a constraint on the maximal absolute correlation between the corresponding residual vector and the predictors. The tuning parameter \(\delta\) controls the amount of data-fidelity: a small value of \(\delta\) corresponds to a good fit, and a larger value of \(\delta\) leads to heavy shrinkage of the estimated regression coefficients. [16] point out several reasons as to why the feasibility set in (1.1) might serve as a good measure for data-fidelity. In particular, this set is invariant with respect to orthogonal transformations on the data, \((y, X)\). It can also be shown that \(\delta\) controls\(^1\) the residual sum of squares: the latter can be made arbitrarily close to the minimal, least-squares value by decreasing \(\delta\). The \(\ell_1\)-Dantzig Selector, like the Lasso, is used extensively as a model fitting routine to obtain a path of sparse linear models, as the data-fidelity parameter is allowed to vary [31]. Observe that the \(\ell_1\)-Dantzig Selector criterion can be reformulated in terms of the maximum likelihood equations for a linear regression with a Gaussian response. This reformulation allows a natural extension of the approach to more general response distributions [30]. Note that Problem (1.1) can be rewritten as a linear optimization problem, which can be solved quite easily for problems with \(p\) in the order of thousands using modern convex optimization solvers [5, 11]. Under some mild conditions, and even for \(p\) much larger than \(n\), the corresponding estimator achieves a loss within a logarithmic factor of the ideal mean squared error achieved if the locations of the nonzero coordinates were known [16, 8].

The \(\ell_1\)-Dantzig Selector, however, has limitations. In the presence of highly correlated covariates, the estimator tends to choose a dense model, typically bringing in an important variable together with its correlated cousins, which does not significantly hurt the \(\ell_1\)-norm of the corresponding coefficient vector. If one increases the data-fidelity threshold \(\delta\), the selected model becomes sparser, however, in the process, important variables might get left out. This is largely due to the nature of the bias imparted by the \(\ell_1\)-norm, which penalizes both large and small coefficients in a similar fashion. Similar issues also arise in the case of Lasso: see, for example, the discussions in [36, 26, 46, 13], and the

\[^1\] More formally, we have: \(\|X^T(y - X\beta)\|_\infty \geq \frac{\lambda_{\min}(X)}{2np} \left(\|y - X\beta\|^2 - \|y - X\hat{\beta}_{LS}\|^2\right)^{\frac{1}{2}}\), where, \(\lambda_{\min}(X)\) is the minimum nonzero singular value of \(X\), and \(\hat{\beta}_{LS}\) is any unrestricted least-squares solution — see Proposition A.1 in [21] for a proof of this result.
references therein. If the $\ell_0$-pseudo-norm is used instead of the $\ell_1$-norm, the aforementioned problems can be ameliorated: given multiple representations of the model with similar data-fidelity, the $\ell_0$-pseudo-norm will always prefer the most parsimonious representation. In addition, the $\ell_0$-pseudo-norm does not shrink the regression coefficients: once an important variable enters the model, it comes in unshrunk with its full effect, which, in turn, drains the effect of its correlated cousins and naturally leads to a sparser model.

**Our Proposal.** The preceding discussion suggests a natural question: what if we replace $\|\beta\|_1$ in Problem (1.1) with its $\ell_0$-version: $\|\beta\|_0 := \sum_{i=1}^p 1(\beta_i \neq 0)$? This leads to the following discrete optimization problem, which also happens to define the estimator that we propose:

$$\begin{align*}
\text{(1.2)} \quad & \min_{\beta} \|\beta\|_0 \quad \text{subject to} \quad \|X^\top (y - X\beta)\|_\infty \leq \delta.
\end{align*}$$

We refer to the above estimator as the *Discrete* Dantzig Selector. A couple of questions that may be asked at this point are:

- Is the estimator defined via Problem (1.2) computationally *tractable*?
- Does the *Discrete* Dantzig Selector lead to solutions with superior statistical properties, when compared to its $\ell_1$ counterpart?

Addressing these questions and answering them affirmatively is the main focus of this paper.

The objective function in Problem (1.1), represented by $\|\beta\|_1$, may be thought of as a *convexification* of the discrete quantity $\|\beta\|_0$, which counts the number of nonzeros in the regression coefficient vector $\beta$. The corresponding estimator seeks solutions with small $\ell_1$-complexity. While this often leads to sparse solutions, i.e. those with few nonzero coefficients, the sparsity is an indirect consequence of minimizing $\|\beta\|_1$. The *Discrete* Dantzig Selector on the other hand, targets sparsity *directly*, in its very formulation. Problem (1.2) may be interpreted as a procedure that searches among all coefficient vectors $\beta$ with good data-fidelity, i.e. $\{\beta : \|X^\top (y - X\beta)\|_\infty \leq \delta\}$, for a coefficient vector with the maximal sparsity, i.e. the smallest $\|\beta\|_0$. The sparsity of the solution increases as $\delta$ increases. At first glance, many readers may dismiss the estimator defined via Problem (1.2) as computationally intractable, due to the nonconvexity of the objective function in the optimization problem. However, Problem (1.2) can indeed be solved to *global* or *near global* accuracy, using techniques in modern discrete optimization. Specifically, Problem (1.2) can be reformulated as a Mixed Integer Linear Optimization, or *MIL0*, problem — a generalization of linear optimization problems where some of the optimization parameters are binary [6]. Due to the major advances in algorithmic research in *MIL0* over the past fifty years, these methods are widely considered as a mature technology in a subfield of mathematical programming [43, 28]. Algorithmic advances coupled with hardware and software improvements have made *MIL0* problems solvable to *provable* optimality for various problem sizes of practical interest. In this sense, it is perhaps appropriate to perceive *MIL0* as a computationally *tractable* tool.
We note that the view of computational tractability that we adopt here is the ability of a method to provide high quality solutions, with provable optimality certificates, for problem types that are encountered in practice, in times that are appropriate for the applications being addressed [29]. Our approach is aligned with an exciting recent line of work in computational statistics: the use of Mixed Integer Optimization and, more broadly, modern optimization techniques to solve certain classes of discrete problems arising in statistical estimation tasks — see, for example, the recent works [7, 4]. Further background on MILO appears in Section 2.1.

In this paper, we bring together recent advances from diverse areas of modern mathematical optimization methods: first-order continuous optimization and MILO techniques. Thus, we provide a novel unified algorithmic approach that (a) performs favorably over standalone of-the-shelf MILO solvers applicable for Problem (1.2), in terms of obtaining good quality solutions with provable certificates of optimality, and (b) scales gracefully to problem sizes up to \( p = 10,000 \) or even larger. In an extensive series of experiments with synthetic and real data we demonstrate that our unified approach solves, to global optimality, instances of Problem (1.2) with \( n \approx 500, p \approx 100 \) in seconds, underdetermined problems with \( n \approx 900, p \approx 3000 \) in minutes. While it takes marginally longer to provide certificates, or guarantees, of global optimality, the corresponding times are quite reasonable: in all the aforementioned instances the certificates of optimality are available within an hour. Our approach scales to several instances of problems with \( n \leq p \) and \( p \) in the range 5,000 to 10,000, delivering optimal solutions in approximately an hour and proving optimality within at most two days, in all instances. We also find that the statistical properties of our estimates are substantially better than those of computationally friendlier alternatives, like the ℓ1-Dantzig Selector, in terms of both the estimation error and the variable selection properties (assuming that the data is generated from an underlying sparse linear regression model). More generally, for a given amount of the data-fidelity, the proposed method delivers significantly sparser solutions. A detailed description of the results is provided in Section 7.

Examples. To provide the reader with some intuition, we present a set of three examples, which illustrate the differences between the solutions to Problems (1.1) and (1.2). The following simple example\(^2\) demonstrates how the ℓ1-based method Dantzig Selector might experience difficulty in producing a sparse solution in cases where the signal predictors are highly correlated.

Example 1. Let \( p = n + 1 \). Take the first feature as \( x_1 = (1, \tau, \ldots, \tau)^T \), take the \((i, j)\)th entry of the feature matrix, \( X \), as \( x_{ij} = 1 \) \((i = j - 1)\) for \( i = 1, \ldots, n \) and \( j \geq 2 \), and set \( y = x_1 - x_2 \).

The ℓ1-norm of the sparse representation of the response, \( y = x_1 - x_2 \), equals 2. Note that, given the available predictors, the response admits only one other exact representation, \( y = \tau x_3 + \ldots + \tau x_p \). The ℓ1 cost for this dense representation is \( \tau(n - 1) \), which is lower than the corresponding value for the sparse representation when \( \tau \) is small. Consequently, as long as \( \tau(n - 1) < 2 \), both the Lasso and the ℓ1-Dantzig selector select

\(^2\)This example was suggested to us by Emmanuel Candes
the dense representation of the response. Alternatively, \( \ell_0 \)-based methods recover the sparse representation. More specifically, consider the solution to Problem (1.2): if the tuning parameter, \( \delta \), is set below \( \tau/(1+\tau) \), then the estimator exactly recovers the sparse representation of the response.

\[ \text{Discrete Dantzig Selector Coefficient Profiles} \]

\[ \text{\( \ell_1 \)-Dantzig Selector Coefficient Profiles} \]

**Fig 1:** Coefficient profiles for the *Discrete* Dantzig Selector and the \( \ell_1 \)-Dantzig Selector, as a function of the data-fidelity parameter, \( \delta \). The dashed vertical lines in the top row of plots indicate the locations where the number of active variables changes. [Left Panel] corresponds to Example 1; [Middle Panel] corresponds to Example 1'; the “true” nonzero coefficients of +1 and −1 are shown as horizontal starred lines. [Right panel] corresponds to the path for the Diabetes dataset. The numbers overlain on the profiles indicate the different features.

Figure 1 demonstrates the difference between the *Discrete* Dantzig Selector and the \( \ell_1 \)-Dantzig selector, by displaying the coefficient profiles for both methods. Note that the profiles for the *Discrete* Dantzig Selector are constructed in a piece-wise constant fashion, where for each given model size, the displayed coefficients are taken from the solution corresponding to the lowest attainable value of \( \| X^T (y - \hat{X}\beta) \|_\infty \). The left panel of Figure 1 corresponds to Example 1, with \( n = 10, p = 11 \) and \( \tau = 1/2p \), which we slightly modified by adding noise to the response: \( y = x_1 - x_2 + \epsilon \). \( \epsilon \)'s are independently generated from a centered Gaussian distribution, corresponding to the Signal to Noise
Now consider Example 1', which is similar in spirit to Example 1. Here, the first two features, \( x_1 \) and \( x_2 \), are drawn from a centered bivariate Gaussian distribution with correlation 0.7. The remaining \( p - 2 \) features are drawn from an independent Gaussian ensemble. All the features are standardized to have unit \( \ell_2 \)-norm, and the response is generated the same way as in the previous example. The middle panel in Figure 1 displays the corresponding coefficient profiles, with \( n = 10, p = 12 \). The Discrete Dantzig Selector exactly recovers the true model for a wide range of the tuning parameter, \( \delta \). As \( \delta \) is decreased, and noise variables come into the model, their coefficients remain highly shrunk, while the coefficients for the signal variables remain near their true values. On the other hand, the \( \ell_1 \)-Dantzig selector is unable to recover the true model, and produces a large estimation error for all values of the tuning parameter.

The right panel in Figure 1 corresponds to the well known Diabetes dataset [19], where \( n = 442 \) and \( p = 10 \); here all the variables (including the response) were standardized to have unit \( \ell_2 \)-norm and zero mean. Note that, despite some similarities, the sequences of predictors entering the model are different for the two approaches.

Note that instead of formulation (1.2), one may prefer to minimize the the data-fidelity term, subject to a constraint on the number of nonzeros in \( \beta \):

\[
\min_\beta \| X^\top (y - X\beta) \|_\infty \quad \text{subject to} \quad \| \beta \|_0 \leq k.
\]

The framework developed in this paper may be adapted to Problem (1.3). In this paper, however, we focus on Problem (1.2).

Context and Related Work. In a recent work, [7], the authors study the best subset selection problem with the least squares loss, i.e.,

\[
\min_\beta \frac{1}{2} \| y - X\beta \|_2^2 \quad \text{subject to} \quad \| \beta \|_0 \leq k.
\]

[7] show that Problem (1.4) can be written as a mixed integer quadratic optimization (MIQO) problem, i.e. a convex quadratic optimization problem where a subset of the variables are binary. For underdetermined problems (with \( n < p \)) [7] (see Section 5.3.2) point out that MIQO solvers take a long time to prove optimality, by producing matching upper and lower bounds. For instances with \( n = 50 \) and \( p = 1000 \), for example, [7] demonstrate that MIQO can be used to certify locally optimal solutions (and not globally optimal solutions) for Problem (1.4) in a reasonable amount of time. Fortunately, the situation is significantly different for Problem (1.2), which can be expressed as a MILO problem (see Section 2). We observed in our computational experiments that Problem (1.2) is orders of magnitude faster than (1.4) for underdetermined problems, in obtaining solutions with certificates of global optimality, via matching upper and lower bounds. On several randomly generated problem instances\(^3\) with \( n = 100 \) and \( p = 200 \)

---

\(^3\)For a model generated as \( y_i = \mu_i + \epsilon_i, i = 1, \ldots, n \); we define SNR as follows: \( \text{SNR} = \text{Var}(\mu)/\text{Var}(\epsilon) \).

\(^4\)Here, \( y = X\beta + \epsilon \), with \( X \sim \text{MVN}(0, I) \), and the true \( \beta \) had ten nonzero entries, each being equal to one, the errors were mean-zero Gaussian and the SNR was ten.
we found that Problem (1.2) was solved to global optimality, i.e., zero optimality gap with a median time of about 4 minutes. On the same instances, the MIQO formulation for Problem (1.4) took more than 7 hours of computation time to obtain similar optimality certificates. In addition, a MIL0 formulation for Problem (1.2) consumes much less memory than a comparable MIQO formulation for Problem (1.4). The aforementioned computational superiority of MIL0 over MIQO should not come as a surprise. Indeed, it is widely acknowledged in the integer programming community (see, for example, the nice review papers [28, 14]) that current algorithms for MIL0 problems are a much more mature technology than MIQO. This is probably due, in part, to the extensive research that has been carried out in MIL0 over the past half a century. In comparison, algorithmics for MIQO problems is a relatively newer field, and is very much an active area of research — see, for example, the recent developments [44, 9], and the references therein. In fact, a large class of algorithms for MIQO programs rely on the efficiency of MIL0 [44, 28].

Recently, [34] proposed an interesting new approach, MIPGO, which is a discrete optimization framework for globally solving penalized least-squares problems with non-convex quadratic penalty functions (such as MCP or SCAD, for example). MIPGO studies purely continuous non-convex quadratic optimization problems, which are extremely challenging to solve to global optimality and, like the framework in [7], are less scalable than the Discrete Dantzig Selector framework presented herein — see for example [28, 14]. We discuss MIPGO in some more detail in Section 7.1. In the same section we also conduct a numerical comparison and show that, for a given computational budget, the Discrete Dantzig Selector obtains solutions of superior quality for several problem instances with $p$ up to one thousand.

Thus, the superior computational scalability of the corresponding optimization methods forms a principal motivation to study Problem (1.2), as an effective estimation procedure for sparse linear regression. In addition, from a statistical viewpoint, in terms of estimating sparse regression models subject to good data-fidelity, the Discrete Dantzig Selector may be perceived as a natural, interpretable and useful alternative to least squares with variable selection, Problem (1.4) — in the same way as the $\ell_1$-Dantzig Selector may be viewed as an appealing alternative to Lasso.

**Contributions.** Our main contributions can be summarized as follows:

1. We propose a new high-dimensional linear regression estimator: the Discrete Dantzig Selector, which minimizes the number of nonzero regression coefficients, subject to a budget on the maximal absolute correlation between the features and the residuals. We show that the estimator can be expressed as the solution to a MIL0 problem — a computationally tractable framework that delivers provably optimal global solutions. Our proposal, based on mixed integer linear optimization, is more computationally scalable than the recently proposed methods in [7].

\[\text{For example, on problem instances with } n = p \in \{1.5, 2, 2.5, 3\} \times 10^3, \text{ we observed that Problem (1.4) requires at least twice as much memory as that for Problem (1.2), within the first 800 seconds of computation time. The memory requirement for a MIQO for Problem (1.4) with } p = 3000 \text{ was more than 12GB.}\]
(based on mixed integer quadratic optimization) and [34] (based on continuous non-convex quadratic optimization).

