Computing Estimators of Dantzig Selector type via Column and Constraint Generation

Rahul Mazumder* Stephen Wright† Andrew Zheng‡

August 17, 2019

Abstract

We consider a class of linear-programming based estimators in reconstructing a sparse signal from linear measurements. Specific formulations of the reconstruction problem considered here include Dantzig selector, basis pursuit (for the case in which the measurements contain no errors), and the fused Dantzig selector (for the case in which the underlying signal is piecewise constant). In spite of being estimators central to sparse signal processing and machine learning, solving these linear programming problems for large scale instances remains a challenging task, thereby limiting their usage in practice. We show that classic constraint- and column-generation techniques from large scale linear programming, when used in conjunction with a commercial implementation of the simplex method, and initialized with the solution from a closely-related Lasso formulation, yields solutions with high efficiency in many settings.

1 Introduction

We consider the prototypical problem of sparse signal recovery from linear measurements [9, 28, 10]: given a model matrix $X \in \mathbb{R}^{n\times p}$ with n samples and p features, response $y \in \mathbb{R}^n$ generated via the model $y = X\beta^0 + \epsilon$, where, $\beta^0 \in \mathbb{R}^p$ is sparse (that is, has few nonzero entries) and the errors are i.i.d. Gaussian with mean zero and variance σ^2 (i.e., $\epsilon \sim N(0, \sigma^2 I)$). We consider the case in which the number of variables is much larger than the number of samples ($p \gg n$) and our task is to estimate β^0 from (y, X), exploiting the knowledge that β^0 is sparse.

We assume throughout that the columns of X have been standardized to have mean zero and unit ℓ_2-norm. The ℓ_1-norm $\|\beta\|_1$ is often used as a convex surrogate to the cardinality of β, which is a count of the number of nonzero elements in β. The celebrated Dantzig Selector [9] approximates β^0 by minimizing $\|\beta\|_1$ subject to a constraint on the maximal absolute correlation between the features and the vector of residuals (given by $r := y - X\beta$). The optimization problem of this

*MIT Sloan School of Management, Operations Research Center and Center for Statistics, MIT.
†Department of Computer Sciences, University of Wisconsin-Madison.
‡Operations Research Center, MIT.

recovery problem is as follows:

$$(\ell_1\text{-DS}) \qquad \underset{\beta}{\text{minimize}} \quad \|\beta\|_1 \quad \text{s.t.} \quad \|X^T(y - X\beta)\|_\infty \leq \lambda, \tag{1}$$

where $\lambda > 0$ controls the data-fidelity term. Ideally, the value of λ should be such that the unknown signal vector β^0 is feasible, that is, $\|X^T\epsilon\|_\infty \leq \lambda$ holds (with high probability, say). The constraint in (1) can be interpreted as the ℓ_∞-norm of the gradient of the least squares loss $\frac{1}{2}\|y - X\beta\|_2^2$. An appealing property of the Dantzig Selector estimator is that it is invariant under orthogonal transformations of (y, X).

Problem (1) can be reformulated as a linear program (LP), and thus solved via standard LP algorithms and software (for example, commercial solvers like Gurobi and Cplex) for instances of moderate size. As pointed out in [4], efficient algorithms for ℓ_1-DS are scarce:

> "...Typical modern solvers rely on interior-point methods which are somewhat problematic for large scale problems, since they do not scale well with size."

Although important progress has been made on algorithms for ℓ_1-DS in subsequent years (see, for example, [4, 23, 30, 25, 22]), large-scale instances of (1) (with p of a million or more) still cannot be solved. The main goal of our work is to improve our current toolkit for solving ℓ_1-DS and related problems, bringing to bear some underutilized classical tools from large scale linear programming.

The Dantzig Selector is closely related to the Lasso [28], which combines a least squares data-fidelity term with an ℓ_1-norm penalty on β. While the Lasso and Dantzig Selectors yield different solutions [15], for suitably chosen regularization parameters, they both lead to estimators with similar statistical properties in terms of estimation error, under suitable assumptions on X, β^0, and σ (see [6]). A version of the Lasso that has the same objective as (1) is

$$\underset{\beta}{\text{minimize}} \quad \|\beta\|_1 \quad \text{s.t.} \quad \|y - X\beta\|_2 \leq \delta, \tag{2}$$

where $\delta \geq 0$ is a parameter that places a budget on the data fidelity, defined here as the ℓ_2-norm of the residuals. The explicit constraint on data fidelity makes this formulation appealing, but it poses computational challenges [4] because of the difficulty of projecting onto the constraint. The following alternative, unconstrained version has become the most popular formulation of Lasso:

$$(\text{Lasso}) \qquad \underset{\beta}{\text{minimize}} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \tag{3}$$

where $\lambda \geq 0$ is a regularization parameter that controls the ℓ_1-norm of β. (This is the formulation that we refer to as "Lasso" in the remainder of the paper.) There are several highly efficient algorithms for (3) (see, for example, [3, 17, 31]), making it an extremely effective tool in the context of sparse learning based on ℓ_1-minimization.

1.1 Algorithms for Lasso and Dantzig Selector: Fixing the Gap in Performance.

Common algorithms for solving (3) are based on proximal gradient methods [3, 31], coordinate descent [17], or homotopy methods [14]. Several efficient implementations of these methods are available publicly. There are key differences between the Lasso and ℓ_1-DS in terms of computational properties and associated solvers: ℓ_1-DS is essentially an LP whereas Lasso is a convex quadratic program (QP). Although LPs are generally thought to be easier to solve than QPs of a similar size, the Lasso QP can be solved with remarkable efficiency, at least when β is quite sparse and the matrix X has felicitous properties.

While first order optimization algorithms [4, 7] have also led to good algorithms to solve ℓ_1-DS, they are still much slower than Lasso. To illustrate, to compute a path of 100 solutions for a problem with $n = 200$, $p = 12,000$, `glmnet` [17] takes 0.24 seconds with minimal memory requirement on a modest desktop computer. On the other hand, for the same dataset (and machine), solving ℓ_1-DS for a path of 100 λ values by the parametric simplex method of [25] takes several minutes and requires at least 10GB of memory. The software package `flare` [22], based on the Alternating Direction Method of Multipliers (ADMM) [7] has prohibitive memory requirements and would not run on a modest desktop machine. The differences between solvers grow with problem size. As a consequence of the difficulties of solving the LP formulation, ℓ_1-DS remains somewhat underutilized, in spite of its excellent statistical properties. This paper seeks to address the striking difference in computational performance between the Lasso and the Dantzig Selector by proposing efficient methods for the latter. We make use of classical techniques from optimization: column generation and constraint generation. These techniques were first proposed as early as 1958 [16, 11] in the context of solving large scale LPs but, to our knowledge, have not been applied to ℓ_1-DS or its relatives discussed below.