2. We develop new discrete first-order methods, motivated by recent algorithmic developments in first-order continuous convex optimization, to obtain high quality feasible solutions for the Discrete Dantzig Selector problem. These solutions are passed onto MILO solvers as warm-starts. Our proposal leads to advantages over the off-the-shelf state-of-the-art integer programming algorithms in terms of (a) obtaining superior upper bounds for a given computational budget and (b) aiding MILO solvers in obtaining tighter lower bounds and hence improved certificates of optimality. Exploiting problem specific information, we also propose enhanced MILO formulations, which further improve the algorithmic performance of MILO solvers.

3. We characterize the statistical properties of the Discrete Dantzig Selector and demonstrate both theoretically and empirically its advantages over ℓ1-based approaches. Our results also apply to approximate solutions for the Discrete Dantzig Selector optimization problem.

4. Our approach obtains optimal solutions for $p \approx 500$ in a few minutes, $p \approx 3000$ within fifteen minutes and for problems with $p = 10,000$ in an hour. Certificates of optimality are obtained at the expense of higher computation times—for instances with $p \approx 500$ they are obtained within half-hour, for $p \approx 3000$ they are achieved around an hour and for $p = 10,000$ the certificates arrive in the range from three to forty hours. To the best of our knowledge, we present herein, the largest problem instances in subset selection for which provably optimal solutions can be obtained.

Roadmap. The remainder of the paper is organized as follows. Section 2 describes the optimization methodology behind the proposed approach, and discusses its connections with the ℓ1-Dantzig Selector optimization problem. In Section 3, the statistical properties of the Discrete Dantzig Selector are analyzed from a theoretical point of view; the results are compared to the ℓ1-Dantzig Selector and the Lasso. The framework of discrete first-order methods is described in Section 4. A detailed discussion of MILO formulations, together with problem specific enhancements, is presented in Section 5. Section 6 gathers numerical results on the computational performance of our algorithms in a variety of settings. An empirical analysis of the statistical properties of the Discrete Dantzig Selector is conducted in Section 7. Some technical details are provided in the Appendix.

2. Overview of the Proposed Methodology. Herein, we introduce and summarize the general aspects of the proposed methodology. Further details and enhancements are provided in Sections 4 and 5.
Fig 2: Typical evolution of a MILP algorithm for Problem (1.2), as a function of time. [Left Panel] displays the progress of Upper Bounds (UB) and Lower Bounds (LB) for the optimal value of the objective function. The upper bounds, which correspond to feasible solutions for Problem (1.2) are seen to stabilize at the global minimum within a few seconds. The lower bounds provide a certificate of how far the current solution might be from the global solution — these bounds progressively improve as the MILP algorithm explores more nodes in the branch and bound tree. Observe that the certificate of global optimality arrives at a later stage, even though the algorithm finds the global solution very quickly. [Right Panel] displays the evolution of the corresponding MILP Optimality Gap (in %), defined as (UB - LB)/UB, with time.

2.1. Mixed Integer Linear Optimization Preliminaries. The general form of a MILP problem is as follows:

$$\min_{\alpha} \quad a^T \alpha$$

subject to

$$A \alpha \leq b$$

$$\alpha_i \in \{0, 1\}, \quad i = 1, \ldots, m_1$$

$$\alpha_j \geq 0, \quad j = m_1 + 1, \ldots, m,$$

where $a \in \mathbb{R}^{m \times 1}$, $A \in \mathbb{R}^{d \times m}$ and $b \in \mathbb{R}^{d \times 1}$ are the problem data, the symbol $\leq$ denotes element-wise inequalities, and we optimize over $(\alpha_1, \ldots, \alpha_m) := \alpha \in \mathbb{R}^m$ containing both discrete $(\alpha_i, i = 1, \ldots, m_1)$ and continuous $(\alpha_i, i = m_1 + 1, \ldots, m)$ variables. For background on MILP, we refer the reader to [6, 32]. The last 25 years, 1991-2015, have witnessed a tremendous improvement in both algorithms and software for MILP — the overall speedup for MILP problems is estimated to be around 450 billion! For more details, we refer the reader to Section A in the Appendix. Some modern integer optimization solvers include CPLEX, GLPK, GUROBI, KNITRO, MOSEK, SCIP — see also [33].

2.2. MILP formulations for the Discrete Dantzig Selector. Assuming without loss of generality that Problem (1.2) has a finite solution, it can be obtained by solving

$$\Gamma_1 := \min_\beta \|\beta\|_0$$

subject to

$$\|X^T(y - X\beta)\|_\infty \leq \delta$$

$$\|\beta\|_\infty \leq M_U,$$

where $\beta$ is the solution to Problem (2.1).
where, \( M_U \) is a large but finite number, also known as the “Big-M” parameter \([6]\) in the parlance of MIL0. We present a MIL0 formulation for Problem (2.1) (and also, Problem (1.2))

\[
\min_{\beta, z} \sum_{i=1}^{p} z_i \\
\text{subject to} \quad -\delta \leq d_j - (q_j, \beta) \leq \delta, \quad j = 1, \ldots, p \\
- M_U z_j \leq \beta_j \leq M_U z_j, \quad j = 1, \ldots, p \\
z_j \in \{0, 1\}, \quad j = 1, \ldots, p,
\]

where the optimization variables are \( z \) (binary) and \( \beta \) (continuous); the problem data consists of \( d := X^\top y = (d_1, \ldots, d_p)^\top \) and \( Q_{p \times p} := X^\top X = [q_1, \ldots, q_p] \). The binary variable \( z_i \) controls whether \( \beta_i \) is zero or not: if \( z_i = 0 \) then \( \beta_i = 0 \) and if \( z_i = 1 \) then \( \beta_i \) is free to vary in the interval \( [-M_U, M_U] \). The objective function \( \sum_{i=1}^{p} z_i \) controls the number of nonzeros in the model. For motivation, we refer the reader to Figure 2, showing the performance of the above MIL0 formulation on the Diabetes dataset \([19]\) with \( n = 442, p = 64 – \) we mean-centered and scaled \( y, x_i \)’s to have unit \( \ell_2 \)-norm.

Formulation (2.2) has intriguing connections to the \( \ell_1 \)-Dantzig Selector, as we discuss below. The binary variables \( z_i \in \{0, 1\} \) in Problem (2.2) can be relaxed into continuous variables \( z_i \in [0, 1] \), leading to:

\[
\min_{\beta, z} \sum_{i=1}^{p} z_i \\
\text{subject to} \quad -\delta \leq d_j - (q_j, \beta) \leq \delta, \quad j = 1, \ldots, p \\
- M_U z_j \leq \beta_j \leq M_U z_j, \quad j = 1, \ldots, p \\
0 \leq z_j \leq 1, \quad j = 1, \ldots, p,
\]

which is a convex relaxation of Problem (2.2). Note that Problem (2.3) is equivalent to:

\[
\Gamma_2 := \min_{\beta} \ \frac{1}{M_U} \sum_{i=1}^{p} |\beta_j| \\
\text{subject to} \quad \|X^\top (y - X\beta)\|_\infty \leq \delta \\
- M_U \leq \beta_j \leq M_U, \quad j = 1, \ldots, p.
\]

Problem (2.4) modifies the \( \ell_1 \)-Dantzig Selector problem:

\[
\Gamma_3 := \min_{\beta} \ \frac{1}{M_U} \|\beta\|_1 \quad \text{subject to} \quad \|X^\top (y - X\beta)\|_\infty \leq \delta,
\]

by including an \( \ell_\infty \) constraint on \( \beta \). It follows from the above that: \( \Gamma_3 \leq \Gamma_2 \leq \Gamma_1 \). The last inequality is typically strict, and, depending upon the data, the gap between the values, as well as between the corresponding optimal solutions, can be substantial, as illustrated in Figure 1. The above discussion provides another viewpoint for explaining the differences between the \( \ell_1 \)-Dantzig Selector and the Discrete Dantzig Selector estimators.

We now present another MIL0 formulation for Problem (1.2), based on Specially Ordered Sets \([6]\). We introduce binary variables \( z_i \in \{0, 1\} \), which satisfy the condition \( (1 –
zm)βm = 0 for all i = 1, ..., p — in other words, if zi = 0, then βi = 0, and if zi = 1, then βi is unconstrained. This condition can be modeled via integer optimization using Specially Ordered Sets of Type 1 [6] (SOS-1). More specifically,

\[(1 - zi)βi = 0 \iff (βi, 1 - zi) : SOS-1\]

for every i = 1, ..., p. This leads to the following MILP formulation for Problem (1.2):

\[
\begin{align*}
\min_{β, z} & \quad \sum_{i=1}^{p} zi \\
\text{subject to} & \quad -δ ≤ d_j - ⟨q_j, β⟩ ≤ δ \quad j = 1, \ldots, p \\
& \quad (β_j, 1 - z_j) : SOS-1 \quad j = 1, \ldots, p \\
& \quad z_j ∈ \{0, 1\} \quad j = 1, \ldots, p.
\end{align*}
\]

Note that in formulation (2.5) the parameter MU does not appear explicitly and can be used if the user does not wish to specify any a-priori value of MU.

We emphasize that MU appearing in formulation (2.2) should not be interpreted as a statistical tuning parameter — it appears from a purely algorithmic viewpoint and, as we saw, has interesting connections to the ℓ1-Dantzig Selector problem. MU might be taken to be arbitrarily large (but finite) to obtain a solution to Problem (1.2). In Section 5 we describe several data driven methods to estimate MU, we also discuss other structured formulations of Problem (2.2), which lead to improved algorithmic performance: they deliver tighter computational lower bounds in smaller amounts of time. We now proceed towards an analysis of the statistical properties of the Discrete Dantzig Selector and investigate its comparative advantages over its ℓ1-counterpart.

3. Statistical Properties: Theory. In this section we study the statistical properties of the Discrete Dantzig estimator. In particular, we characterize its connections with the Best Subset selection estimator in the case of the orthonormal design, we investigate its oracle properties in the classical asymptotic regime and, finally, we analyze its estimation, prediction and variable selection performance in the high dimensional setting. To improve readability, all the technical proofs are presented in Section B in the Appendix.

3.1. Orthonormal Design. Here we assume \( x_j^T x_k = 1 \{j = k\} \). Our goal is to compare and connect the Discrete Dantzig estimator, \( \hat{β} \), which solves the optimization problem (1.2), with the Best Subset selection estimator, \( \hat{β}^{BS} \), which solves (1.4). In particular, we want to understand the relationship between the tuning parameter δ in the Discrete Dantzig optimization problem and the tuning parameter k, which controls the ℓ0 norm of the Best Subset solution. Note that Problem (1.2) does not generally have a unique optimizer, so we use \( \hat{β} \) to refer to just one of the Discrete Dantzig solutions.

Define \( c_j = x_j^T y \) for j = 1, ..., p. To simplify the presentation, and without loss of generality, suppose that the predictors are indexed in such a way that |c1| ≥ |c2| ≥ ... ≥ |cp|. Suppose also that \( c_k ≠ c_{k+1} \), to ensure uniqueness of the Best Subset solution.
Theorem 3.1. 1. The Best Subset selection estimator is uniquely defined as
\[ \hat{\beta}_{BS} = (c_1, \ldots, c_k, 0, \ldots 0)^\top. \]

2. Suppose that \( \delta \in [c_{k+1}, c_k) \). Then, the set of all the Discrete Dantzig solutions is
\[ \left\{ (c_1 + u_1, \ldots, c_k + u_k, 0, \ldots, 0)^\top, |u_j| \leq \delta, j = 1, \ldots, k \right\}. \]

It follows that each Best Subset estimator is a Discrete Dantzig solution for an appropriately chosen \( \delta \). The coefficients of both estimators are obtained by the hard thresholding of the covariances \( c_j \), however, the nonzero coefficients of the Discrete Dantzig estimator are allowed to deviate from the value of the covariance by an amount bounded above by \( \delta \).

3.2. Fixed \( p \) asymptotics. To avoid confusion, we refer to the true coefficient vector as \( \beta^* \). In this subsection we treat the number of predictors, \( p \), as fixed and let the number of observations, \( n \), tend to infinity. The standard assumption for deriving asymptotic results in this setting is that \( \beta^* \) does not depend on \( n \), and \( n^{-1}X^\top X \) converges to a non-singular covariance matrix \( C \). We rewrite the above assumption to be consistent with the scaling \( \|x_j\|_2 = 1 \) for \( j = 1, \ldots, p \), which is used throughout this paper. Thus, we require that \( X^\top X \) converges to \( C \) as \( n \) tends to infinity, and \( \beta^* = n^{1/2}\hat{\beta}^* \), for some fixed vector \( \hat{\beta}^* \). We also impose a standard assumption that \( \epsilon_i \) are i.i.d. with zero mean and finite variance. Given an index set \( J \subseteq \{1, \ldots, p\} \) we write \( X_J \) for the sub-matrix of \( X \) that consists of the columns identified by \( J \). Let \( J^* \) denote the support of the vector \( \beta^* \). We define the Oracle estimator, \( \hat{\beta}^O \), as the least-squares estimator computed using only the true predictors. In other words, the support of \( \hat{\beta}^O \) equals \( J^* \), and \( \hat{\beta}^O_J = (X_J^\top X_J)^{-1}X_J^\top y \).

Theorem 3.2. Let \( \delta \to \infty \) and \( \delta = o(n^{1/2}) \). Suppose that matrix \( C \) is invertible. Then, with probability tending to one,

1. The support of each solution to the Discrete Dantzig optimization problem equals \( J^* \);

2. Both the true coefficient vector, \( \beta^* \), and the Oracle estimator, \( \hat{\beta}^O \), belong to the set of solutions to the Discrete Dantzig optimization problem.
with probability tending to one, every polished estimator coincides with the Oracle estimator.

**Corollary 3.1.** Under the assumptions of Theorem 3.2, equality \( \widehat{\beta}^P = \widehat{\beta}^O \) holds with probability tending to one.

Consequently, estimator \( \widehat{\beta}^P \) satisfies the oracle property in the sense of Fan and Li [20].

### 3.3. High Dimensional Setting.

Here we focus on the case where \( p \) is large, possibly much larger than \( n \). We discuss the properties of the global, as well as approximate, solutions to Problem (1.2). We also comment on the estimator obtained from the closely related Problem (1.3), in which \( \|\beta\|_0 \) is constrained, rather than minimized. We assume that the error terms in the underlying linear model are mean zero Gaussian with variance \( \sigma^2 \) and, as before, use \( \beta^* \) to refer to the true regression coefficient vector. We start with some notation. For every vector \( \theta \in \mathbb{R}^p \) and index set \( J \subseteq \{1, \ldots, p\} \) we write \( \theta_J \) for the sub-vector of \( \theta \) determined by \( J \).

**Definition 3.1.** Given positive integers \( k \) and \( m \), such that \( m \in [k, p-k] \), and a positive \( c_0 \), let

\[
\gamma(k) = \min_{\theta \neq 0, \|\theta\|_0 \leq k} \frac{\|X\theta\|_2}{\|\theta\|_2},
\]

\[
\kappa(k, c_0) = \min_{J_0 \subseteq \{1, \ldots, p\}, |J_0| \leq k} \min_{\theta \neq 0, \|\theta\|_0 \leq k \cap \theta_{J_0} \leq c_0} \frac{\|X\theta\|_2}{\|\theta\|_2},
\]

\[
\kappa(k, c_0, m) = \min_{J_0 \subseteq \{1, \ldots, p\}, |J_0| \leq k} \min_{\theta \neq 0, \|\theta\|_0 \leq k \cap \theta_{J_0} \leq c_0} \frac{\|X\theta\|_2}{\|\theta\|_2}.
\]

In the above \( \theta_{J_{01}} \) stands for \( J_0 \cup J_1 \), where \( J_1 \) identifies the \( m \) largest in magnitude coordinates of \( \theta \) outside of \( J_0 \).

We use \( s^* \) to denote \( \|\beta^*\|_0 \). As we discuss in the next subsection, quantities \( [\kappa(s^*, c_0)]^{-1} \) and \( [\kappa(s^*, c_0, m)]^{-1} \), for \( m \geq s^* \) and \( c_0 \geq 1 \), appear in the error bounds for the Lasso and the original Dantzig selector, while \( [\gamma(2s^*)]^{-1} \) appears in the bounds for Discrete Dantzig Selector. The following result establishes some useful relationships for these quantities.

**Proposition 3.1.** For all positive integers \( k \) and \( m \), with \( m \in [k, p-k] \), and all \( c_0 \geq 1 \) the following holds: \( \gamma(2k) \geq \kappa(k, c_0)/\sqrt{2} \) and \( \gamma(2k) \geq \kappa(k, c_0, m) \).