Our approach exploits the sparsity that is typically present in the solution of ℓ_1-DS: at optimality, an optimal β will have few nonzeros. If we can identify the nonzero components efficiently, we may avoid having to solve a full LP formulation that includes all p components of β. Column generation starts by selecting a subset of components in β and solving a reduced version of (1) that includes only these components (that is, it fixes the components of β that are *not* selected to zero). If the optimality conditions for the full problem are not satisfied by the solution of the reduced LP, more components of β are added to the formulation in a controlled way, and a new reduced LP is solved, using the previous solution as a warm start. The process is repeated until optimality for the full LP is obtained. Whenever new components are added to the reduced LP, we add new columns to the constraint matrix, hence the name *column generation*.

We make use too of another key property of ℓ_1-DS: redundancy of the constraints in (1). Typically, the number of components of $X^T(y - X\beta)$ that are at their bounds of $-\lambda$ and λ at the solution is small, of the same order as the number of nonzero components of β at the solution. This observation suggests a procedure in which we solve a reduced LP with just a subset of constraints enforced. We then check the constraints that were not included in this formulation to see if they are violated by the solution of the reduced LP. If so, we add (some of) these violated constraints to the LP formulation, and solve the modified reduced LP. This process is repeated until a solution of the

original problem is obtained. This procedure is known as *constraint generation*.

While column generation and constraint generation are commonly used as separate entities to solve large scale LPs, it makes sense to use them jointly in solving ℓ_1-DS, and a combination of the two strategies can be implemented with little complication. The procedures can benefit from good initial guesses of the nonzero components of β and the active constraint set for (1). We use efficient Lasso solvers to obtain these initial guesses.

1.2 Other Examples

Several other examples of sparse linear models are also amenable to column and constraint generation techniques. These include basis pursuit denoising [10] and a Dantzig selector version of one-dimensional total variation denoising (also known as fused Lasso) [24, 29]. Each of these problems can be formulated as a linear program task and, like ℓ_1-DS, they are computationally challenging.

Basis Pursuit. The noiseless version of ℓ_1 sparse approximation, popularly known as Basis Pursuit [10], is given by the following optimization problem:

$$\underset{\beta}{\text{minimize}} \quad \|\beta\|_1 \quad \text{s.t.} \quad y = X\beta, \tag{4}$$

which can be formulated as an LP. This problem can be interpreted as a limiting version of (3) as $\lambda \to 0+$. It may be tempting to solve (3) for a small value of $\lambda \approx 0$ (possibly with warm-start continuation) to obtain a solution to (4). However, this approach is often inefficient in practice, because obtaining accurate solutions to the Lasso becomes increasingly expensive as $\lambda \downarrow 0$. It has been pointed out in [13] that solving (4) to high accuracy using existing convex optimization solvers or specialized iterative algorithms is a daunting task, for large instances. Our own experiments show that current solvers based on ADMM fail to solve (4) for $p \geq 10^4$ (with $n < 1000$), while our proposed approach, described below, solves problems with $p \approx 4 \times 10^5$ within 3-4 minutes. Our approach relies on column generation, exploiting the familiar observation that the solution β of (4) is sparse.

Fused Dantzig Selector. The Fused Lasso [29] or the total variation penalty [27] is a commonly used ℓ_1-based penalty that encourages the solution to be (approximately) piecewise constant. The unconstrained formulation of this problem is

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|D^{(0)}\beta\|_1, \tag{5}$$

where $\lambda \geq 0$ is a regularization parameter and $D^{(0)} \in \mathbb{R}^{(p-1)\times p}$ is the first order difference operator matrix, defined by

$$D^{(0)}\beta = (\beta_2 - \beta_1, \beta_3 - \beta_2, \ldots, \beta_p - \beta_{p-1})^T, \tag{6}$$

which represents differences between successive components of β. As we show in Section 2.4, (5) can be expressed as a Lasso problem of the standard form (3), with a modified model matrix $\tilde{X} \in \mathbb{R}^{n \times p-1}$ and response $\tilde{y} \in \mathbb{R}^n$ [1]. This suggests a natural Dantzig Selector analog of the fused Lasso problem:

$$\underset{\alpha \in \mathbb{R}^{p-1}}{\text{minimize}} \quad \|\alpha\|_1 \quad \text{s.t.} \quad \|\tilde{X}^T(\tilde{y} - \tilde{X}\alpha)\|_\infty \leq \lambda, \tag{7}$$

a formulation that is amenable to the column and constraint generation methods developed in this paper. In the special case of $X = \mathbb{I}$, the problem has additional structure that we can exploit to solve instances with $p \approx 5 \times 10^5$, well beyond the capabilities of alternative methods.

1.3 Related work and Contributions

The Dantzig Selector formulations presented here — (1), (4), and (7) — can all be expressed as LPs and solved with interior point methods or simplex-based methods, as implemented in commercial solvers [2] (Gurobi, Cplex, Mosek, XPress, etc) or open-source codes (GLKP, lpsolve, among others). Specialized implementations for ℓ_1-DS and Basis Pursuit have been investigated for several years. An interior point method was used in [9], a first-order method for a regularized version of ℓ_1-DS was described in [4][3], and methods based on ADMM were discussed in [23, 30, 7]. Using a homotopy continuation approach, [20] extend the framework of LARS [14] to find the solution path to ℓ_1-DS, which is piecewise linear in λ. Homotopy continuation methods applicable to ℓ_1-DS have also been proposed by [2, 8, 26], but these works do not appear to use column and constraint generation methods, which are the main focus of our work. For the ℓ_1-DS problem, the algorithms of [2, 8, 26] compute the full $p \times p$ matrix $X^T X$ at the outset. This operation is memory-intensive, so these approaches can handle values of p only up to a few thousands on a modest desktop computer.

Our methods solve the problems (1), (4), and (7) at a single value of the regularization parameter, but they can be extended to solve these problems on a grid of regularization parameters via a warm start continuation strategy. We show that the classical tools of column and constraint generation can be effective in solving large-scale instances of these problems. Our work is related to the proposal of [12] who explored column and constraint generation to solve regularized linear SVM problems (with a hinge loss) that can be expressed as LPs. (Regularizers considered in [12] include the ℓ_1-norm, group ℓ_1-norm, and the Slope penalty.) Our Dantzig Selector problems have structural properties different from the SVM problems. They also have the unique advantage that they can be initialized using Lasso. This fact plays an important role in the practical computational efficiency of our approaches.

Our methods are based on the simplex algorithm, which is better at making use of the available warm-start information than interior-point methods. A memory-friendly version of Gurobi's simplex solver, applied to an LP formulation of (1) that avoids formation of $X^T X$ by using auxiliary

[1] We define $(\tilde{y}, \tilde{X})$ as follows. Define $D = [e_1^T; D^{(0)}] \in \mathbb{R}^{p \times p}$, where $e_1 = (1, 0, 0, \ldots, 0)^T$, and define $H = D^{-1}$. For any $A \subset \{1, 2, \ldots, p\}$, let H_A be the submatrix of H containing the columns indexed by A, and let P_A denote the projection operator onto the column space of H_A. We set $\tilde{y} := (I - P_A)y$ and $\tilde{X} := (I - P_A)XH_B$, with $A = \{1\}$ and $B = \{2, \ldots, p\}$.

[2] Commercial solvers such as Gurobi, Cplex, Mosek are free for academic use.

[3] The authors add a small ridge penalty to the objective and optimize the dual via gradient methods

variables, works well for ℓ_1-DS with n in the hundreds and p in the thousands. In fact, this approach can be faster than some specialized algorithms [26, 4, 23]. We show simplex performance can be improved substantially by using column and constraint generation when $p \gg n$, for problems in which the underlying solution is sufficiently sparse. We refer to our framework as Dantzig-Linear Programming (DantzigLP for short). Because we use a simplex engine as the underlying solution, a primal-dual solution is available at optimality. If we decide to terminate the algorithm early due to computational budget constraints, our framework delivers a certificate of suboptimality. DantzigLP can solve instances of the ℓ_1-DS problem with $n \approx 10^3$ and $p \approx 10^6$; Basis Pursuit with $n \approx 10^3$ and $p \approx 10^5$; and Fused Lasso with $n = p \approx 10^6$; all within a few minutes and with reasonable memory requirements. To our knowledge, problems of this size are beyond the capabilities of current solvers. A Julia implementations of our DantzigLP framework can be found at `https://github.com/atzheng/DantzigLP`.

Notation. We denote $[n] := \{1, 2, \dots n\}$. The identity matrix is denoted by $\mathbb{I}$ (with dimension understood from the context). When operating on vector-valued operands $u, v \in \mathbb{R}^n$, the inequality $u \leq v$ denotes elementwise comparison. For any matrix $X \in \mathbb{R}^{n \times p}$ and index sets $I \subset [n]$ and $J \subset [p]$, we denote by $X_{I,J}$ the $|I| \times |J|$ submatrix of X that consists of the rows of X indexed by I and the columns of X indexed by J. The notation $X_{*,J}$ denotes a submatrix consisting of all rows of X but only the columns indexed by J (A similar convention applies to $X_{I,*}$). For a vector $v \in \mathbb{R}^n$ and a set $S \subset [n]$, v_S denotes the subvector of v restricted to the indices in S. The notation $\mathbf{1}$ denotes the vector $(1, 1, \dots, 1)^T$, whose length is defined by the context.

2 Methodology

2.1 Column and Constraint Generation for Large-Scale LP

Column generation [16, 11, 5] is a classical tool to solve large scale LPs with a large number of variables and a relatively small number of constraints, when we anticipate an optimal solution with few nonzero coefficients. The basic idea is simple: we solve a small LP involving just a subset of columns, and incrementally add columns into the model, re-solving the LP after each addition, until optimality conditions for the original problem are satisfied. *Constraint generation* is used when the number of constraints is large relative to the number of variables, when we expect a relatively small subset of the constraints to be active at optimality.

For the sake of completeness, we provide an overview of these techniques in this section, referring the reader to [5] for a more detailed treatment.

Given problem data $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ with decision variable $x \in \mathbb{R}^n$, we consider the LP (`Primal-Full`), whose dual (`Dual-Full`) has decision variable $v \in \mathbb{R}^m$. We assume that A has full rank.

$$\begin{array}{lll} \underset{x}{\text{minimize}} & c^T x & \\ \text{s.t.} & Ax \geq b & (\texttt{Primal-Full}) \\ & x \geq 0 & \end{array} \qquad \begin{array}{lll} \underset{v}{\text{maximize}} & b^T v & \\ \text{s.t.} & A^T v \leq c & (\texttt{Dual-Full}) \\ & v \geq 0. & \end{array}$$

We will assume that (`Primal-Full`) has a finite optimal solution. By LP duality theory, the dual also has a solution with the same optimal objective value.

The solutions of (`Primal-Full`) and (`Dual-Full`) can be derived from the solutions to reduced problems of the following form, for some index sets $I \subset [m]$ and $J \subset [n]$:

$$\begin{array}{lll} \underset{x_J}{\text{minimize}} & c_J^T x_J & \\ \text{s.t.} & (A_{I,J})x_J \geq b_I & (\texttt{Primal}(I,J)) \\ & x_J \geq 0, & \end{array} \qquad \begin{array}{lll} \underset{v_I}{\text{maximize}} & b_I^T v_I & \\ \text{s.t.} & A_{I,J}^T v_I \leq c_J & (\texttt{Dual}(I,J)) \\ & v_I \geq 0. & \end{array}$$

The subsets I and J are not known in advance; the simplex method can be viewed as a search for these subsets in which typically one element is changed at each iteration. Sufficient conditions for the solutions x_J of (`Primal`(I,J)) and v_I of (`Dual`(I,J)) to be extendible to solutions of (`Primal-Full`) and (`Dual-Full`) are that:

$$A_{i,J}x_J \geq b_i, \text{ for all } i \in [m] \setminus I; \quad A_{I,j}^T v_I \leq c_j, \text{ for all } j \in [n] \setminus J. \tag{8}$$

If these conditions are satisfied, we obtain solutions x^* and v^* of (`Primal-Full`) and (`Dual-Full`), respectively, by setting $x_J^* = x_J$ and $x_{J^c}^* = 0$, and $v_I^* = v_I$ and $v_{I^c}^* = 0$.

Column and constraint generation are techniques for systematically expanding the sets I and J until the optimality conditions (8) are satisfied by the solutions of (`Primal`(I,J)) and (`Dual`(I,J)). At each iteration, we solve a reduced problem of this form, then seek indices i and j for which the conditions (8) are violated. Some of these indices are added to the sets I and J, and the new (slightly larger) versions of (`Primal`(I,J)) and (`Dual`(I,J)) are solved using a variant of the simplex method, typically warm-started from the previous reduced problem. An outline of the approach is shown in Algorithm 1.