Recall the setting of Example 1. When \( \tau(n-1) < 2 \), the \( \ell_1 \) methods, such as the original Dantzig selector, fail to recover the sparse representation of the response. Note also that \( \kappa(2, c_0) = \kappa(2, c_0, m) = 0 \), for \( m \geq 2 \) and \( c_0 \geq 1 \). On the other hand, \( \gamma(4) > 0 \)
for $p > 4$, and the Discrete Dantzig Selector succeeds in recovering the correct sparse representation, for every sufficiently small value of tuning parameter, $\delta$.

The following theorem establishes several useful bounds for the Discrete Dantzig Selector.

**Theorem 3.3.** Suppose that $\hat{\beta}$ solves optimization problem (1.2) for $\delta = \sigma \sqrt{2(1 + a) \log p}$, where $a \geq 0$. The following bounds hold with probability bounded below by $1 - (p^a \sqrt{\pi \log p})^{-1}$:

\[
\|\hat{\beta}\|_0 \leq s^* \\
\|\hat{\beta} - \beta^*\|_1 \leq 4 \left[\gamma(2s^*)\right]^{-2} s^* \delta \\
\|\hat{\beta} - \beta^*\|_2^2 \leq 8 \left[\gamma(2s^*)\right]^{-4} s^* \delta^2 \\
n^{-1}\|X(\hat{\beta} - \beta^*)\|_2^2 \leq 8\left[\gamma(2s^*)\right]^{-2} s^* \delta^2.
\]

**Remark.** It follows from the proof of Theorem 3.3 that the above result
(i) holds uniformly over the set $\{\beta^* : \|\beta^*\|_0 \leq s^*\}$;
(ii) also holds for the solution to Problem (1.3) with $k = s^*$.

We now compare the above bounds to those established for the popular $\ell_1$-based approaches. Under the assumed scaling of the predictors, and for every positive integer $m$, such that $m \in [s^*, p - s^*]$, Theorem 7.1 in [8] gives the following error bounds for the $\ell_1$-Dantzig Selector estimator, $\hat{\beta}_{DS}$:

\[
\|\hat{\beta}_{DS} - \beta^*\|_1 \leq 8 \left[\kappa(s^*, 1)\right]^{-2} s^* \delta \\
\|\hat{\beta}_{DS} - \beta^*\|_2^2 \leq 16 \left(1 + \sqrt{s/m}\right)^2 \left[\kappa(s^*, 1, m)\right]^{-4} s^* \delta^2 \\
n^{-1}\|X(\hat{\beta}_{DS} - \beta^*)\|_2^2 \leq 16\left[\kappa(s^*, 1)\right]^{-2} s^* \delta^2.
\]

By Proposition 3.1, the right hand sides of the above inequalities are at least as large as the corresponding bounds in Theorem 3.3. Moreover, the differences in the two sets of bounds can potentially be quite significant. Consider the setting of Example 1 for illustration. The upper bounds in Theorem 3.3 are finite for $p > 4$, while the three bounds in display (3.1) are infinite.

Examining Theorem 7.2 in [8], we conclude that the corresponding error bounds for the Lasso are at least as large as those given in display (3.1). The same result also provides an upper bound on the $\ell_0$-pseudo-norm of the Lasso estimator, $\hat{\beta}_{Lasso}$:

\[
\|\hat{\beta}_{Lasso}\|_0 \leq \frac{64 \phi_{\text{max}}}{\left[\kappa(s^*, 3)\right]^2} s^*,
\]

where $\phi_{\text{max}}$ is the maximum eigenvalue of the matrix $X^T X$. Note that the above bound is infinite in the setting of Example 1. In general, this upper bound is at least 64 times as large as the one for the Discrete Dantzig Selector estimator. We informally summarize the above findings as follows: when compared to the $\ell_1$-based approaches, the Discrete Dantzig Selector satisfies as good or better estimation and prediction error bounds, while achieving significantly higher level of sparsity.
We can sharpen the bounds in Theorem 3.3 by making them dependent on the support of \( \beta^* \). More specifically, given an index set \( J^* \) we define
\[
\tilde{\gamma}(J^*) = \min_{J \subseteq \{1, \ldots, p\}, |J| = 2s^*, J \supseteq J^*} \min_{\theta \neq 0, \theta_{J^c} = 0} \frac{\|X\theta\|_2}{\|\theta\|_2}.
\]
Then, Theorem 3.3 holds with \( \gamma(2s^*) \) replaced by \( \tilde{\gamma}(\{k : \beta^*_k \neq 0\}) \), and the corresponding result is uniform over \( \beta^* \).

The following corollary to Theorem 3.3 shows that the Discrete Dantzig Selector successfully recovers the support of the true coefficient vector, provided the nonzero coefficients are appropriately bounded away from zero. Define \( |\beta^*|_{\min} = \min\{|\beta^*_k|, \beta^*_k \neq 0\} \).

**Corollary 3.2.** If \( |\beta^*|_{\min} > 4\sigma [\gamma(2s^*)]^{-2} \sqrt{(1 + a)s^* \log p} \), then the estimator from Theorem 3.3 exactly recovers the support of \( \beta^* \), with probability bounded below by \( 1 - (p^a/\sqrt{\pi \log p})^{-1} \).

We now consider an estimator \( \hat{\beta} \) that is a feasible solution to the optimization problem (1.2), but not necessarily the optimal solution. Recall that our algorithms produce \( \hat{\beta} \) together with a lower bound on the minimum value of the objective function, \( \|\beta\|_0 \). We denote this lower bound by \( \hat{s}_{LB} \). The next result shows that if the algorithm is stopped when \( \|\hat{\beta}\|_0 \) is within a prespecified multiplicative factor of \( \hat{s}_{LB} \), the bounds from Theorem 3.3 continue to hold after an appropriate adjustment. The corresponding proof follows the argument in the proof of Theorem 3.3, making only minor modifications.

**Theorem 3.4.** Suppose that \( \hat{\beta} \) is a feasible solution to the optimization problem (1.2), corresponding to \( \delta = \sigma \sqrt{2(1 + a) \log p} \), where \( a \geq 0 \), such that \( \|\hat{\beta}\|_0 \leq (1 + \psi)\hat{s}_{LB} \). Then, the following bounds hold with probability bounded below by \( 1 - (p^a/\sqrt{\pi \log p})^{-1} \):
\[
\begin{align*}
\|\hat{\beta}\|_0 & \leq (1 + \psi)s^* \\
\|\hat{\beta} - \beta^*\|_1 & \leq 2(2 + \psi) [\gamma([2 + \psi]s^*)]^{-2} s^* \delta \\
\|\hat{\beta} - \beta^*\|_2 & \leq 4(2 + \psi) [\gamma([2 + \psi]s^*)]^{-4} s^* \delta^2 \\
n^{-1}\|X(\hat{\beta} - \beta^*)\|_2^2 & \leq 4(2 + \psi) [\gamma([2 + \psi]s^*)]^{-2} s^* \delta^2.
\end{align*}
\]

Note that constant \( \psi \) is typically quite small in practice, for example, 0.1 (see the right panel in Figure 2 for an illustration of the evolution of \( \psi \) over time for the Diabetes dataset.) Thus, the corresponding effect on the error bounds is generally minor.

4. **Scaling up MIL0 via Discrete First-Order Methods.** In this section we propose new algorithms, referred to as discrete first-order methods, which deliver good upper bounds for Problem (1.2). It is important to note that we do not use these algorithms to obtain the global solution to Problem (1.2): unlike the MIL0 framework, the proposed methods do not provide lower bounds. Instead, the solutions obtained by our methods are passed to MIL0 solvers as warm-starts. The proposed algorithms are
inspired by recent advances of first-order methods in convex optimization [38, 37, 40],
and can be viewed as their novel non-convex adaptations. We summarize their key
advantages:

- They provide excellent upper bounds to Problem (1.2) with low computational cost,
time and memory requirements.
- MIL0 solvers accept these solutions as warm-starts and consequently improve upon
them. This hybrid approach outperforms the stand-alone capabilities of an off-the-shelf
MIL0 solver, producing high quality upper bounds in amounts of time that are orders
of magnitude smaller.
- The solutions obtained can be used to improve the overall run-time of MIL0 solvers,
including certificates of global optimality.

We validate our proposed methods on a variety of synthetic and real-data experiments.

4.1. Discrete First-Order Methods. Problem (1.2) involves the minimization of a dis-
tinuous objective function over a polyhedral set. Thus, it is not directly amenable
to simple proximal gradient type algorithms [37, 38, 7]. We propose two algorithms:
Algorithm 1 (see Section 4.1.1) and Algorithm 2 (see Section 4.1.2), both of which
can be used as stand-alone solvers for obtaining good quality upper bounds to Prob-
lem (1.2). We also present a hybrid method, Algorithm 3, which combines the strengths
of both Algorithms 1 and 2, by using the solution obtained from Algorithm 1 as an
initialization to Algorithm 2. In our experiments Algorithm 3 showed the best empirical
performance. Section 6 presents numerical results illustrating the performance of
our framework. We emphasize that Algorithms 1—3 only provide good upper bounds,
they do not certify the quality of solutions via lower bounds. A main purpose of these
algorithms is to provide good quality upper bounds to initialize the MIL0 solvers — the
latter, in turn, are often found to improve upon the upper bounds obtained from these
first-order algorithms.

4.1.1. The Variable Splitting Method. We present our first discrete first-order method
based on the Alternating Direction Method of Multipliers [3] — an old method in nonlin-
erar optimization, recently popularized by [12], mainly in the context of convex optimiza-
tion. We choose this method because of its simplicity and good performance in practice.
To apply this algorithm, we decouple the feasible set \( \{ \beta : \| \mathbf{X}^T (\mathbf{y} - \mathbf{X}\beta) \|_\infty \leq \delta \} \) and
the discontinuous function \( \beta \mapsto \| \beta \|_0 \). Observe that Problem (1.2) can be equivalently
rewritten as:

\[
(4.1) \quad \min_{\alpha, \beta} \| \beta \|_0 \quad \text{subject to} \quad \| \mathbf{X}^T (\mathbf{y} - \mathbf{X}\alpha) \|_\infty \leq \delta, \quad \alpha = \beta.
\]

We consider the Augmented Lagrangian given by:

\[
(4.2) \quad \mathcal{L}_\lambda(\beta, \alpha; \nu) := \| \beta \|_0 + \frac{\lambda}{2} \| \beta - \alpha \|_2^2 + \langle \nu, \alpha - \beta \rangle
\]
for some value of $\lambda > 0$, where, $\nu$ may be thought of as a “dual” variable\textsuperscript{6} that along with $\lambda$ controls the proximity between $\alpha$ and $\beta$. The Alternating Direction method of multipliers leads to the following update sequence:

\begin{align}
(4.3) & \quad \beta_{k+1} \in \arg\min_{\beta} L_\lambda(\beta, \alpha_k; \nu_k) \\
(4.4) & \quad \alpha_{k+1} \in \arg\min_{\alpha} L_\lambda(\beta_{k+1}, \alpha; \nu_k) \text{ subject to } \|X^T(y - X\alpha)\|_\infty \leq \delta \\
(4.5) & \quad \nu_{k+1} = \nu_k + \lambda (\alpha_{k+1} - \beta_{k+1}).
\end{align}

Step (4.3) can be performed via a hard thresholding operation\textsuperscript{18} and (4.4) involves a simple projection onto the polytope $\{\alpha : \|X^T(y - X\alpha)\|_\infty \leq \delta\}$, which can be done efficiently, as detailed in Section C.1.

**Algorithm 1**

1. Input $(\beta_1, \alpha_1, \nu_1)$, choose $\lambda > 0$, and repeat the following steps until convergence.
2. Update $(\beta_k, \alpha_k, \nu_k)$ to $(\beta_{k+1}, \alpha_{k+1}, \nu_{k+1})$ via (4.3)–(4.5).
3. Stop if $\|\beta_{k+1} - \beta_k\|_2 \leq \tau_1 \|\beta_k\|_2$ and $\|\beta_k - \alpha_k\|_2 \leq \tau_2 \max\{\|\beta_k\|_2, \|\alpha_k\|_2\}$; otherwise go to Step 2.

We found Algorithm 1 to work quite well in our experiments. The algorithm may be sensitive to the choice of $\lambda$ — affecting the solution and the time until convergence. We recommend using multiple values of $\lambda$, and choosing the best solution among them. In Section 4.1.3 we address these shortcomings and describe improvements that work better in practice.

### 4.1.2. Sequential Linear Optimization

We now describe another nonlinear optimization algorithm for obtaining upper bounds for Problem (1.2), motivated by ideas popularly used in nonconvex penalized regression (see, for example, [36] and references therein). Let us consider a family of nonconvex functions, $\rho_\gamma(|\beta|)$, parametrized by $\gamma \in (0, \bar{\gamma}]$, such that $\gamma = \bar{\gamma}$ corresponds to $\rho_\gamma(|\beta|) = |\beta|$, and, as $\gamma$ decreases to 0, $\rho_\gamma(|\beta|)$ becomes a progressively better approximation to $1(\beta \neq 0)$. In other words,

\begin{equation}
(4.6) \quad \|\beta\|_0 = \lim_{\gamma \to 0^+} \sum_{i=1}^p \rho_\gamma(|\beta_i|).
\end{equation}

We make the following assumption about $\rho_\gamma(\cdot)$:

**Assumption:** $\rho_\gamma(\beta)$ is symmetric in $\beta$ around zero. For every $\gamma$, the map $|\beta| \mapsto \rho_\gamma(|\beta|)$ is concave and differentiable on $(0, \infty)$.

\textsuperscript{6}Following the terminology in [12], if instead of $\|\beta\|_0$ we had a convex function, then $\nu$ would be a dual variable, and its corresponding update step (4.5) would be the dual update. We will with a slight abuse of terminology use the term “dual” here.

\textsuperscript{7}Here, $\tau_1, \tau_2$ are tolerances for convergence, typically, taken to be equal and set to $10^{-4}$. 


Some popular choices of $\rho_\gamma(\cdot)$ are $\rho_\gamma(t) = \log(t+\frac{1}{\gamma})/\log(t+1)$ and $\rho_\gamma(t) = t^\gamma$ for $t \geq 0$. We refer the reader to [36] (and references therein) for more context and examples of nonconvex penalty functions used in sparse linear regression.

We propose to compute good upper bounds for the following continuous nonconvex optimization problem:

\begin{equation}
\min_{\beta} h(\beta) := \sum_{i=1}^{p} \rho_\gamma(\|\beta_i\|) \quad \text{subject to} \quad \|X^T(y - X\beta)\|_\infty \leq \delta,
\end{equation}

especially for $\gamma \approx 0$. In light of (4.6), this leads to good upper bounds for Problem (1.2). Note that, the concavity of $|\beta| \mapsto \rho_\gamma(|\beta|)$ leads to the following upper bound:

\begin{equation}
h(\beta) = \sum_{i=1}^{p} \rho_\gamma(|\beta_i|) \leq \sum_{i=1}^{p} \rho_\gamma(|\tilde{\beta}_i|) + \sum_{i=1}^{p} \rho'_\gamma(|\tilde{\beta}_i|) \left(|\beta_i| - |\tilde{\beta}_i|\right) := \tilde{h}(\beta; \tilde{\beta}),
\end{equation}

where $\rho'_\gamma(\cdot)$ denotes the derivative of $|\beta| \mapsto \rho_\gamma(|\beta|)$, with the convention that $\rho'_\gamma(0) = \infty$ if the derivative is unbounded as $|\beta| \to 0^+$. Inequality (4.8) suggests that we sequentially minimize an upper bound to the objective function in (4.7). This leads to the following iterative scheme:

\begin{equation}
\beta^{k+1} \in \arg \min_{\beta} \sum_{i=1}^{p} \rho'_\gamma(|\beta_i^k|) |\beta_i| \quad \text{subject to} \quad \|X^T(y - X\beta)\|_\infty \leq \delta,
\end{equation}

where we assume, without loss of generality, that $\beta^1$ is feasible for Problem (4.9). The above sequential approximation of the function $h(\beta)$ is similar to the popular reweighted $\ell_1$-minimization method, used in signal processing [15] for sparse linear model estimation with the least squares loss.

We now present a simple and, to our knowledge, new finite time convergence rate of the algorithm. We introduce the following quantity:

\begin{equation}
\Delta(\theta) := \min_{\beta} \sum_{i=1}^{p} \rho'_\gamma(|\theta_i|) (|\beta_i| - |\theta_i|) \quad \text{subject to} \quad \|X^T(y - X\beta)\|_\infty \leq \delta,
\end{equation}

which we use to define a first-order stationary point for Problem (4.7).