Algorithm 1 Constraint and Column Generation to solve (`Primal-Full`) and (`Dual-Full`)

Initialize $I \subset [m]$, $J \subset [n]$;
repeat
  Solve (`Primal`(I,J)) and (`Dual`(I,J)) to obtain x_J and v_I;
  Choose $I_s \subset I_v := \{i \in [m] \setminus I : A_{i,J}x_J < b_i\}$;
  Choose $J_s \subset J_v := \{j \in [n] \setminus J : A_{I,j}^T v_I > c_j\}$;
  Set $I \leftarrow I \cup I_s$ and $J \leftarrow J \cup J_s$;
until $I_v = \emptyset$ and $J_v = \emptyset$;
Set $x_{J^c} = 0$ and $v_{I^c} = 0$ and **terminate**.

Many variants are possible within this framework. One could define the sets I_s and J_s to contain

only the smallest valid index, or the most-violated index. More commonly, I_s and J_s are chosen to have cardinality greater than 1 where possible. In large-scale problems (analogous to partial pricing in the simplex method), not all checks in (8) are even performed. When A is too large to store in memory, for example, we can stream columns of A to calculate the quantities $A_{I,j}^T v_I - c_j$ until enough have been calculated to define J_v.

When Algorithm 1 is implemented with $I = [m]$ but J a strict subset of $[n]$, it reduces to column generation. (In this case, I_s and I_v are null at every iteration.) Similarly, when I is a strict subset of $[m]$ but $J = [n]$, Algorithm 1 reduces to constraint generation.

The success of constraint and column generation hinges on the ability to generate initial guesses for I and J that requires few additional iterations of Algorithm 1 to identify the solutions of (`Primal-Full`) and (`Dual-Full`).

The DantzigLP Framework. Combining column and constraint generation LP techniques with methods for finding good initializations for the initial column and constraint sets I and J, we develop DantzigLP, a general framework for solving large-scale versions of the the Dantzig Selector-type problems described in Section 1. Initializations for I and/or J are obtained typically by solving the Lasso variant of a given problem. This basic approach can be tailored to a large range of problems, and in many cases the problem structure admits fast algorithms for both initialization and column and constraint generation.

All of the Dantzig-type problems described in Section 1 (except for Basis Pursuit) have a tunable regularization parameter λ. Practitioners often wish to compute the estimator for a grid of λ values specified a-priori. DantzigLP makes this process efficient by leveraging a simplex-based solver's warm start capabilities. Given the solution for one value of λ, DantzigLP can efficiently find the solution for its neighboring value of λ (within the column and constraint generation framework). Repeating this process yields a path of solutions.

2.2 The Dantzig Selector

We show how the DantzigLP framework applies to ℓ_1-DS. We present an LP formulation for ℓ_1-DS—the primal (9) and its corresponding dual (10) are as follows:

$$\begin{aligned} \underset{\beta^+,\beta^-,r}{\text{minimize}} \quad & \sum_{i\in[p]} (\beta_i^+ + \beta_i^-) \\ \text{s.t.} \quad & -\mathbf{1}\lambda \le X^T r \le \lambda \mathbf{1} \\ & r = y - X(\beta^+ - \beta^-) \\ & \beta^+, \beta^- \ge 0 \end{aligned} \tag{9}$$

$$\begin{aligned} \underset{\nu^+,\nu^-,\alpha}{\text{maximize}} \quad & -\sum_{i\in[p]} \lambda(\nu_i^+ + \nu_i^-) - \alpha^T y \\ \text{s.t.} \quad & -\mathbf{1} \le X^T \alpha \le \mathbf{1} \\ & X(\nu^+ - \nu^-) + \alpha = 0 \\ & \nu^+, \nu^- \ge 0. \end{aligned} \tag{10}$$

The primal problem (9) has decision variables $\beta^+, \beta^- \in \mathbb{R}^p$ denoting the positive and negative parts of β, and $r \in \mathbb{R}^n$ corresponding to the residual vector. The dual variables $\nu^+, \nu^- \in \mathbb{R}^p$ correspond to the inequality constraints $-\lambda \le X^T r$ and $X^T r \le \lambda$ respectively, and α corresponds to the

equality constraint $r = y - X(\beta^+ - \beta^-)$. At optimality, the following complementarity conditions hold:

$$\nu^+ \circ (X^T\alpha - \lambda\mathbf{1}) = 0 \quad \text{and} \quad \nu^- \circ (X'\alpha + \lambda\mathbf{1}) = 0,$$

where, "$\circ$" denotes componentwise multiplication. Therefore, $X^T_{*,i}r < \lambda \implies \nu^+_i = 0$ and $-X^T_{*,i}r < \lambda \implies \nu^-_i = 0$. Formulation (9) does not require computation and storage of the memory-intensive $p \times p$ matrix X^TX; this is avoided by introducing auxiliary variable r. Moreover, the related Lasso problem (3) gives us reason to expect that this problem is a good candidate for both constraint and column generation. Optimality conditions for solution β^L of (3) can be written as follows:

$$r^L := y - X\beta^L, \quad X^T_{*,j}r^L \in \begin{cases} \{\lambda\} & \text{if } \beta^L_j > 0 \\ [-\lambda, \lambda] & \text{if } \beta^L_j = 0 \\ \{-\lambda\} & \text{if } \beta^L_j < 0. \end{cases} \tag{11}$$

This suggests that, for the Lasso solution β^L at least, the number of active constraints in (9) is similar to the number of nonzero components in β^L, which is typically small. If the solution of the Dantzig selector has similar properties to the Lasso solution, then we would expect both the number of nonzero components in the solution of (9) and the number of active constraints to be small relative to the dimensions of the problem. (The papers [20] and [1] demonstrate conditions under which the Dantzig and Lasso solution paths, traced as functions of the regularization parameter λ, are in fact identical.)

For a subset $J \subset [p]$ of columns of X and $I \subset [p]$ of rows of X (note that I and J need not be the same), we define a reduced column and constraint version of (9) as follows:

$$\begin{aligned} \underset{\beta^+, \beta^-, r}{\text{minimize}} \quad & \sum_{j \in J} \left(\beta^+_j + \beta^-_j\right) \\ \text{s.t.} \quad & y - X_{*,J}(\beta^+_J - \beta^-_J) = r \\ & (X_{*,I})^T r \le \lambda\mathbf{1} \\ & (X_{*,I})^T r \ge -\lambda\mathbf{1} \\ & \beta^+, \beta^- \ge 0. \end{aligned} \tag{DS(I,J)}$$

Our constraint and column generation strategy for solving ℓ_1-DS solves problems of this form at each iteration, initializing I and J from the Lasso solution, and alternately expanding I and J until a solution of ℓ_1-DS is identified. We initialize J to be the subset of components $j \in [p]$ for which $\beta_j \neq 0$, while I is the set of indices $i \in [p]$ such that $|X^T_{*,i}r| = \lambda$. The strategy is specified as Algorithm 2.

Note that each time we solve DS(I,J), we warm-start from the previous instance. The violation checks that define I_v and J_v can be replaced by relaxed versions involving a tolerance ϵ. That is, we can define

$$I_v \leftarrow \{i \in [p] \setminus I \,:\, |X^T_{*,i}r| > \lambda + \epsilon\}, \quad J_v \leftarrow \{j \in [p] \setminus J \,:\, |X^T_{*,j}\alpha| > 1 + \epsilon\}. \tag{12}$$

Algorithm 2 Constraint and Column Generation to solve ℓ_1-DS

Solve (3) to obtain initial I and J;
loop
  Calculate $I_v \leftarrow \{i \in [p] \setminus I \, : \, |X^T_{*,i} r| > \lambda\}$;
  if $I_v \neq \emptyset$ **then**
    Choose $\emptyset \neq I_s \subset I_v$; Set $I \leftarrow I \cup I_s$; Solve DS(I, J);
  else
    Using α from the dual solution of DS(I, J), calculate $J_v \leftarrow \{j \in [p] \setminus J \, : \, |X^T_{*,j} \alpha| > 1\}$;
    if $J_v \neq \emptyset$ **then**
      Choose $\emptyset \neq J_s \subset J_v$; Set $J \leftarrow J \cup J_s$; Solve DS(I, J);
    else
      terminate.
    end if
  end if
end loop

(We use $\epsilon = 10^{-4}$ in our experiments.)

Computing a path of solutions. We can extend Algorithm 3 to solve ℓ_1-DS for a path of λ values of the form $\Lambda = \{\lambda_i, i \in [k]\}$ in decreasing order. We first obtain the Lasso solution for the smallest λ value, which typically corresponds to the densest solution in the Lasso path. (This strategy, which is the opposite of that used in solvers for Lasso — see for example [31] — reduces the overhead of continuously updating our LP model with new columns and constraints as we move across the λ-path.) The Lasso solution can be used to supply initial guesses of index sets I_0 and J_0 for the first value λ_1. The final index sets I_1 and J_1 for λ_1 can then be used as initial guesses for λ_2, and so on. Optimal basis information (basis matrices and their factorizations) for each value of λ can also be carried over to the next value.

Existing approaches. Many interesting approaches have been presented to solve the ℓ_1-DS problem. [4] presents a general framework for by solving a regularized version of the problem with first order gradient based methods, but these methods do not scale well. [23, 30] use ADMM for ℓ_1-DS, which may lead to large feasibility violations. Homotopy methods were presented in [26] and were shown to outperforms existing algorithms for ℓ_1-DS. Other homotopy based methods of appear in [2, 8]. All the homotopy algorithms [2, 8, 26] compute the matrix $X^T X$ at the outset; this memory intensive computation precludes the possibility of solving large scale instances $p \approx 10^6$ with $p \gg n$. Column and constraint generation has not been considered in these aforementioned papers.

Computational results of our methods are presented in Section 3.

2.3 Basis Pursuit

We study how our DantzigLP framework can be applied to (4), which admits the following LP representation. (Recall that we assume that the feasible set is nonempty.)

$$\underset{\beta^+,\beta^-}{\text{minimize}} \quad \sum_{j\in[p]}(\beta_j^+ + \beta_j^-) \qquad \text{s.t.} \qquad y = X(\beta^+ - \beta^-), \;\; \beta^+ \geq 0, \;\; \beta^- \geq 0. \tag{BP-Full}$$

Consider a subset of features $J \subset [p]$ and a restriction of (BP-Full) to the indices in J. We obtain the following reduced primal (BPP(J)) and dual (BPD(J)):

$$\begin{array}{ll} \underset{\beta_J^+,\beta_J^-}{\text{minimize}} & \sum_{j\in J}(\beta_j^+ + \beta_j^-) \\ \text{s.t.} & y = X_{*,J}(\beta_J^+ - \beta_J^-) \\ & \beta_J^+, \beta_J^- \geq 0 \end{array} \tag{BPP(J))}$$

$$\begin{array}{ll} \underset{v}{\text{maximize}} & v^T y \\ \text{s.t.} & X_{*,J}^T v \leq \mathbf{1} \\ & -X_{*,J}^T v \leq \mathbf{1}. \end{array} \tag{BPD(J))}$$

For the column generation procedure, we need a subset J (preferably of small size) for which (BPP(J)) is feasible. Accordingly, we seek an approximation to the largest value of λ for which the Lasso yields a a feasible solution for (BPP(J)). We find such a value by solving the Lasso for a sequence of decreasing values of λ, checking after each solution whether the resulting solution is feasible for (BPP(J)), and if so, defining J to be the support obtained of this solution.

If we cannot find a set J for which (BPP(J)) is feasible, we append the current J with an additional $n - |J|$ columns to obtain a feasible solution[4]. As before, the Lasso continuation approach is used just to obtain a good initialization for J for our column generation framework, not to obtain a solution for BP.

The approach is summarized in Algorithm 3; and computational results are presented in Section 3.

Algorithm 3 Column Generation for Basis Pursuit (BP-Full)

Solve a sequence of Lasso problems (3) to obtain initial J;
loop
  Solve (BPP(J)), with $v \in \mathbb{R}^n$ as the dual solution;
  Calculate $J_v := \{j \in [p] \setminus J \, : \, |X_{*,J}^T v| > 1\}$;
  if $J_v = \emptyset$ **then**
    terminate;
  else
    Choose $\emptyset \neq J_s \subset J_v$ and set $J \leftarrow J \cup J_s$;
  end if
end loop

[4]If the entries of X are drawn from a continuous distribution, then $|J| = n$ will lead to a feasible solution for (BPP(J)). In all our experiments, the Lasso continuation approach did lead to a J for which (BPP(J)) was feasible. Note that the Lasso path often leads to solutions for which the number of nonzeros exceeds n.