**Definition 4.1.** Suppose $\hat{\theta}$ is feasible for Problem (4.7). We say that $\hat{\theta}$ is a first-order stationary point for Problem (4.7), if $\Delta(\hat{\theta}) = 0$. $\hat{\theta}$ is said to be an $\phi$ accurate first-order stationary point if $\Delta(\hat{\theta}) \geq -\phi$.

The following theorem (for a proof see Section C.4) presents a finite time convergence rate analysis of the above algorithm to a first-order stationary point.

**Theorem 4.1.** Consider Problem (4.7) for a fixed $\gamma > 0$, with the above assumption on $\rho_\gamma(\cdot)$ in place. The update sequence $\beta^k$, defined via (4.9), leads to a decreasing sequence...
of objective values for Problem (4.7): \( h(\beta^{k+1}) \leq h(\beta^k) \) for all \( k \geq 1 \). In addition, for every \( \mathcal{K} > 0 \) we have the following finite-time convergence rate:

\[
\min_{1 \leq k \leq \mathcal{K}} \left\{ -\Delta(\beta^k) \right\} \leq \frac{1}{\mathcal{K}} \left( h(\beta^1) - \hat{h} \right),
\]

where the sequence of objective function values satisfies \( h(\beta^k) \downarrow \hat{h} \) as \( k \to \infty \).

Theorem 4.1 implies that for any \( \phi > 0 \), it takes at most \( \mathcal{K} = O\left( \frac{1}{\phi} \right) \) many iterations to reach a \( \phi \)-accurate first-order stationary point, i.e., there exists a \( 1 \leq k^* \leq \mathcal{K} \) such that \( \Delta(\beta^{k^*}) > -\phi \). The sequence \( \beta^k \) leads to an estimate \( \hat{\beta}_\gamma \), an upper bound Problem (4.7), for a fixed \( \gamma \). We recommend taking a sequence of decreasing values of \( \gamma \in \{ \gamma_1, \ldots, \gamma_N \} \), where \( \gamma_i > \gamma_{i+1} \), and using \( \hat{\beta}_{\gamma_i} \) as a warm-start for solving Problem (4.9) for a smaller value of \( \gamma = \gamma_{i+1} \). In our numerical experiments, this continuation strategy delivered better upper bounds to Problem (1.2). We summarize the aforementioned method as Algorithm 2.

**Algorithm 2**

1. Take a decreasing sequence of \( \gamma \) values \( \{ \gamma_1, \ldots, \gamma_N \} \); initialize with \( \beta^0 = 0 \); and fix a value of \( \text{Tol} = 10^{-5} \) (say). Set \( \kappa = 1 \) and \( \gamma = \gamma_{\kappa} \).

2. Use the update sequence rule (4.9) until some convergence criterion is met: \( (-\Delta(\beta^k)) < \text{Tol} \). Let \( \hat{\beta}_\gamma \) denote the estimate of \( \beta \), upon convergence.

3. Set \( \kappa \leftarrow \kappa + 1 \), \( \gamma = \gamma_{\kappa} \) and \( \beta^0 \leftarrow \hat{\beta}_\gamma \). If \( \kappa \leq N \), then goto Step 2. If \( \kappa > N \), exit with \( \beta \) as an upper bound to Problem (1.2).

The linear optimization Problem (4.9) can be solved quite efficiently using simplex methods. For larger problems, i.e. \( p \) larger than a few thousand, we recommend using modern first-order method as described in Section C.2. Since Algorithm 2 requires solving several instances of related problems of the form (4.9), the warm-start capabilities of simplex methods and first-order methods become particularly attractive.

4.1.3. **Algorithm 3: Combining the Strengths of Algorithm 1 and Algorithm 2.** In our empirical studies we observed that Algorithm 1 is more effective in obtaining good upper bounds than Algorithm 2 for a given time limit. Algorithm 2, on the other hand, has stronger convergence guarantees than Algorithm 1. Algorithm 1 leads to an estimate of \( \beta \) that is sparse but approximately satisfies\(^8\) the feasibility constraint of Problem (1.2). Algorithm 2, in contrast, leads to solutions that are both sparse and feasible — these advantages make Algorithm 2 an important tool in our framework. We, thus, propose to combine the best features of Algorithms 1,2 to develop a hybrid variant: Algorithm 3, which we recommend to use in practice. Algorithm 3 is quite simple: it uses the solution obtained from Algorithm 1, say, \( \hat{\beta}^{(1)} \), to create a set \( \mathcal{I} \subset \{ 1, \ldots, p \} \), which includes the

\(^8\)This is because Algorithm 1 delivers a pair, \( \alpha, \beta \), which are approximately equal: \( \alpha \approx \beta \); \( \alpha \) is feasible for Problem (1.2) but need not be exactly sparse; \( \beta \), on the other hand, is sparse but approximately feasible.
nonzeros in $\hat{\beta}^{(1)}$, and then applies Algorithm 2 on this set $I$ — the details of this method are presented in Section C.5.

5. Advanced MIL0 Formulations and Certificates of Optimality. This section is dedicated to enhancements of the basic Discrete Dantzig Selector formulation (2.2), presented in Section 2.2. These are particularly useful in delivering tighter lower bounds, thereby providing certificates of global optimality in shorter times.

Observe that, unlike (2.2), Problem (2.5) does not contain any parameter $M_U$ in its formulation. Problem (2.5) may be preferred over Problem (2.2) when the different nonzero values of $|\hat{\beta}_i|$’s have widely different amplitudes. In general, however, we found empirically that the algorithmic performances of formulations (2.5) and (2.2) are comparable. The MIL0 formulations (2.2) and (2.5) are found to work quite well in obtaining good upper bounds for up to $p = 10,000$, once they are warm-started via the discrete first-order methods described in Section 4.1. If additional problem-specific information, which we dub “intelligence”, is supplied to the MIL0 formulations (2.2) and (2.5), the results are found to improve substantially — better lower bounds are obtained in significantly shorter times, thereby proving optimality much faster — as shown in Section 6.2.

More specifically, we use the term “intelligence” to broadly refer to two components: (a) providing an advanced warm-start to the MIL0 solver, obtained via our discrete first-order methods; and (b) arming the MIL0 solver with information in the form of interval bounds on the regression coefficients $\beta_j$, predictions $x_i^\top \beta$, and also bounds on $\|\beta\|_1$ and $\|X\beta\|_1$. Of course, the resulting formulation, with these additional bounds, should lead to a solution for Problem (1.2). We, thus, present the following structured version of formulation (2.5):

$$\min_{\beta, z} \sum_{i=1}^p z_i$$
subject to

$$-\delta \leq d_j - \langle x_j, \xi \rangle \leq \delta \quad j = 1, \ldots, p$$

$$(\beta_j, 1 - z_j) : \text{SOS-1} \quad j = 1, \ldots, p$$

$$z_j \in \{0, 1\} \quad j = 1, \ldots, p,$$

$$\xi = X\beta$$

$$-M_U^j \leq \beta_j \leq M_U^j \quad j = 1, \ldots, p$$

$$-M_U^{\xi, i} \leq \xi_i \leq M_U^{\xi, i} \quad i = 1, \ldots, n$$

$$\|\beta\|_1 \leq M_\ell$$

$$\sum_{i=1}^n |\xi_i| \leq M^\xi_\ell,$$

where the optimization variables are $\beta \in \mathbb{R}^p, z \in \{0, 1\}^p, \xi \in \mathbb{R}^n$, and the parameters $M_U^j, M_U^{\xi, i}, M_\ell, M^\xi_\ell$ control, respectively, upper bounds on $|\beta_i|$, $|\langle x_i, \beta \rangle|$, $\|\beta\|_1$ and $\|X\beta\|_1$. 
Problem (5.1) is equivalent to the following constrained version of Problem (1.2):

\[
\begin{align*}
\min_{\beta} \quad & \|\beta\|_0 \\
\text{subject to} \quad & \|X^T(y - X\beta)\|_\infty \leq \delta \\
& -M_{ij}^U \leq \beta_j \leq M_{ij}^U \quad j = 1, \ldots, p \\
& |\langle x_i, \beta \rangle| \leq M_{ij}^{\xi} \quad i = 1, \ldots, n \\
& \|\beta\|_1 \leq M_{\ell} \\
& \|X\beta\|_1 \leq M_{\ell}^{\xi}.
\end{align*}
\]

(5.2)

Section 5.1 presents several strategies to compute these parameters such that a solution to Problem (5.2) is also a solution to Problem (1.2).

We present a few variations of the formulation presented in (5.1) that might be preferred from a computational viewpoint, depending upon the problem instance under consideration. For large values of \(p\) and \(n\) (approximately a few thousand), the constraints appearing in (5.1a) and (5.1c) may be replaced by:

\[-\delta \leq d_j - \langle \tilde{x}_j, \xi \rangle \leq \delta, \quad \xi = \tilde{X}\beta,
\]

where, \(\tilde{X}^T\tilde{X} = X^TX\) and \(\tilde{X}\) is triangular — this leads to a sparse representation of the constraints appearing in (5.1).

When \(n\) is large and \(p\) is smaller, it is useful to perform a variable reduction by removing the variable \(\xi\) from (5.1). This will replace constraint (5.1a) by \(-\delta \leq d_j - \langle a_j, \beta \rangle \leq \delta\), and constraints (5.1c) and (5.1e) will be dropped. Instead of the SOS-1 constraints (5.1b) one can also use binary variables:

\[-M_{ij}^U z_j \leq \beta_j \leq M_{ij}^U z_j, \quad z_j \in \{0, 1\}, \quad j = 1, \ldots, p,
\]

in which case, the constraints (5.1d) are no longer required. We note that formulation (5.1) is an optimization problem with many more continuous variables than formulation (2.5). This implies that the MILP solver needs to do more work at every node, by solving larger convex linear programs. However, the advantage is that the resulting formulation is more structured, and, thus, tighter lower bounds may be obtained by exploring fewer nodes.

Section 6.2 presents computational results illustrating the performance of the above computational framework.

5.1. Specification of Parameters. We present herein, several data-driven ways to compute the parameters in formulation (5.1). We note that [7] discuss structured MIQO formulations for Problem (1.4). The methods appearing in [7] for estimating the parameters, however, are quite different from those studied herein. In particular, [7] rely crucially on being able to compute analytic expressions for least squares solutions for a given subset size—such expressions are not available for Problem (1.2).
5.1.1. Specification of Parameters via Linear Optimization. We present several methods based on linear optimization that can be used to estimate the parameters appearing in Problem (5.1), in such a way that these estimates lead to \( \hat{\beta} \), a solution to Problem (1.2).

**Bounds on \( \hat{\beta}_i \)'s.** Consider the following pair of convex linear optimization problems:

\[
\begin{align*}
\mu_i^+ := & \max_{\beta} \beta_i \\
& \text{subject to } \|X^T(y - X\beta)\|_\infty \leq \delta,
\end{align*}
\]

\[
\begin{align*}
\mu_i^- := & \min_{\beta} \beta_i \\
& \text{subject to } \|X^T(y - X\beta)\|_\infty \leq \delta,
\end{align*}
\]

for \( i = 1, \ldots, p \). Note that \( \mu_i^+ \) and \( \mu_i^- \) provide upper and lower bounds on \( \hat{\beta}_i \) for every \( i \). \( \mu_i^+ \) is typically a strict upper bound to \( \hat{\beta}_i \), because (5.3) does not account for the fact that solutions to Problem (1.2) are sparse. Similarly, \( \mu_i^- \) is a lower bound to \( \hat{\beta}_i \), and it is easy to see that \( M_i^U = \max\{|\mu_i^+|, |\mu_i^-|\} \) is an upper bound to \( |\hat{\beta}_i| \). Note that solutions to Problem (5.3) are finite only if the feasible set is bounded. If \( n > p \) and if the entries of \( X \) are drawn from a continuous probability measure, then the bounds are finite with probability one. The above bounds can be made tighter by using information about upper bounds on Problem (1.2) as obtained via the discrete first-order methods. We describe such methods in Section C.6 (Appendix). Once upper bounds on \( |\hat{\beta}_i| \), i.e. \( M_i^U \), are obtained, they can be used to compute bounds on \( \|\hat{\beta}\|_1 \) and \( \|\hat{\beta}\|_\infty \) as follows:

\[
\|\hat{\beta}\|_1 \leq M_U = \max_{i=1,\ldots,p} M_i^U \quad \text{and} \quad \|\hat{\beta}\|_\infty \leq \sum_{i=1}^{\alpha_0} M_i^{(i)},
\]

where, \( \alpha_0 \) denotes an upper bound to Problem (1.2) and \( M_U^{(1)} \geq M_U^{(2)} \geq \ldots \geq M_U^{(p)} \).

**Bounds on \( \langle x_i, \hat{\beta} \rangle \)'s.** We present a technique to obtain bounds on \( \langle x_i, \hat{\beta} \rangle \), by solving the following pair of linear optimization problems:

\[
\begin{align*}
v_i^+(\alpha_0) := & \max_{\beta} \langle x_i, \beta \rangle \\
& \text{subject to } \|X^T(y - X\beta)\|_\infty \leq \delta \\
& \|\beta\|_\infty \leq M_U \\
& \|\beta\|_1 \leq M_U \alpha_0,
\end{align*}
\]

\[
\begin{align*}
v_i^-(\alpha_0) := & \min_{\beta} \langle x_i, \beta \rangle \\
& \text{subject to } \|X^T(y - X\beta)\|_\infty \leq \delta \\
& \|\beta\|_\infty \leq M_U \\
& \|\beta\|_1 \leq M_U \alpha_0,
\end{align*}
\]

for every \( i = 1, \ldots, n \).

Analogous to the bounds derived via Problem (5.3), it is also possible to compute more conservative bounds on \( \langle x_i, \hat{\beta} \rangle \), by dropping the constraints \( \|\beta\|_\infty \leq M_U \) and \( \|\beta\|_1 \leq M_U \alpha_0 \) in Problem (5.4). This gives nontrivial bounds even for the under-determined \( n < p \) case, as long as \( X \) has rank \( n \) (this is in contrast with the bounds from Problem (5.3) being loose when \( n < p \)). It is also possible to estimate bounds on \( \langle x_i, \hat{\beta} \rangle \) by including an additional constraint: \( \|X\beta\|_\infty \leq M_U^i \), and using an iterative
method as described in Section C.6 (see Step-1— Step-4) while computing bounds on the regression coefficients.

The quantity \( v_i = \max \{ v_i^+(\alpha_0), -v_i^-(\alpha_0) \} \) provides an upper bound to \( |\langle x_i, \hat{\beta} \rangle| \). In particular, this leads to the following upper bounds: \( \|X\hat{\beta}\|_\infty \leq \max_{i=1,\ldots,n} v_i \) and \( \|X\hat{\beta}\|_1 \leq \sum_{i=1}^n v_i \), which can be thought of as a completely data-driven method to estimate bounds appearing in (5.1).

**Computational Cost.** Computing the quantities appearing in (5.3), (C.22) and (5.4) requires solving at least \( 2(p+n) \) linear optimization problems. However, these individual problems are quite simple to parallelize and they need to be solved once, before proceeding to solve Problem (5.1). These linear optimization problems can be solved by simplex based linear problem solvers quite easily for \( p \) in the lower thousands (typically less than a minute with Gurobi’s simplex solver).

### 5.1.2. Specification of Parameters from warm-starts

We present herein, simple practical methods to compute the parameter values by using good upper bounds to Problem (1.2). Let \( \hat{\beta}^0 \) denote a solution that corresponds to a good upper bound to Problem (1.2). \( \hat{\beta}^0 \) can be obtained from Algorithm 3, for example. One can also use the solution obtained from Algorithm 3 as a warm-start to Problem (2.5) and allow it to run for a few minutes — the resulting estimate may be used as \( \hat{\beta}^0 \). The parameters appearing in the bounds can be based on \( \hat{\beta}^0 \), as follows. To be on the conservative side, we recommend setting the same bound for all the \( \mathcal{M}_i^U \)'s, for example, they can all be assigned the value \( \tau \|\hat{\beta}^0\|_\infty \); similarly, a conservative bound for all the \( \mathcal{M}_i^{\xi,i} \)'s is given by \( \tau \|X\hat{\beta}^0\|_\infty \). In addition, we can set \( \mathcal{M}_\ell = \min \left\{ \tau \|\hat{\beta}^0\|_0, \tau \|\hat{\beta}^0\|_\infty \right\} \) and \( \mathcal{M}_\xi^\ell = \tau \|X\hat{\beta}^0\|_\infty \), for some value of \( \tau \in \{1.5, 2\} \).