2.4 The Fused Dantzig Selector

2.4.1 Signal estimation

We discuss how the DantzigLP framework can be extended to the Dantzig analog of (5) with $X = \mathbb{I}$ (the identity matrix), which is

$$\frac{1}{2}\|y - \beta\|_2^2 + \lambda\|D^{(0)}\beta\|_1, \tag{13}$$

where $D^{(0)}$ is defined in (6). To express this problem in Lasso form, we define the $n \times n$ matrix $D = [e_1^T; D^{(0)}]$ (where $e_1 = (1, 0, 0, \ldots, 0)^T$), which has full rank. Its inverse $H = D^{-1}$ is

$$H_{i,j} = \begin{cases} 1 & \text{if} \quad j = 1 \\ i - j & \text{if} \quad i > j \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

We can now rewrite (5) in terms of the variables $\alpha := D\beta$, and recover the solution β of (13) by setting $\beta = H\alpha$. Defining $A := \{1\}$ and $B := \{2, \ldots, n\}$, we write (13) as follows:

$$\underset{\alpha_A, \alpha_B}{\text{minimize}} \ \frac{1}{2}\|y - H_A\alpha_A - H_B\alpha_B\|_2^2 + \lambda\|\alpha_B\|_1. \tag{15}$$

This formulation differs slightly from the standard Lasso problem in that the ℓ_1 penalty term excludes α_A. The Dantzig analog of (15) is

$$\underset{\alpha_A, \alpha_B}{\text{minimize}} \ \|\alpha_B\|_1 \ \text{ s.t. } \ \|H_B^T(y - H_A\alpha_A - H_B\alpha_B)\|_\infty \le \lambda, \ H_A^T(y - H_A\alpha_A - H_B\alpha_B) = 0. \tag{16}$$

(Note the constraint $H_A^T(y - H_A\alpha_A - H_B\alpha_B) = 0$, which arises as an optimality condition for α_A in (15).) Recalling that $H^{-1} = D$, and introducing auxiliary variables $\beta = H\alpha$, $r = y - \beta$, and $g = D^T r$, we rewrite (16) as follows:

$$\begin{aligned} \underset{r, \alpha^+, \alpha^-, \beta, \nabla}{\text{minimize}} \quad & \sum_{i \in B} \left(\alpha_i^+ + \alpha_i^-\right) \\ \text{s.t.} \quad & \beta = D(\alpha^+ - \alpha^-) \\ & r = y - \beta \\ & g = D^T r \\ & g_A = 0 \\ & g_B \le \lambda \mathbf{1} \\ & g_B \ge -\lambda \mathbf{1} \\ & \alpha^+, \alpha^- \ge 0. \end{aligned} \tag{17}$$

We apply column and constraint generation to formulation (17): Column generation because of sparsity in α_B, and constraint generation because few of the constraints $g_B \in [-\lambda\mathbf{1}, \lambda\mathbf{1}]$ are expected

to be active at optimality. The formulation (17) exploits the following key characteristics:

- $H^{-1} = D$ is banded. Writing constraints in terms of D rather than H yields a constraint matrix with $O(n)$ nonzero entries. (A direct LP representation for ℓ_1-DS with $n = p$ as in (9) would lead to a constraint matrix with $O(n^2)$ nonzeros.)
- Each component of α^+ and α^- appears in exactly one equality constraint, reducing the cost of computing each reduced cost to $O(1)$ time ($\bar{c}_j = 1 - v_j$ for each α_j^+, and $\bar{c}_j = 1 + v_j$ for each α_j^-) — much cheaper than the corresponding cost for a general ℓ_1-DS problem, which requires the $O(n)$ operation $v^T X_j$.
- We can check each constraint violation $g_i \in [-\lambda, \lambda]$ in $O(1)$ time, compared to a general ℓ_1-DS problem which requires the $O(n)$ operation $X_{*,i}^T r$ for each constraint.

To obtain a good initialization for (17), we use the solution of (13), which can be computed efficiently via dynamic programming [21] at $O(n)$ cost.

2.4.2 Beyond signal estimation: Regression

We now consider problem (5) with general design matrix X.

As in (15), we introduce a new variable $\alpha = H\beta$. We let H_A be the submatrix of H containing the columns indexed by A, and P_A denote the projection operator onto the column space of H_A. With this notation in place, we rewrite (5) in standard Lasso form with model matrix $\tilde{X} = (I - P_A)XH$ and response $\tilde{y} = (I - P_A)y$. The corresponding Dantzig Selector problem (7) is an instance of ℓ_1-DS problem with problem data $(\tilde{y}, \tilde{X})$. Since (7) lacks the structure as the signal estimation problem in Section 2.4.1 (where $X = \mathbb{I}$), we apply the DantzigLP procedure (Algorithm 3) to this problem, with a special initialization. The initial point is obtained by solving (5) using a proximal gradient method [3]. Each step of this method requires calculation of the proximal map defined by

$$\Theta(u_k) := \arg\min_{\beta} \quad \frac{L}{2} \left\| \beta - \left(u_k - \frac{1}{L} \nabla f(u_k) \right) \right\|_2^2 + \lambda \|D^{(0)} \beta\|_1,$$

where L is the largest eigenvalue[5] of $X^T X$ and $f(u) = 1/2\|y - Xu\|_2^2$. The operator $\Theta(u_k)$ is computed via dynamic programming [21], which is highly efficient. If we set $u_k = \beta_k$ and $\beta_{k+1} = \Theta(\beta_k)$ we get the usual (unaccelerated) proximal gradient algorithm. We use the accelerated variant which enjoys an improved convergence rate. It sets $\tilde{u}_{k+1} = \Theta(u_k)$ where $u_{k+1} = \tilde{u}_k + q_{k-1}(\tilde{u}_k - \tilde{u}_{k-1})/q_{k+1}$ and $q_{k+1} = (1 + \sqrt{1 + 4q_k^2})/2$. The sequence is initialized with $u_1 = \tilde{u}_0 = 0$ and $q_1 = 1$.

This proximal gradient approach to solving (5) is much faster than a coordinate descent procedure [17] applied to the Lasso reformulation of (5) with problem data $(\tilde{y}, \tilde{X})$.

[5] This can be computed via the power method or by computing the largest singular value of X with cost $O(\min\{n,p\}^2 \max\{n,p\})$.

3 Computational Results

This section presents computational results showing the performance of our proposed DantzigLP framework for the ℓ_1-DS problem (Section 3.1), Basis Pursuit (Section 3.2), and the Fused Dantzig Selector (Section 3.3).

3.1 Computational experience with Dantzig Selector

We implement DantzigLP in Julia[6], using Gurobi's dual simplex solver[7] as the LP solver and Lasso.jl (a Julia implementation of `glmnet` [17]) as the Lasso solver. At each iteration of column and constraint generation, we add up to 30 columns with the most negative reduced costs, and up to 50 of the most violated constraints. Unless stated otherwise, we solve problems to within a tolerance of 10^{-4} for both column and constraint generation violations.

3.1.1 Experiments on synthetic datasets

Data Generation. Our first set of experiments were performed on synthetic data. The rows of X are drawn from a multivariate Gaussian distribution $\text{MVN}(0, \Sigma)$ with mean zero and covariance Σ where, $\Sigma_{ij} = \rho$ for all $i \neq j$ and $\Sigma_{ii} = 1$. We then sparsify[8] X by setting its entries to 0 independently with probability π; and finally normalize so that the columns of X to have unit ℓ_2-norm. To generate the true β^0, we choose a set $S \subset [p], |S| = n/5$ and set entries of β^0_S as i.i.d. draws from a standard Gaussian distribution. The remaining components $\beta^0_{S^c}$ are set to zero. We then generate $y = X\beta^0 + e$ with $e_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$; and σ^2 is chosen so as to achieve a signal-to-noise ratio (SNR) of 10. (Note that we define SNR as the ratio $\text{Var}(X\beta)/\sigma^2$.)

Comparison with ADMM. A popular method for the ℓ_1-DS problem is based on ADMM [23, 30, 7]. In Figure 1, we compare DantzigLP with `flare` [23], a publically available implementation of ADMM, plotting the violation in feasibility ($\max\{\|X^T(y - X\beta)\|_\infty - \lambda, 0\}$) and the difference between the objective function from its optimal value ($|\|\beta\|_1 - \|\beta^*\|_1|$) as a function of runtime. (We use absolute value in the objective measure because infeasibility can result in a β with smaller norm than the solution β^*.) We find that ADMM is slow by both measures (an observation made also by [26]) and that DantzigLP is much faster. Each path in Figure 1 represents a single simulated problem instance with data generated by the means above, where $\rho = 0$, $\pi = 0$, $n = 200$, and $p = 1000$. We set $\lambda = \|X^T e^0\|_\infty$ where, $e^0 = (y - X\beta^0)$. Since ADMM has difficulty finding a solution with an absolute feasibility violation of less than 0.01 in many cases, we do not consider it further in the experiments below.

[6] Our Julia/JuMP implementation can be found at `https://github.com/atzheng/DantzigLP`.

[7] We use Gurobi version 7.5 in our experiments. All computations were performed on a Mac Pro desktop machine with specs: 2.7GHz 12-Core Intel Xeon E5. Unless otherwise specified the memory budget was 64GB of RAM.

[8] Sparsification may destroy the correlation structure among columns of X.

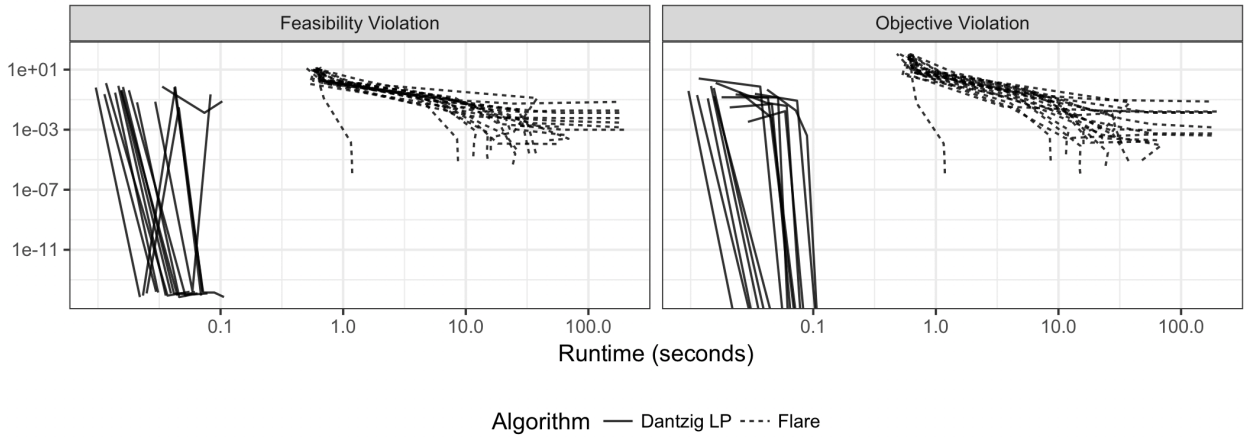

Figure 1: Feasibility and objective violations as a function of runtime for DantzigLP (including time required for Lasso initialization) and the ADMM implementation Flare, for the ℓ_1-DS problem. In all instances, DantzigLP reaches an optimal solution with zero feasibility violation within a few iterations. In comparison, Flare takes much longer to improve the objective and feasibility violations, often failing to converge even after hundreds of seconds.

Comparison with PSM. The recently proposed parametric simplex-based solver PSM described in [26] can solve the ℓ_1-DS problem and is a state-of-the-art solver for ℓ_1-DS. In the next set of tests, we compare the following approaches.

1. PSM, as implemented in the companion R package `fastclime`, for a path of 50 λ values logarithmically spaced between $\lambda_{\min} = 2\|X^T e^0\|_\infty$ and $\lambda_{\max} = \|X^T y\|_\infty$.
2. Gurobi (dual simplex method) applied to the full LP model (9) for $\lambda = \lambda_{\min}$. We denote these results by "Full LP (Single)."
3. DantzigLP applied to (9) for the same 50 λ values as in the PSM tests. We denote these results by "DantzigLP (Path)."
4. DantzigLP applied to (9) for $\lambda = \lambda_{\min}$. We denote these results by "DantzigLP (Single)."

Note that PSM always computes a full path of solutions via homotopy, even if the solution is required at just one value of λ. The times shown for our DantzigLP methods include the times required to compute a Lasso path, usually between 0.1–1 seconds[9].

Table 1 shows that when computing a path of 50 λ-values, PSM is usually outperformed by DantzigLP (Path). In computing a solution to ℓ_1-DS at a single λ, PSM is seen to be outperformed by solving the full LP model with Gurobi (denoted by "Full LP"), with DantzigLP (Path) still faster.