The method described above leads to parameter specific bounds as a simple by-product of our general algorithmic framework and requires no additional computation as the methods in Section 5.1.1. The bounds in Section 5.1.1 are exact since they are implied by the bounds from Problem (1.2).

### 6. Numerical Experiments: Algorithmic Performance

In this section, we report extensive numerical experiments that demonstrate: (a) the usefulness of the discrete first-order methods (Section 4.1) in obtaining good quality upper bounds, especially when they are used to provide warm-starts to MILP solvers — this is shown in Section 6.1; and (b) how advanced warm-starts, coupled with the enhanced formulations presented in Section 5.1, can be used to improve the overall run-time for off-the-shelf MILP solvers, when proving global optimality for the Discrete Dantzig Selector problem — this is shown in Section 6.2.

All computations were carried out on Columbia University’s high performance computing (HPC) facility, [http://hpc.cc.columbia.edu/](http://hpc.cc.columbia.edu/), on the Yeti cluster computing environment. The discrete first-order methods were implemented in MATLAB 2014a, and
### Type-1 \((n = 100, p = 1000)\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time (in secs)</th>
<th>With Warm</th>
<th>Vanilla</th>
<th>Quality of Upper Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\delta_1)</td>
<td>90 (*)</td>
<td>0</td>
<td>42.85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0</td>
<td>42.85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>28.57</td>
<td></td>
</tr>
<tr>
<td>(\delta_2)</td>
<td>120</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(\delta_3)</td>
<td>57 (*)</td>
<td>0</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0</td>
<td>86.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>26.66</td>
<td></td>
</tr>
<tr>
<td>(\delta_4)</td>
<td>120</td>
<td>3.12</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>210 (*)</td>
<td>0</td>
<td>15.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>15.62</td>
<td></td>
</tr>
</tbody>
</table>

### Type-2 \((n = 300, p = 1000)\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time (in secs)</th>
<th>With Warm</th>
<th>Vanilla</th>
<th>Quality of Upper Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\delta_1)</td>
<td>120</td>
<td>6.66</td>
<td>146.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>132 (*)</td>
<td>0</td>
<td>73.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(\delta_2)</td>
<td>50 (*)</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0</td>
<td>214.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>14.28</td>
<td></td>
</tr>
<tr>
<td>(\delta_3)</td>
<td>35 (*)</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0</td>
<td>29.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>25.92</td>
<td></td>
</tr>
<tr>
<td>(\delta_4)</td>
<td>40 (*)</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0</td>
<td>73.44</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>23.43</td>
<td></td>
</tr>
</tbody>
</table>

### Type-3 \((n = 600, p = 2000)\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time (in secs)</th>
<th>With Warm</th>
<th>Vanilla</th>
<th>Quality of Upper Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\delta_1)</td>
<td>500</td>
<td>11.11</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>875 (*)</td>
<td>0</td>
<td>137.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>950</td>
<td>0</td>
<td>33.33</td>
<td></td>
</tr>
<tr>
<td>(\delta_2)</td>
<td>500</td>
<td>7.14</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>530 (*)</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>950</td>
<td>0</td>
<td>28.57</td>
<td></td>
</tr>
<tr>
<td>(\delta_3)</td>
<td>55 (*)</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>950</td>
<td>0</td>
<td>126.37</td>
<td></td>
</tr>
<tr>
<td>(\delta_4)</td>
<td>500</td>
<td>0.8</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>560 (*)</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>950</td>
<td>0</td>
<td>77.6</td>
<td></td>
</tr>
</tbody>
</table>

### Type-4 \((n = 58, p = 2000)\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time (in secs)</th>
<th>With Warm</th>
<th>Vanilla</th>
<th>Quality of Upper Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\delta_1)</td>
<td>300</td>
<td>16.66</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>367 (*)</td>
<td>0</td>
<td>216.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(\delta_2)</td>
<td>300 (*)</td>
<td>0</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td></td>
<td>370</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(\delta_3)</td>
<td>300</td>
<td>5</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>560 (*)</td>
<td>0</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>0</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>(\delta_4)</td>
<td>145 (*)</td>
<td>0</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>0</td>
<td>165</td>
<td></td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>0</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

### Table 1

Tables showing “Quality of Upper Bounds”, defined as \(100 \times (h_{alg} - \hat{h})/\hat{h}\), where \(h_{alg}\) refers to the objective value obtained by algorithm “alg” (at the given time), and \(\hat{h}\) is the best objective value found in the entire run-time duration of the algorithms. Two cases of “alg” \(\in \{ \text{“With Warm”, “Vanilla”} \}\) have been considered: “With Warm” denotes MILO warm-started with a solution from Algorithm 3, and “Vanilla” denotes a MILO solver without any warm-start specification. “With Warm” is found to obtain the best upper bound for a given computation time-limit in all the instances. In several instances, MILO is found to improve the solution obtained via Algorithm 3, after accepting it as a warm-start. For Type-1,2 the total time limit was 500 secs; for Type-3 it was 950 secs, and for Type-4 the algorithms were considered for a total time limit of 600 secs. For method “With Warm”, the times reported show the overall time taken by Algorithm 3 and the MILO algorithm. An asterisk “(*)” indicates that the best solution is obtained at that time. A “–” means that no feasible solution was obtained by the algorithm in that time.
we used Gurobi [25] version 6.0.3. For all experiments in Sections 6.1 and 6.2 (except the large scale examples) we used 16GB of memory.

6.1. Obtaining Good Quality Upper Bounds. From a practical viewpoint, being able to obtain good quality upper bounds to Problem (1.2) is, perhaps, of foremost importance. To demonstrate the effectiveness of our computational framework in this regard, we perform a series of experiments on the data-types described below.

**Type-Synth:** We generate a Gaussian ensemble, $X_{n \times p} \sim \text{MVN}(0, \Sigma)$, for the features, where $\sigma_{ij} = \rho^{|i-j|}$ for some value of $\rho \in [0, 1)$. The underlying true regression coefficient vector, $\beta^* \in \mathbb{R}^p$, has $\beta^*_j = 1$ for $k^*$ equi-spaced values of $j \in \{1, \ldots, p\}$ and $\beta^*_j = 0$ for the remaining values of $j$.

**Type-1:** This is of Type-Synth with $n = 100$, $p = 1000$, $\rho = 0$, $k^* = 10$. We studied Problem (1.2) for four different values, $\delta_{k^*}$, $k = 1, \ldots, 4$, of the parameter $\delta$, where $(\delta_1, \ldots, \delta_4) = \bar{\delta}(1, 1.5, 0.5, 0.2)$, and $\bar{\delta}$ is defined below.

**Type-2:** This is of Type-Synth with $n = 300$, $p = 1000$, $\rho = 0.8$, $k^* = 25$. Here, $(\delta_1, \ldots, \delta_4) = \bar{\delta}(1, 1.5, 0.5, 0.2)$.

**Type-3:** This is of Type-Synth with $n = 600$, $p = 2000$, $\rho = 0.8$, $k^* = 40$. Here, $(\delta_1, \ldots, \delta_4) = \bar{\delta}(1, 1.5, 0.5, 0.2)$.

**Type-4:** This is a semi-synthetic dataset: we considered the Radiation sensitivity gene expression dataset from Ch. 16 of the book [26]. The features were randomly downsampled to $p = 2000$ and there were $n = 58$ observations. We generated response $y$ based on a linear model with $\|\beta^*\|_0 = 10$, $\beta^*_j = 1$ for $j \leq 10$, and $\beta^*_j = 0$ for $j > 10$. Here, $(\delta_1, \ldots, \delta_4) = \bar{\delta}(1.2, 0.5, 0.2)$.

**Type-5:** This is of Type-Synth with $n = 100$, $p = 1000$, $\rho = 0$, $k^* = 10$; we considered one value of $\delta$, which was set to $\bar{\delta}$.

In each of the above examples, after $X$ was generated, we normalized its columns to have unit $\ell_2$-norm. Then, the response was generated as $y = X\beta^* + \epsilon$, where $\epsilon_i \sim \text{N}(0, \sigma^2)$, and $\sigma^2$ was adjusted to match the selected value of $\text{SNR} = \text{Var}(x^\top \beta^*)/\sigma^2$ (taken as 3 in all the above cases); the reference value of the tuning parameter was set to $\bar{\delta} = \|X^\top (y - X\beta^*)\|_\infty$.

We studied different first-order algorithms described in Section 4. Algorithm 3 was empirically seen to have the best performance over its constituents, Algorithms 1 and 2, when used separately. Hence, we used Algorithm 3 in all the experiments to obtain good upper bounds to Problem (1.2). The solution obtained from Algorithm 3 was passed as a warm-start to the MIL0 formulation (2.2) (for a large value of $M_U = 10^3$) — this hybrid MIL0 approach is denoted by “With Warm” in Table 1. We compared this method with the vanilla MIL0 formulation (2.2) (for a large value of $M_U = 10^3$), which was implemented without any warm-start information. Table 1 shows the objective values

---

obtained by these two methods — the MILo algorithm aided with advanced warm-starts was found to perform the best across all the examples. For the hybrid approach (“With Warm”), in many of the instances, the solution obtained by Algorithm 3 was further improved by MILo. In some cases, the vanilla MILo approach took a while before it was able to find a feasible solution. For example, in the Type-5 setting (which does not appear in Table 1) the best solution was delivered by Algorithm 3 within one minute; in contrast, the vanilla MILo algorithm failed to find a feasible solution within the total time limit of 1000 seconds.

Diabetes Dataset (n=442,p=64)

Fig 3: The evolution of MILo Optimality gaps (defined in Figure 2), as functions of time (in secs), for different values of the data fidelity parameter δ (leading to solutions with different number of nonzeros), presented for MILo methods with and without problem-specific information. Here, “With Intelligence” refers to a MILo algorithm aided with an advanced warm-start, in addition to the bounds specified in Section 5, and “Vanilla” refers to a MILo algorithm without any such additional information. MILo is found to benefit significantly from additional problem-specific information.

6.2. Computing Lower Bounds and Certificates of Global Optimality. Here we demonstrate how our framework delivers provably optimal solutions to Problem (1.2). In our first series of experiments we considered the popular diabetes dataset [19], which we examined with interaction terms included, giving us p = 64 and n = 442. All the features and the response were mean-centered and standardized to have unit ℓ2-norm. Figure 3 shows the performance of two versions of MILo: “With Intelligence” and “Vanilla”. “With Intelligence” refers to MILo formulation (5.1), where a MILo solver is provided with an advanced warm-start, say, \( \tilde{\beta}^0 \). The parameter specifications are obtained based on the method in Section 5.1.2; here, we used binary variables instead of the SOS-1 variables and used the box constraints and \( \ell_1 \)-constraint on \( \beta \). The “Vanilla” version of MILo was not provided with any such problem-specific information — we used formulation (2.2), as in Section 6.1. Our experimental results (Figure 3) show that “Intelligence” significantly enhances the performance of the MILo solver, in terms of proving global
6.2.1. Moderate Scale Examples. We considered some examples of Type-Synth for $\rho = 0$ and different values of $n, p, k^*$; in all the examples, we set $\delta = \|X^\top (Y - X\beta^*)\|_\infty$. The results for MIL0 with intelligence are displayed in Figure 4. In all the examples, the global solution $\|\hat{\beta}\|_0$ matched the generating model size $k^*$. We obtained an advanced warm-start $\hat{\beta}^0$ from a combination of Algorithm 3 and MIL0 formulation (2.5), where the latter was allowed to run for an overall time limit of 500 seconds. The warm start $\hat{\beta}^0$ was used to initialize formulation (5.1) — the parameter specifications in the formulation were obtained based on the method in Section 5.1.2. We also experimented with the version of formulation (5.1) that considers only box constraints on $\beta$; the results were often found to be roughly similar — both methods proved optimality, though there were sometimes differences in the total run-time (roughly around a few minutes). For all the synthetic examples presented in Figure 4, the vanilla version of MIL0 took much longer to prove optimality and hence they are not shown in Figure 4.

In all these instances, the $\ell_1$-Dantzig Selector resulted in a solution that was denser than the corresponding Discrete Dantzig Selector.

6.2.2. Large Scale Examples. We considered several large scale examples with $p$ ranging from 4500 to 10,000 — we note that these problem-sizes are orders of magnitude greater...
### Table 2

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k*</th>
<th>Upper Bound</th>
<th>Lower Bound</th>
<th>MILO Gap</th>
<th>Time (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,000</td>
<td>5,000</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>0</td>
<td>8.3</td>
</tr>
<tr>
<td>4,000</td>
<td>5,000</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>0</td>
<td>20.0</td>
</tr>
<tr>
<td>7,000</td>
<td>7,000</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>21.7</td>
</tr>
<tr>
<td>6,000</td>
<td>7,000</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>0</td>
<td>20.0</td>
</tr>
<tr>
<td>4,000</td>
<td>8,000</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>41.9</td>
</tr>
<tr>
<td>3,000</td>
<td>8,000</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>18.3</td>
</tr>
<tr>
<td>3,000</td>
<td>9,000</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>47.2</td>
</tr>
<tr>
<td>4,000</td>
<td>9,000</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>44.4</td>
</tr>
<tr>
<td>1,000</td>
<td>10,000</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>14.2</td>
</tr>
<tr>
<td>5,000</td>
<td>10,000</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>2.5</td>
</tr>
<tr>
<td>6,000</td>
<td>4,500</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>47.5</td>
</tr>
<tr>
<td>10,000</td>
<td>10,000</td>
<td>30</td>
<td>30</td>
<td>27</td>
<td>10%</td>
<td>42.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k*</th>
<th>Upper Bound</th>
<th>Lower Bound</th>
<th>MILO Gap</th>
<th>Time (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6,000</td>
<td>4,500</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>5.0</td>
</tr>
<tr>
<td>6,000</td>
<td>4,500</td>
<td>40</td>
<td>40</td>
<td>37</td>
<td>10%</td>
<td>12.5</td>
</tr>
</tbody>
</table>

Table showing times taken to reach global optimal solutions for several problem instances (both synthetic and real data), with \( p \) up to 10,000. For each instance, we list the upper bound, the lower bound, the corresponding MILO (Optimality) Gap and the time taken to reach the listed lower bound by the MILO solver equipped with “Intelligence” (as described in Figure 4). The times (in hours) refer to those taken by the MILO solver after being provided with a warm-start.

In all the instances, the best upper bound was obtained around approximately one hour; however, it took much longer to obtain a certificate of global optimality via the (almost) matching lower bounds and the MILO Gap. The MILO Gap was found to be zero in all the instances apart from the two where the algorithm was terminated upon obtaining a lower bound within 10% of the upper bound. The results demonstrate that provably optimal Discrete Dantzig Selector solutions can be obtained for large scale instances.

We studied a host of synthetic examples, all generated as in Section 6.2.1. We also considered a semi-synthetic dataset derived from the famous Gisette data [http://archive.ics.uci.edu/ml/datasets/Gisette](http://archive.ics.uci.edu/ml/datasets/Gisette). Here, we generated a response \( y \), based on the Gisette data covariates (each feature was standardized to have zero mean and unit \( \ell_2 \) norm) — here, \( n = 6000 \) and \( p = 4500 \), we set \( \beta_i^* = 1, i = 1, \ldots, k^* \) and the remaining \( \beta_i^* = 0 \); SNR=3 and considered two instances with \( k^* = 20, 30 \).

The algorithmic set-up was similar to that used in Section 6.2.1. The results are presented in Table 2. In all these instances, the \( \ell_1 \)-Dantzig Selector resulted in a solution that was more dense than those obtained via the Discrete Dantzig Selector. We note that the MILO solver delivered solutions that matched the optimal solution of the data generating mechanism. Typically, the time taken to prove optimality marginally increases with larger values of \( k^* \) (for a fixed \( n, p \)); for a fixed \( k^* \), \( p \) the times taken to...
certify optimality increases with decreasing values of $n$. The examples demonstrated in this paper, show the largest instances of discrete optimization problems for exact variable selection, that can be solved to provable optimality.

7. Numerical Experiments: Statistical Properties. We conducted a series of synthetic data experiments to understand the statistical properties of the Discrete Dantzig Selector and compare them to those of the $\ell_1$-Dantzig Selector and related variants.

We used six different datasets in our analysis.

**Example-A:** This is of Type-Synth with $n = 200$, $p = 500$, $\rho = 0$ with $k^* = 20$.

**Example-B:** This dataset was similar to the one taken in Example-A, but the amplitudes and signs of the true regression coefficients were allowed to vary: the twenty nonzero $\beta_j^*$’s were equally spaced in the interval $[-10, 10]$, with $\min_j: \beta_j^* \neq 0 |\beta_j^*| \approx 0.53$.

**Example-C:** This is of Type-Synth with $n = 100$, $p = 500$, $\rho = 0.85$ with $k^* = 10$.

**Example-D:** We set $n = 100, p = 300$ and let $X \sim \text{MVN}(0, \Sigma)$, where $\sigma_{12} = \sigma_{21} = 0.7$, and all the remaining $\sigma_{jk}$ are equal to zero. We also took $\beta_1^* = 1$ and $\beta_2^* = -1$, with the remaining coefficients $\beta_j^*$ set to zero, resulting in $\|\beta^*\|_0 = 2$. (This example is a larger version of Example $1'$ described in Section 1 and illustrated in Figure 1.)

**Example-E:** This is similar to Example-B, but with a larger problem size: $n = 500, p = 2000$ and $\|\beta^*\|_0 = 25$.

**Example-F:** This is of Type-Synth with $n = 500, p = 2000, \rho = 0.7$ and $k^* = 50$.

In each of the above cases, after $X$ was generated, we normalized its columns to have unit $\ell_2$-norm. Then, the response was generated as $y = X\beta^* + \epsilon$, where $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$, and $\sigma^2$ was adjusted to match the selected value of SNR, which was varied across $\{3, 7, 10\}$ in the examples.

We considered the following estimators in our analysis:

- “Warm” — this method applies a heuristic strategy to obtain upper bounds to Problem (1.2). We used\(^{11}\) Algorithm 2, described in Section 4.1.2.
- “L0-DS” — the solution obtained from “Warm” is taken as a warm-start to a MILP solver and subsequently allowed to run with a time limit of 4000 seconds.
- “L0-DS-Pol” — this is a “polished” version of the Discrete Dantzig Selector estimator “L0-DS” and is obtained by performing a simple least squares fit on the support of the “L0-DS” estimate.
- “L1-DS” — this is the original $\ell_1$-Dantzig Selector.

\(^{11}\)This is similar to a re-weighted $\ell_1$-minimization [15] method applied to Problem (1.2). We took the penalty $\rho_\gamma(|\beta|) \propto \log(\frac{|\beta|}{\gamma} + 1)$ on a geometrically decreasing grid of ten $\gamma$ values: $\gamma_i = 10^{-2} \times 0.8^{i-1}$ for $i = 1, \ldots, 10$.
• “L1-DS-Pol” — this is a polished version of the “L1-DS”.

Each of the above estimators were computed on a range of approximately thirty different \( \delta \) values around \( \bar{\delta} = \|X^T(y - X^T \beta^*)\|_\infty \). We considered ten different replications (based on different \( \epsilon \) realizations) and averaged the results. The optimal tuning parameter \( \delta^{opt} \) for every model was selected based on the value of \( \delta \) that minimized the prediction accuracy with respect to the true regression coefficients. For this chosen value of \( \delta^{opt} \) we considered different metrics to assess the performance of the different estimators. We computed the squared \( \ell_2 \)-error in estimating the regression coefficients:
\[
\|\hat{\beta} - \beta^*\|_2^2.
\]
We also considered the “Variable Selection error”, which is defined as
\[
\sum_{j=1}^p I(\hat{S}_j \neq S^*_j),
\]
where \( \hat{S}_j \) is the \( j \)th coordinate of \( \hat{S} := \text{Supp}(\hat{\beta}) \), and \( S^*_j \) is the \( j \)th element of \( S^* = \text{Supp}(\beta^*) \). Finally, we computed “Prediction Accuracy”, defined as
\[
\frac{\|X\hat{\beta} - X\beta^*\|_2^2}{\|X\beta^*\|_2^2},
\]
and “number of nonzeros”, which refers to the number of nonzero coefficients in \( \hat{\beta} \). A collection of representative results with SNR=10 is displayed in Figure 5. A larger display of additional examples with varying SNR values is presented in Table 4, which is given in Section D of the Appendix. In Table 5, given in the same section, we provide comparisons with the polished version of the \( \ell_1 \)-Dantzig Selector.

Our experiments show that polishing the \( \ell_1 \)-Dantzig Selector may lead to marginally better solutions relative to the original Dantzig Selector, but the corresponding statistical performance is inferior to that of the estimates based on the Discrete Dantzig Selector estimator.

We note that the performance of the Lasso was found to be quite similar to that of the \( \ell_1 \)-Dantzig Selector. The statistical performance of the subset selection procedure (1.4), as described in [7], was found to be similar to that of the Discrete Dantzig Selector for \( p \leq 1000 \). Because the main focus of the paper is to show that Discrete Dantzig Selector is a computationally tractable procedure, which delivers estimates with better statistical properties than its \( \ell_1 \) counterpart, we restrict our numerical studies to the methods listed above.

Summary of Findings. Based on the experimental results, we observe that the Discrete Dantzig Selector and its polished variant perform quite well, when compared to the competing methods in terms of estimating \( \beta^* \); they also demonstrate superior variable selection properties — the \( \ell_0 \) methods obtain the sparsest models across all the examples. “Warm” does not perform very well when compared to “L0-DS”, even though both methods attempt to solve Problem (1.2) — this shows estimators based on rigorous optimization procedures have better statistical properties. We observe that “L0-DS” and “L0-DS-Pol” possess similar variable selection properties, however, the latter may lead to better estimators of \( \beta^* \) and \( X\beta^* \), due to the least squares post-processing. Polishing of the \( \ell_1 \)-Dantzig Selector may not lead to better solutions, due to the weak variable selection properties of “L1-DS”. In some cases, when the value of \( \rho \) is quite large and, consequently, the covariates are highly correlated (see Example-C, Example-D and Example-E), the basic problem of variable selection becomes difficult: instead of choosing a “signal” variable, the “L0-DS” chooses its correlated surrogate. In these cases, as expected, we observe that the “L0-DS” incurs relatively large variable selection error — the prediction accuracy of these models however, demonstrate a more
THE DISCRETE DANTZIG SELECTOR

Example-A (n=200, p=500) Example-B (n=200, p=500) Example-C (n=100, p=500) Example-D (n=100, p=300)

Fig 5: The statistical performance of the Discrete Dantzig Selector (L0-DS); its polished version, which does a least squares refitting on the support (L0-DS-Pol); the heuristic estimates delivered by Algorithm 2 (Warm); and the original $\ell_1$-Dantzig Selector (L1-DS). We display three different metrics: [top panel] squared $\ell_2$-error in estimating $\beta$, [middle panel] the 0-1 variable selection error and [bottom panel] the number of nonzeros in the optimally selected model; the horizontal dotted line shows the number of nonzeros in the “true” model. We observe that the Discrete Dantzig Selector based approaches perform very well in terms of obtaining a model with high quality estimation and variable selection properties. Their models are substantially sparser than those for the $\ell_1$-based methods. The heuristic approach, “Warm” which approximately optimizes Problem (1.2), falls short in obtaining high quality statistical estimates. A number of additional experiments (with results similar to this figure), are presented in Section D of the Appendix.
In all the above instances, data was generated as per Type-Synth, with SNR=10. Both the methods Discrete Dantzig Selector and MIPGO (with the MCP penalty) were run on twenty different values of the tuning parameter, and the best solutions are reported. L0-DS-Pol refers to the least squares solution obtained on the variables selected by Discrete Dantzig Selector. For every value of the tuning parameter, each method was run for a time budget of $t_1, t_2$ secs with $t_1 < t_2$ as specified in the “Time” column. The Discrete Dantzig Selector method reaches the best solution within $t_1$ seconds, much earlier than its competitor. The MIPGO method is seen to take orders of magnitude time longer to get a solution of the same quality as the Discrete Dantzig Selector— the differences become more pronounced with increasing problem size.

<table>
<thead>
<tr>
<th>Example</th>
<th>Metric</th>
<th>Time (secs)</th>
<th>L0-DS-Pol</th>
<th>MIPGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho = 0$</td>
<td>Variable Selection</td>
<td>60</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>$n = 100$</td>
<td>Error</td>
<td>200</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>$p = 200$</td>
<td>$|\hat{\beta} - \beta^*|^2_2$</td>
<td>60</td>
<td>0.309</td>
<td>20.00</td>
</tr>
<tr>
<td>$k^* = 20$</td>
<td></td>
<td>200</td>
<td>0.309</td>
<td>8.79</td>
</tr>
<tr>
<td>$\rho = 0.7$</td>
<td>Variable Selection</td>
<td>60</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>$n = 100$</td>
<td>Error</td>
<td>200</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>$p = 200$</td>
<td>$|\hat{\beta} - \beta^*|^2_2$</td>
<td>60</td>
<td>0.063</td>
<td>0.27</td>
</tr>
<tr>
<td>$k^* = 10$</td>
<td></td>
<td>200</td>
<td>0.063</td>
<td>0.076</td>
</tr>
<tr>
<td>$\rho = 0.7$</td>
<td>Variable Selection</td>
<td>100</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>$n = 300$</td>
<td>Error</td>
<td>500</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>$p = 600$</td>
<td>$|\hat{\beta} - \beta^*|^2_2$</td>
<td>100</td>
<td>0.094</td>
<td>9.14</td>
</tr>
<tr>
<td>$k^* = 20$</td>
<td></td>
<td>500</td>
<td>0.094</td>
<td>1.548</td>
</tr>
<tr>
<td>$\rho = 0.7$</td>
<td>Variable Selection</td>
<td>150</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>$n = 300$</td>
<td>Error</td>
<td>1000</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>$p = 1000$</td>
<td>$|\hat{\beta} - \beta^*|^2_2$</td>
<td>150</td>
<td>0.175</td>
<td>25.00</td>
</tr>
<tr>
<td>$k^* = 25$</td>
<td></td>
<td>1000</td>
<td>0.175</td>
<td>14.893</td>
</tr>
</tbody>
</table>

Table 3

7.1. Comparisons with MIPGO [34]. MIPGO is a discrete optimization framework for minimizing a regularized version of the least squares loss, with a non-convex quadratic penalty (for example, SCAD or MCP). This corresponds to a non-convex quadratic optimization problem, which the authors express as a discrete linear optimization problem via linear complementary constraints [24]. This representation results in many more binary variables (several multiples of $p$) than required for the Discrete Dantzig Selector. For example, in the case of the MCP penalty, the approach in [34] presents a MILP with $4p$ binary variables and many more continuous variables. In particular, with $n = 300$ and $p = 1000$, the MIPGO solver creates a problem with 38,000 variables and 27,000 equality constraints, which after presolve reduces to a problem with approximately 7,000 continuous and 4,000 binary variables. These optimization problems are substantially larger than (2.2), which is a MILP with $p$ continuous and as many binary variables. As a result, the MIPGO formulation becomes computationally expensive as the dimensionality of the problem increases. This is illustrated in Table 3, where we show on several synthetic instances that, with a particular time budget, our proposed approach, the Discrete Dantzig Selector formulation (2.2) equipped with a warm-start from Algorithm 2, leads to better solutions than MIPGO. The quality of solutions produced by MIPGO is found to improve with more computation time, but the time taken

---

12We used the code of [34], obtained from the first author’s website. The numbers are read off from the Gurobi log report.
can be substantially larger than that of the Discrete Dantzig Selector. The synthetic datasets for Table 3 were generated the same way as those in Section 6, and we set the concavity parameter for the MCP penalty in the MIPGO code to its default value of \( a = 2 \).

**Acknowledgements.** The authors would like to thank Emmanuel Candes for helpful comments and encouragement. R.M. will like to thank graduate student Jonathan Goetz for research assistance during the early phase of this project and Juan-Pablo Vielma for helpful discussions. R.M.’s research was partially supported by ONR N000141512342 and a grant from the Moore Sloan Foundation. Peter Radchenko’s research was partially supported by NSF Grant DMS-1209057.

**References.**

34


APPENDIX

APPENDIX A: MILO PRELIMINARIES

As already alluded to in Section 1, there has been significant progress in the theory and practice of MILO over the past fifty years. Specifically, the computational power of MILO solvers has undergone impressive advances over the past twenty-five years — the cumulative machine-independent speedup factor in MILO solvers between 1991 and 2015 is estimated to be 780,000 [10]. This progress can be attributed to the inclusion of both theoretical and practical advances into MILO solvers. Some of the main factors responsible for this speedup are advances in cutting plane theory, improved heuristic methods, disjunctive programming for branching rules, techniques for preprocessing MILO’s, using linear optimization as a black box to be called by MILO solvers, and improved linear optimization methods [10]. In addition, there have been substantial improvements in hardware speed: the overall hardware speedup from 1993 to 2015 is approximately estimated to be $10^{5.75} \sim 570,000$ [sup]. When both hardware and software advances are combined, the overall speedup for MILO problems is estimated to be around 450 billion!

One attractive feature of MILO solvers, which is a stark contrast to heuristic approaches, is that the former provide (a) feasible solutions, which are also upper bounds to the minimum objective value and (b) lower bounds for the optimal value of the objective function. As a MILO solver makes its way to the global optimum, the lower bounds become tighter, thereby providing improved certificates of sub-optimality (see Figure 2 for an illustration). This aspect of MILO solvers is quite useful, especially if one decides to stop the solver before reaching the global optimum. In the modern day world, MILO plays a key role in various impactful application areas of operations research: revenue management, air-traffic control, scheduling and matching tasks, production planning and others [45, 6, 17]. In this paper, we show how the power of MILO can be used in the context of a problem of fundamental importance in statistics, namely, sparse linear model estimation — we build upon recent line of work in computational statistics, at the interface of modern discrete optimization and fundamental techniques in statistical modeling [4, 7].

APPENDIX B: PROOFS FOR SECTION 3

Proof of Theorem 3.1. Let $\tilde{\epsilon}$ denote the projection of $y$ onto the orthogonal complement to the space spanned by the predictor vectors $x_j, j = 1, \ldots, p$. Note that

$$\|y - X\beta\|^2 = \sum_{j=1}^{p} (c_j - \beta_j)^2 + \|\tilde{\epsilon}\|^2.$$

Thus, under the constraint $\|\beta\|_0 \leq k$, the smallest sum of squares is achieved by setting $\beta_j = c_j$ for $j = 1, \ldots, k$ and $\beta_j = 0$ for $j = k + 1, \ldots, p$. This completes the proof of part 1.

Note that

$$|x_j^\top (y - X\beta)| = |c_j - \beta_j|.$$
Thus, the constraint \( \|X^T(y - X\beta)\|_\infty \leq \delta \) is satisfied if and only if \( \beta_j \in [c_j - \delta, c_j + \delta] \) for \( j = 1, \ldots, p \). In order to minimize \( \|\beta\|_0 \), coefficients \( \beta_j \) for which \( 0 \in [c_j - \delta, c_j + \delta] \) are set to zero. Thus, \( \beta_j = 0 \) if and only if \( |c_j| \leq \delta \), which implies \( \beta_j = 0 \) for \( j > k \). This completes the proof of part 2.

**Proof of Theorem 3.2.** Throughout this proof we omit the words “with probability tending to one” to improve the presentation. The constraint \( \|X^T(y - X\beta)\|_\infty \leq \delta \) implies \( \|X^T X(\beta^* - \beta) + X^T \epsilon\|_\infty \leq \delta \). Recall that \( \delta = o(n^{1/2}) \) and note that \( \|X^T \epsilon\|_\infty = O_p(1) \), due to the scaling of the predictors and the assumptions on the \( \epsilon \). Consequently, \( \|X^T X(\beta^* - \beta)\|_2 = o_p(n^{1/2}) \). Because \( X^T X \) converges to an invertible matrix \( C \), we conclude that there exists a \( o_p(1) \) sequence of random variables \( b_n \), such that the bound

\[
\|n^{-1/2} \hat{\beta} - n^{-1/2} \beta^*\|_2 \leq b_n
\]

simultaneously holds for all the Discrete Dantzig solutions \( \hat{\beta} \). Recall that \( \beta^* = n^{1/2} \hat{\beta}^* \), for some fixed vector \( \hat{\beta}^* \). Consequently, \( \beta^*_j \neq 0 \) implies \( \hat{\beta}_j \neq 0 \) for \( j = 1, \ldots, p \). In other words the support of each Discrete Dantzig solution contains \( J^* \). It is only left to show that the cardinality of each such support cannot be greater than \( |J^*| \). Note that with probability tending to one, \( \beta^* \) is feasible for the Discrete Dantzig optimization problem. Indeed,

\[
\|X^T(y - X\beta^*)\|_\infty = \|X^T \epsilon\|_\infty = O_p(1),
\]

which is bounded above by \( \delta \), due to the assumption \( \delta \to \infty \). Thus, inequality \( \|\hat{\beta}\|_0 \leq |J^*| \) holds for each Dantzig Selector solution, which completes the proof of part 1.

In the paragraph above we deduced that the minimum value of the Discrete Dantzig objective function equals \( |J^*| \). We also showed that \( \beta^* \) is feasible for the Discrete Dantzig optimization problem. A similar argument establishes the feasibility of \( \beta^O \). Consequently, \( \beta^* \) and \( \beta^O \) are indeed Discrete Dantzig solutions, which completes the proof of part 2.

**Proof of Proposition 3.1.** Consider an arbitrary nonzero \( \theta \in \mathbb{R}^p \), such that \( \|\theta\|_0 \leq 2k \). Let \( J_0 \) be the index set of the \( k \) largest, in magnitude, coordinates of \( \theta \). Observe that \( \|\theta_{J_0}\|_1 \leq \|\theta_{J_0}\|_1 \) and \( \|\theta\|_2 = \|\theta_{J_0}\|_2 \), because \( m \geq k \). Thus

\[
\frac{\|X\theta\|_2}{\|\theta\|_2} = \frac{\|X\theta\|_2}{\|\theta_{J_0}\|_2} \geq \kappa(k, c_0, m),
\]

for \( c_0 \geq 1 \), which implies \( \gamma(2k) \geq \kappa(k, c_0, m) \). Also note that

\[
2 \frac{\|X\theta\|_2^2}{\|\theta\|_2^2} \geq \frac{\|X\theta\|_2^2}{\|\theta_{J_0}\|_2^2} \geq \left[ \kappa(k, c_0) \right]^2,
\]

for \( c_0 \geq 1 \), which gives \( \gamma(2k) \geq \kappa(k, c_0)/\sqrt{2} \).

**Proof of Theorem 3.3.** Note that \( X^T \epsilon \) is a mean zero Gaussian vector, such that the variance of each component is \( \sigma^2 \). Consequently, it follows from well-known maximal
inequalities for Gaussian variables that bound $|\mathbf{X}^\top \mathbf{e}|_\infty \leq \delta$ holds with probability at least $1 - (p^s \sqrt{\pi \log p})^{-1}$. The rest of the proof is conducted on the set where the above bound is valid. Note that on this set $\beta^*$ is a feasible solution for the optimization problem (1.2), which implies $\|\beta\|_0 \leq \|\beta^*\|_0$. Recall that we denote $\|\beta^*\|_0$ by $s^*$ and derive the following inequalities:

$$
\gamma(2s^*)^2 \|\beta - \beta^*\|_2^2 \leq \|\mathbf{X}(\beta - \beta^*)\|_2^2 = (\beta - \beta^*)^\top \mathbf{X}^\top \mathbf{X}(\beta - \beta^*) \\
\leq \|\mathbf{X}^\top \mathbf{X}(\beta - \beta^*)\|_\infty \|\beta - \beta^*\|_1 \\
\leq \left( \|\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\beta)\|_\infty + n^{-1} \|\mathbf{X}^\top \mathbf{e}\|_\infty \right) \|\beta - \beta^*\|_1.
$$

Because both $\|\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\hat{\beta})\|_\infty$ and $\|\mathbf{X}^\top \mathbf{e}\|_\infty$ are bounded above by $\delta$, we derive

$$
\|\beta - \beta^*\|_2^2 \leq (2s^*)^{-1} 2\delta \|\beta - \beta^*\|_1.
$$

Applying inequality $\|\beta - \beta^*\|_1 \leq (2s^*)^{1/2} \|\beta - \beta^*\|_2$ to either the left or the right hand side of the above display yields the $\ell_1$ and the $\ell_2$ estimation bounds, respectively, in the statement of Theorem 3.3.

Finally, to establish the prediction error bound, observe the following inequality:

$$
\|\mathbf{X}(\hat{\beta} - \beta^*)\|_2^2 \leq 2\delta \|\beta - \beta^*\|_1,
$$

which is a direct consequence of the two displays given above. We then complete the proof by combining the above display with the inequalities

$$
\|\beta - \beta^*\|_1 \leq (2s^*)^{1/2} \|\beta - \beta^*\|_2 \leq (2s^*)^{1/2} \gamma(2s^*)^{-1} \|\mathbf{X}(\hat{\beta} - \beta^*)\|_2.
$$

Proof of Theorem 3.4. We again focus on the set of high probability, where inequality $\|\mathbf{X}^\top \mathbf{e}\|_\infty \leq \delta$ holds. Because $\beta^*$ is a feasible solution to the optimization problem (1.2), we have $\|\beta^*\|_0 \geq \hat{s}_{LB}$. Thus, $\|\beta\|_0 \leq (1 + \psi)\|\beta^*\|_0$. The rest of the proof is identical to the one for Theorem 3.3, with one exception: the bound $\|\beta\|_0 \leq \|\beta^*\|_0$ contains an additional factor $(1 + \psi)$.

APPENDIX C: ADDITIONAL ALGORITHM DETAILS AND PROOFS

C.1. Details on Algorithm 1. Step (4.3) can be performed via a hard thresholding operation [18], as it is of the form:

$$
\hat{\beta}(\lambda') := \arg \min_\beta \|\beta - \mathbf{c}\|_2^2 + \lambda' \|\beta\|_0,
$$

for an appropriately chosen $\lambda' = \frac{2}{\lambda}$ and $\mathbf{c} = \mathbf{a} + \frac{1}{\lambda} \mathbf{v}$; a solution is given by $\hat{\beta}_j(\lambda') = c_j \mathbf{1}(|c_j| > \sqrt{\lambda'})$, $j = 1, \ldots, p$. The update step (4.4) involves the following projection:

$$
\min_\alpha f(\alpha) := \|\alpha - \mathbf{c}\|_2^2 \text{ subject to } \|\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\alpha)\|_\infty \leq \delta,
$$
where, \( \bar{c} = \beta - \nu/\lambda \). While the projection \( (C.2) \) can be computed by using standard quadratic programming methods, in our experience, we found them\(^{13}\) to be quite time consuming for larger problems \( (p \geq 1000) \), especially because this projection needs to be computed for every iteration (indexed by \( k \)) of \( (4.3)-(4.5) \). Thus, we recommend using specialized first-order methods — these methods also naturally make use of warm-start information, which is particularly useful to us due to the iterative nature of the updates \( (4.3)-(4.5) \). Unless \( X \) has uncorrelated columns, it is not straightforward to solve \( (C.2) \) in its primal form — we thus consider a dual of Problem \( (C.2) \), for which we apply first-order methods for convex composite minimization \([37]\). To improve the flow of presentation, we relegate the description of a more general first-order method, which also applies to Problem \( (C.2) \), to Section C.3 in the Appendix. We repeat steps \( (4.3)-(4.5) \) until an (approximate) convergence criterion is met — see for example \([35, 12]\) for convergence results for the general method. We terminate the algorithm as soon as the successive changes in the \( \beta \) updates become small, \( \|\beta_{k+1} - \beta_k\|_2 \leq \tau_1 \|\beta_k\|_2 \), and one has approximate primal feasibility, \( \|\alpha_k - \beta_k\|_2 \leq \max\{\|\beta_k\|_2, \|\alpha_k\|_2\} \), where, \( \tau_1, \tau_2 \) are tolerance parameters, typically taken as \( 10^{-4} \).

### C.2. Additional Details on Algorithm 2: Solving Problem \( (4.9) \)

Observe that Problem \( (4.9) \) is of the composite form \([37]\):

\[
\text{(C.3)} \quad \min_{\theta} f_1(\theta) + f_2(\theta) \quad \text{subject to} \quad \theta \in C,
\]

where the function \( f_1(\theta) \) is smooth, with its gradient Lipschitz continuous: \( \|\nabla f_1(\theta) - \nabla f_1(\theta')\| \leq L \|\theta - \theta'\|; f_2(\theta) \) is nonsmooth and \( C \) is a convex set. In our specific case, the smooth component is the zero function, \( f_2(\theta) = \sum_{i=1}^{p} w_i |\theta_i| \) and \( C = \{ \theta : \|X^\top(y - X\theta)\|_\infty \leq \delta \} \). Thus, one may appeal to first-order optimization methods \([1, 38, 37]\) for composite function minimization. This requires solving, at every iteration, a problem of the form:

\[
\text{(C.4)} \quad \theta^{m+1} = \arg \min_{\theta} \frac{L}{2} \|\theta - \overline{\theta}^m\|_2^2 + \sum_{i=1}^{p} w_i |\theta_i| \quad \text{subject to} \quad \|X^\top(y - X\theta)\|_\infty \leq \delta,
\]

for some choice of \( L > 0 \) and \( \overline{\theta}^m \), and \( w_i = |\rho'_i(\vert\beta^k_i\vert)| \). If \( \overline{\theta}^m = \theta^m \), then the above update sequence becomes identical to proximal gradient descent \([1]\). One may also use accelerated gradient descent methods, where \( \overline{\theta}^m = \theta^m + \left(\frac{t_{m-1}}{t_{m+1}}\right)(\theta^m - \theta^{m-1}) \), with \( \overline{\theta}^1 = \theta^1 \) and \( t_1 = 1 \). We describe in Section C.3 first-order gradient methods that can be used to compute solutions to Problem \( (C.4) \). The sequence \( \theta^m \), defined via \( (C.4) \), leads to the solution of Problem \( (4.9) \) as \( m \to \infty \), providing a \( O(\frac{1}{m}) \)-suboptimal solution in \( m \) many iterations if one uses standard proximal gradient descent methods; the convergence rate can be improved to \( O(\frac{1}{m^2}) \) if one uses the accelerated gradient descent version of the algorithm.

Instead of choosing \( f_1(\theta) \equiv 0 \) one may also choose \( f_1(\theta) = \frac{L}{2} \|\theta\|_2^2 \), for a small value of \( \tau > 0 \). Interestingly, for small values of \( \tau \) the minimizer to Problem \( (C.3) \) is also a

\(^{13}\)Our reference is Gurobi’s quadratic programming solver.
minimizer of the problem with the choice \( f_1(\theta) = 0 \). This equivalence of solutions which holds true in much more generality is often known as exact regularization of convex programs in the mathematical programming literature — see for example [22]. Even if the two problems are not equivalent, the choice of \( f_1(\theta) = \frac{\tau}{2} \|\theta\|_2^2 \) always serves as an approximate solution to Problem (4.9). With this choice of \( f_1(\theta) \), one needs to solve a problem of the form:

\[
\min_\theta \frac{\tau}{2} \|\theta\|_2^2 + \sum_{i=1}^p w_i|\theta_i| \quad \text{subject to} \quad \|X^T(y - X\theta)\|_\infty \leq \delta.
\]

A solution to the above problem can be computed by considering its dual and applying (accelerated) proximal gradient methods on the dual formulation, as described in Section C.3. In this approach, a two-stage iterative algorithm of the form (C.4) described above, is not required.

Algorithm 2 suggests that we solve Problem (4.9) repeatedly for different values of \( \gamma \) — it turns out that the overall cost for solving all these problems is quite small. This is because (a) the problems do not change much across different values of \( \gamma \); and (b) for a fixed \( \gamma \), while moving across different values of \( k \), the linear optimization problems are quite similar since the weights \( |\rho^\prime_{\gamma}(|\beta_k^i|)| \) do not change much across \( k \).

Thus the solutions obtained from one linear optimization problem can be used as a warm-start to solve the next linear optimization problem. This is found to reduce the overall computation time. Both the first-order methods (described above) and simplex methods can gracefully take advantage of warm-starts.

### C.3. Dual Gradient Method.

Here we describe dual proximal gradient algorithms that optimize the following convex problem:

\[
\min_{\alpha} \frac{1}{2}\|\alpha - \bar{c}\|_2^2 + \sum_{i=1}^p w_i|\alpha_i| \\
\text{subject to} \quad \|A\alpha - b\|_\infty \leq \delta,
\]

where we assume that \( w_i \geq 0 \) and the set \( \{\alpha : \|A\alpha - b\|_\infty \leq \delta\} \) is nonempty.

The above problem can be written equivalently as:

\[
\min_{\alpha, \zeta} \frac{1}{2}\|\alpha - \bar{c}\|_2^2 + \sum_{i=1}^p w_i|\alpha_i| \\
\text{subject to} \quad \|\zeta\|_\infty \leq \delta, \quad \zeta = A\alpha - b.
\]

The minimum of the above problem can be obtained by maximizing a dual problem, obtained by dualizing the equality constraints \( \zeta = A\alpha - b \); this consequently leads to the following (ignoring irrelevant constants that do not depend upon the optimization
variable $\mu$) problem:

\[
g(\mu) := \min_{\zeta, \alpha} \left( \frac{1}{2} \| \alpha - \bar{c} \|_2^2 + \langle \mu, \zeta - (A\alpha - b) \rangle + \sum_{i=1}^{p} w_i |\alpha_i| \right)
\]

subject to \[\| \zeta \|_{\infty} \leq \delta.\]

The above can be simplified to:

\[
g(\mu) = \min_{\alpha} \left( \frac{1}{2} \| \alpha - \bar{c} \|_2^2 - \| \mu \|_1 \delta - \langle \mu, (A\alpha - b) \rangle + \sum_{i=1}^{p} w_i |\alpha_i| \right)
\]

where

\[
g_1(\mu) = \min_{\alpha} \left( \frac{1}{2} \| \alpha - \bar{c} \|_2^2 - \langle \mu, (A\alpha - b) \rangle + \sum_{i=1}^{p} w_i |\alpha_i| \right)
\]

Note that $\hat{\alpha}$, the unique minimizer of the above problem, is given by:

\[
\hat{\alpha} = \arg \min_{\alpha} \left( \frac{1}{2} \| \alpha - \bar{c} \|_2^2 - \langle \mu, (A\alpha - b) \rangle + \sum_{i=1}^{p} w_i |\alpha_i| \right),
\]

i.e. $\hat{\alpha}_i = \text{sgn}(\bar{c}_i + a_i^T \mu) \cdot \max \left\{ |\bar{c}_i + a_i^T \mu| - w_i, 0 \right\}, i = 1, \ldots, p,$

where $a_i$ is the $i$th column of $A$. It follows from standard convex analysis [41] that the function $\mu \mapsto g_1(\mu)$ is differentiable with its gradient given by:

\[
\nabla g_1(\mu) = -(A\hat{\alpha} - b),
\]

and its gradient is Lipschitz continuous:

\[
\| \nabla g_1(\mu) - \nabla g_1(\mu') \| \leq \| A \|_2 \| \mu - \mu' \|,
\]

where $\| A \|_2$ denotes the largest singular value of $A$ and $\| \cdot \|$ for a vector denotes the usual $\ell_2$-norm.

By using standard quadratic programming duality theory [11], the minimum of Problem (C.5) can be obtained by equivalently maximizing the unconstrained dual problem $g(\mu)$ in the dual variable $\mu$, which is equivalent to the following minimization problem:

\[
\min_{\mu} -g(\mu) = \min_{\mu} (-g_1(\mu) + \delta \| \mu \|_1).
\]

This problem is of the composite form [37], and proximal gradient descent methods [39, 37, 40] apply to it directly.

For the special case of Problem (C.2), the method described above applies with $w_i = 0, i = 1, \ldots, p$, for which

\[
g(\mu) = -\frac{1}{2} \| A^T \mu \|_2^2 + \langle b, \mu \rangle - \| \mu \|_1 \delta,
\]
with $A = X^\top X$ and $b = X^\top y$. Clearly, (C.12) is an $\ell_1$-regularized quadratic program, with the primal dual relationship being: $\alpha = \tau + A^\top \mu$.

Note that Problem (C.5) needs to be solved several times during the course of Algorithm 1 and Algorithm 2, across the different iterations. Fortunately, these problems are not completely unrelated, in fact, they are quite “similar”. In Algorithm 2 the weights $w_i$ change; and in Algorithm 1, the parameter $c$ changes. Since the problems are similar, it is not unreasonable to expect that the optimal dual variables corresponding to these two problems do not change much. Thus, it is useful to initialize the dual variable $\mu$ for one instantiation of Problem (C.5) with the (dual) solution obtained from another instantiation of Problem (C.5). This simple strategy leads to substantial performance gains over solving the problems sequentially.


**Proof.** Note that the sequence $\beta^k$, defined via (4.9) satisfies the following relationship:

(C.13) $\ h(\beta^k) = \bar{h}(\beta^k; \beta^k) \geq \min_{\beta} \bar{h}(\beta; \beta^k) = \bar{h}(\beta^{k+1}; \beta^k).$

subject to $\|X^\top (y - X\beta)\|_\infty \leq \delta$

Observing that $\bar{h}(\beta^{k+1}; \beta^k) \geq h(\beta^k)$, we have:

(C.14) $\ h(\beta^k) = \bar{h}(\beta^k; \beta^k) \geq \bar{h}(\beta^{k+1}; \beta^k) \geq h(\beta^{k+1}),$

and, thus, the sequence $h(\beta^k)$ is decreasing. Subtracting $h(\beta^k)$ from all sides of the above inequality, we derive:

$$0 \geq \bar{h}(\beta^{k+1}; \beta^k) - h(\beta^k) \geq h(\beta^{k+1}) - h(\beta^k).$$

The first part of the above display gives us, using (4.8):

$$0 \geq \bar{h}(\beta^{k+1}; \beta^k) - h(\beta^k) = \sum_{i=1}^{p} \left< \rho'_\gamma(|\beta^k_i|), |\beta^{k+1}_i| - |\beta^k_i| \right> = \Delta(\beta^k),$$

which means $\Delta(\beta^k) \leq 0$ for all $k$. If $\Delta(\beta^k) < 0$, then $\beta^{k+1}$ leads to a strictly improved value of the objective function. If $\Delta(\beta^k) = 0$, then $\beta^k$ is a fixed point of the above update equation. Hence, $\Delta(\beta^k)$ is a measure of how far $\beta^k$ is from a first-order stationary point of Problem (4.7).

The display in (C.14) shows that the objective values are decreasing, and, because the objective values are all bounded below (by zero), the decreasing sequence converges.

In addition, we have that

$$h(\beta^k) - h(\beta^{k+1}) \geq -\Delta(\beta^k).$$
Adding the above for $k = 1, \ldots, K$, we have:

$$h(\beta^1) - h(\beta^{K+1}) \geq \sum_{K \geq k \geq 1} \left\{ -\Delta(\beta^k) \right\} \geq K \min_{1 \leq k \leq K} \left\{ -\Delta(\beta^k) \right\},$$

which leads to the following convergence rate:

\begin{equation}
\min_{1 \leq k \leq K} \left\{ -\Delta(\beta^k) \right\} \leq \frac{1}{K} \left( h(\beta^1) - h(\beta^{K+1}) \right) \tag{C.15}
\end{equation}

\begin{equation}
\leq \frac{1}{K} \left( h(\beta^1) - \hat{h} \right), \tag{C.16}
\end{equation}

where (C.16) follows from (C.15) by using the observation that $h(\beta^k) \downarrow \hat{h}$.

\section*{C.5. Additional details on Algorithm 3.}

More formally, we seek an upper bound to a simple variant of Problem (4.7):

\begin{equation}
\min_{\beta} \ h(\beta) := \sum_{i=1}^{p} \rho_{\gamma}(|\beta_i|) \text{ subject to } \|X^T(y - X\beta)\|_{\infty} \leq \delta, \beta_i = 0, i \in I_c, \tag{C.17}
\end{equation}

where $\text{Supp}(\widehat{\beta}^{(1)}) := \{ i : \widehat{\beta}_{i}^{(1)} \neq 0, i = 1, \ldots, p \} \subset I$, and $I_c$ is the complement of $I$. We assume, of course, that the feasible set in Problem (C.17) is nonempty. A simple method for constructing $I$, which we found to be quite useful in practice, is presented below. Let $B \subset \{1, \ldots, p\}$ and $B^c$ denote its complement. We define the following set:

$$F(B) := \left\{ \beta : \|X^T(y - X\beta)\|_{\infty} \leq \delta, \beta_i = 0, i \in B^c \right\}.$$

Let $\widehat{\alpha}^{(1)}, \widehat{\beta}^{(1)}$ be the solutions produced by Algorithm 1. Suppose we let $B$ denote the support of $\widehat{\beta}^{(1)}$; the size of $B$ is typically much smaller than $p$. If $F(B)$ is nonempty, we take $I = B$. Note, however, that $F(B)$ may be empty, because $\widehat{\alpha}^{(1)}, \widehat{\beta}^{(1)}$ are only approximately equal: $\widehat{\alpha}^{(1)} \approx \widehat{\beta}^{(1)}$. In this case, we need expand the set $B$, so that the set $F(B)$ becomes nonempty. There may be several ways to do this, but we found the following simple method to be quite useful in our numerical experiments.

1. If $F(B)$ is empty, we consider the set $\{ |\widehat{\alpha}_{i}^{(1)}|, i \in B^c \}$ and find the index of the largest element in this set, which we denote by: $\widehat{i} \in \arg\max_{i \in B^c} |\widehat{\alpha}_{i}^{(1)}|.$

2. Make $B$ larger by including this new feature $\widehat{i}$: we thus have $B \leftarrow B \cup \{\widehat{i}\}$.

3. Check if the resulting set $F(B)$ is nonempty, if not, we repeat the above steps until $F(B)$ becomes nonempty.

4. We let $I$ be the resulting set $B$ obtained upon termination: $I = B$.

Problem (C.17), which is an optimization problem of lower dimension than Problem (4.7), is found to deliver solutions that are better upper bounds to Problem (1.2). This also leads to better and numerically more robust solutions than those available directly from Algorithm 1. The general algorithmic framework via sequential linear optimization, presented in Section 4.1.2, can also be applied to obtain good upper bounds to Problem (C.17).
C.6. Tighter bounds on $\hat{\beta}_i$’s. The bounds described via (5.3) can be sharpened by making use of good upper bounds to the solution of Problem (1.2). Towards this end, we need to reformulate (2.2). Note that in Problem (2.2), if we take $M_U$ to be sufficiently large, say, $M_U^\infty$, then this will lead to a solution for Problem (1.2). We rewrite Problem (2.2) as follows:

$$\min_{\beta, z, \alpha} \alpha$$
subject to
$$\sum_{i=1}^p z_i \leq \alpha$$
$$-\delta \leq d_j - \langle q_j, \beta \rangle \leq \delta, \quad j = 1, \ldots, p$$
$$-M_U z_j \leq \beta_j \leq M_U z_j, \quad j = 1, \ldots, p$$
$$z_j \in \{0, 1\}, \quad j = 1, \ldots, p,$$

where the variables are $\beta, z \in \mathbb{R}^p$ and $\alpha \in \mathbb{R}$.

For a fixed $\alpha$, consider the feasible set of Problem (C.18):

$$(C.19) \quad S_\alpha = \left\{ (\beta, z) : \sum_{j=1}^p z_j \leq \alpha, \quad \|X^T(y - X\beta)\|_\infty \leq \delta \right\},$$

Observe that

$$(C.20) \quad S_\alpha \subset \overline{S}_\alpha := \left\{ (\beta, z) : \sum_{j=1}^p z_j \leq \alpha, \quad \|X^T(y - X\beta)\|_\infty \leq \delta \right\},$$

where $\overline{S}_\alpha$ is obtained by relaxing the binary variables $z_j \in \{0, 1\}$ into the continuous variables $z_j \in [0, 1]$, for all $j = 1, \ldots, p$. Noting that $S_\alpha \subset S_{\alpha'}$ for $\alpha \leq \alpha'$; and using this along with (C.20) we have

$$S_{\alpha^*} \subset S_{\alpha_0} \subset \overline{S}_{\alpha_0},$$

where $\alpha^*$ is the optimum value, and $\alpha_0$ is an upper bound to Problem (C.18), and hence $\alpha_0 \geq \alpha^*$. The above inequality leads to the following chain of inequalities:

$$(C.21) \quad \min_{(\beta, z) \in S_{\alpha^*}} \beta_i \geq \min_{(\beta, z) \in S_{\alpha_0}} \beta_i \geq \min_{(\beta, z) \in \overline{S}_{\alpha_0}} \beta_i := \mu_i^-(\alpha_0)$$
$$\max_{(\beta, z) \in S_{\alpha^*}} \beta_i \leq \max_{(\beta, z) \in S_{\alpha_0}} \beta_i \leq \max_{(\beta, z) \in \overline{S}_{\alpha_0}} \beta_i := \mu_i^+(\alpha_0).$$

The quantities at the right end of (C.21), i.e. $\mu_i^-(\alpha_0)$ and $\mu_i^+(\alpha_0)$, can be computed by solving a pair of linear optimization problems:

$$(C.22) \quad \mu_i^+(\alpha_0) := \max_{\beta} \beta_i \quad \text{subject to} \quad \|X^T(y - X\beta)\|_\infty \leq \delta,$$
$$\|\beta\|_\infty \leq M_U, \quad \|\beta\|_1 \leq M_U \alpha_0,$$
$$\mu_i^-(\alpha_0) := \min_{\beta} \beta_i \quad \text{subject to} \quad \|X^T(y - X\beta)\|_\infty \leq \delta,$$
$$\|\beta\|_\infty \leq M_U, \quad \|\beta\|_1 \leq M_U \alpha_0.$$
The quantities $\mu^-_i(\alpha_0)$ and $\mu^+_i(\alpha_0)$ are lower and upper bounds, respectively, for $\hat{\beta}_i$ — the bounds depend upon $\alpha_0$ and $\mathcal{M}_U$. Note that $\mu_i(\alpha_0) := \max \{\mu^+_i(\alpha_0), -\mu^-_i(\alpha_0)\}$ provides an upper bound to $|\hat{\beta}_i|$, which consequently leads to an improved estimate for $\|\beta\|_\infty$ — this suggests a way to adaptively refine $\mathcal{M}_U$, and, thus, $\mu^-_i(\alpha_0)$ and $\mu^+_i(\alpha_0)$, as we summarize below.

1. Start with a crude bound $\mathcal{M}_U < \infty$ and an upper bound $\alpha_0$, and set $\mathcal{M}^{\text{new}}_U = \mathcal{M}^{\text{old}}_U = \mathcal{M}_U$.

2. Solve Problems (C.22) to obtain $\mu^+_i(\alpha_0)$ and $\mu^-_i(\alpha_0)$ for $i \leq p$.

3. Compute: $\mathcal{M}_{U}^\star \leftarrow \max_{i=1,\ldots,p} \{\max \{\mu^+_i(\alpha_0), -\mu^-_i(\alpha_0)\}\}$. If $\mathcal{M}_{U}^\star < \mathcal{M}^{\text{new}}_U$, set $\mathcal{M}_{U}^{\text{old}} \leftarrow \mathcal{M}_{U}^{\text{new}}$, assign $\mathcal{M}_{U}^{\text{new}}$ to a value smaller than $\mathcal{M}_{U}^{\text{old}}$ and go to Step 2; otherwise assign $\mathcal{M}_{U}$ to the previous (larger) value $\mathcal{M}^{\text{old}}_U$ and go to Step 4.

4. Using the value of $\mathcal{M}_U$ obtained from Step 3, compute $\mathcal{M}^i_{U} = \max \{\mu^+_i(\alpha_0), -\mu^-_i(\alpha_0)\}$ for $i \leq p$.

### APPENDIX D: ADDITIONAL EXPERIMENTS

This section complements the experimental results shown in the main body of the paper. Table 4 is a much more elaborate version of the representative results displayed in Figure 5. Here, we consider three different values of SNR, and use an additional metric of performance. The results show that Discrete Dantzig Selector outperforms the $\ell_1$-Dantzig Selector based methods in terms of estimating the true underlying regression coefficients, and does so with better variable selection properties. An important advantage of the Discrete Dantzig Selector based methods is that they deliver models that are very sparse. The polished version of the Discrete Dantzig Selector is found to exhibit better statistical performance than the original Discrete Dantzig Selector estimator. Table 5 compares the polished versions of the Discrete Dantzig Selector and the $\ell_1$-Dantzig Selector, and finds that the performance of the former approach is significantly better.
Table 4

<table>
<thead>
<tr>
<th>Example-A (n=200, p=500)</th>
<th>Example-B (n=200, p=500)</th>
<th>Example-C (n=100, p=500)</th>
<th>Example-D (n=100, p=500)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metric</strong></td>
<td>SNS</td>
<td>L0-DS</td>
<td>L0-DS-Pol</td>
</tr>
<tr>
<td>$|\beta - \hat{\beta}|_2$</td>
<td>3</td>
<td>2.82 (0.14)</td>
<td>0.87 (0.27)</td>
</tr>
<tr>
<td>Variable Selection Error</td>
<td>3</td>
<td>4 (0.47)</td>
<td>4 (0.47)</td>
</tr>
<tr>
<td>Prediction Error</td>
<td>3</td>
<td>0.13 (0.01)</td>
<td>0.05 (0.03)</td>
</tr>
<tr>
<td>Number of Nonzeros</td>
<td>3</td>
<td>2 (0.2)</td>
<td>2 (0.2)</td>
</tr>
<tr>
<td>$|\hat{\beta} - \beta|_2$</td>
<td>7</td>
<td>0.98 (0.09)</td>
<td>0.25 (0.12)</td>
</tr>
<tr>
<td>Variable Selection Error</td>
<td>7</td>
<td>0 (0)</td>
<td>0</td>
</tr>
<tr>
<td>Prediction Error</td>
<td>7</td>
<td>0.03 (0.005)</td>
<td>0.01 (0.005)</td>
</tr>
<tr>
<td>Number of Nonzeros</td>
<td>7</td>
<td>28 (1)</td>
<td>28 (1)</td>
</tr>
<tr>
<td>$|\hat{\beta} - \beta|_2$</td>
<td>10</td>
<td>0.023 (0.008)</td>
<td>0.17 (0.020)</td>
</tr>
<tr>
<td>Variable Selection Error</td>
<td>10</td>
<td>0 (0)</td>
<td>0</td>
</tr>
<tr>
<td>Prediction Error</td>
<td>10</td>
<td>0.023 (0.008)</td>
<td>0.17 (0.020)</td>
</tr>
<tr>
<td>Number of Nonzeros</td>
<td>10</td>
<td>20 (1)</td>
<td>20 (1)</td>
</tr>
</tbody>
</table>

These tables show the statistical performance of four different methods: “L0-DS”, “L0-DS-Pol”, “L1-DS” and “Warm”, described in Section 7. The Discrete Dantzig Selector based methods deliver models with good accuracy in estimating the regression coefficients, and the estimated models are sparser than those for the l₁-based method and the method based on Algorithm 2, which is a heuristic strategy to approximate good upper bounds for the Discrete Dantzig Selector problem. The l₁-based methods become progressively better with higher SNR values.
Example-A

<table>
<thead>
<tr>
<th>Metric</th>
<th>SNR</th>
<th>L0-DS-Pol</th>
<th>L1-DS-Pol</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|\hat{\beta} - \beta^*|^2$</td>
<td>3</td>
<td>0.874 (0.274)</td>
<td>1.648 (0.242)</td>
</tr>
<tr>
<td>Variable Selection Error</td>
<td>3</td>
<td>5 (0.437)</td>
<td>27 (1.565)</td>
</tr>
<tr>
<td>Prediction Error</td>
<td>3</td>
<td>0.555 (0.045)</td>
<td>0.677 (0.035)</td>
</tr>
<tr>
<td>Number of Nonzeros</td>
<td>3</td>
<td>24 (1.2)</td>
<td>47 (2.35)</td>
</tr>
</tbody>
</table>

Example-F

<table>
<thead>
<tr>
<th>Metric</th>
<th>SNR</th>
<th>L0-DS-Pol</th>
<th>L1-DS-Pol</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|\hat{\beta} - \beta^*|^2$</td>
<td>3</td>
<td>17.372 (2.301)</td>
<td>24.679 (1.234)</td>
</tr>
<tr>
<td>Variable Selection Error</td>
<td>3</td>
<td>63.6 (3.18)</td>
<td>162.8 (8.14)</td>
</tr>
<tr>
<td>Prediction Error</td>
<td>3</td>
<td>0.857 (0.043)</td>
<td>0.979 (0.049)</td>
</tr>
<tr>
<td>Number of Nonzeros</td>
<td>3</td>
<td>97 (4.85)</td>
<td>203.6 (10.18)</td>
</tr>
</tbody>
</table>

Table 5: Tables comparing the polished version of Discrete Dantzig Selector with the polished version of $\ell_1$-Dantzig Selector. The statistical performance of latter is inferior, most likely due to its weaker variable selection properties.