We observe that PSM works well on instances with small p (in the hundreds), but its performance deteriorates with increasing p values. PSM computes the whole matrix $X^T X$, leading to large memory requirements by comparison with "Full LP" (which in our formulation does not require

[9]Solving the Lasso for $n = 200, p = 5000$ is the fastest with $n = 1000, p = 10^4$ being the slowest.

n	p	DantzigLP (Path)	DantzigLP (Single)	Full LP (Single)	PSM
200	5000	0.81	0.12	5.1	12.1
200	10000	1.1	0.14	10.3	49.0
500	5000	4.5	1.1	13.9	16.0
500	10000	5.9	1.3	28.9	86.5
1000	5000	24.8	7.9	31.9	22.3
1000	10000	27.2	7.6	65.1	92.3

Table 1: Runtime comparison for ℓ_1-DS (synthetic instances) with $\rho = 0$ and $\pi = 0$. For each n and p, we show the mean runtime (in seconds) of 20 problem instances. DantzigLP (with column and constraint generation) is usually faster than Gurobi's Full LP simplex solver (without column and constraint generation) and also faster than the state-of-the-art simplex-based homotopy solver PSM.

π	DantzigLP (Path)	DantzigLP (Single)	Full LP (Single)	PSM
0	8.2	2.4	16.7	57.3
0.4	6.1	2.0	10.2	55.4
0.8	2.4	0.86	3.5	57.0
0.95	0.79	0.27	1.0	56.3

ρ	DantzigLP (Path)	DantzigLP (Single)	Full LP (Single)	PSM
0	8.2	2.4	16.7	57.3
0.4	3.2	0.57	12.7	15.4
0.8	1.1	0.08	11.7	10.2

Table 2: Runtimes (in seconds, averaged over 20 replications) for solving ℓ_1-DS (synthetic instances) with $n = 500, p = 5000$ for varying sparsity and correlations in X. [Left] We vary sparsity in X, with $\rho = 0$ and $\pi \in \{0, 0.4, 0.8, 0.95\}$ (larger π values correspond to more zeroes in X). PSM has similar runtimes across all sparsity levels, whereas DantzigLP and Gurobi (Full LP) both see substantial reductions in runtime for sparse instances. [Right] We vary correlations in columns of X: X is dense with $\pi = 0$ and $\rho \in \{0, 0.4, 0.8\}$. Runtimes of all algorithms diminish with increasing correlations, with DantzigLP again fastest.

computation of $X^T X$) and also DantzigLP. Of all the methods, DantzigLP has the lowest memory requirements, as it generates new columns and constraints only as necessary.

Varying sparsity and correlation in X. We next explore sensitivity of runtimes of the various approaches to sparsity in X and correlations between columns of X, which are captured by the parameters π and ρ, respectively. Table 2 (Left) shows that the DantzigLP and Full LP approaches exploit sparsity well (runtimes decrease with increasing sparsity), while PSM does not benefit from sparsity, possibly becase the product $X^T X$ (which it computes) remains dense. For the case of dense X, Table 2 (Right) shows that increasing correlations between the columns lead to improved runtimes for all algorithms. (The solutions β tend to be sparser in these cases.) DantzigLP remains the clear winner.

Dependence on sparsity in solution β. In the experiments above, we considered a sequence of λ-values in the range $[\lambda_{\min}, \lambda_{\max}]$ with $\lambda_{\min} = 2\|X^T e^0\|_\infty$. It is well known that smaller values of λ correspond to denser solutions in β. To understand better the dependence of runtime of the various algorithms on sparsity of β, we tried the values $\lambda = \tau\|X^T e^0\|_\infty$, where $\tau \in \{0.1, 0.4, 0.7, 1\}$, showing the results in Table 3. Runtimes for DantzigLP increase with density in β, as expected, mostly because the Lasso solution has a very different support from the solution of ℓ_1-DS, requiring

DantzigLP to generate many more columns than it would for larger λ values. For smaller values of p (results not shown), DantzigLP can be even slower than the vanilla Gurobi implementation for the Full LP, due to the overhead of column generation, although memory consumption is still much smaller ($O(n^2)$ as opposed to $O(np)$). Computation time for PSM also increases as λ decreases, though not as much as DantzigLP in relative terms. Still, DantzigLP performance remains superior for most interesting values of λ.

τ	Avg. L_0	DantzigLP (Path)	DantzigLP (Single)	Full LP (Single)	PSM
0.1	453.	209.	96.9	77.7	334.
0.4	320.	79.2	30.7	74.5	368.
0.7	234.	39.6	15.2	55.6	351.
1.0	182.	13.6	4.7	24.1	125.

Table 3: Runtimes (in secs, averaged over 20 replications) for solving ℓ_1-DS (synthetic instances) with $n = 500, p = 5000$, $\rho = 0$, $\pi = 0$ for $\lambda_\tau = \tau\|X^T e^0\|_\infty$, where $\tau \in \{0.1, 0.4, 0.7, 1.0\}$. DantzigLP (Path) solves for a path of 50 λ values equally spaced between $\lambda_{\max}$ and λ_τ; Full LP and DantzigLP (Single) solve only for λ_τ; PSM solves for the whole path of values from $\lambda_{\max}$ to λ_τ. The column labelled "Avg. L_0" shows average support size of the optimal solution to ℓ_1-DS at λ_τ. As τ decreases, DantzigLP takes longer to solve the problem, although it still outperforms PSM.

Importance of the components of DantzigLP. The DantzigLP framework consists of several components: initialization provided by the Lasso, column generation and constraint generation, and the simplex LP solver. To understand the roles played by these components, we compare several variants in which some or other of them are omitted. Figure 2 compares the following five variants of DantzigLP.

(i) *DantzigLP*: The complete framework described above.

(ii) *Random Init.*: Rather than using a Lasso initialization, we initialize I, J to a random subset of $[p]$ of size $\|\beta^0\|$.

(iii) *Constraint Gen.*: Obtain I from Lasso initialization, but $J = [p]$. That is, only constraint generation is enabled.

(iv) *Column Gen.*: Here, J is obtained from Lasso initialization, with $I = [p]$. That is, only column generation is enabled.

(v) *Full LP*: The full LP model solved with Gurobi, without column or constraint generation.

The boxplots of Figure 2 show the runtime distribution over 20 randomly generated ℓ_1-DS instances with $n = 1,000$, $p = 10,000$, $\rho = 0$ and $\pi = 0$, solved for $\lambda = 2\|X^T e^0\|_\infty$. The figure highlights the importance of all elements of Dantzig LP. We note that column generation alone provides no performance gains over the Full LP approach, due to the overhead of generating new columns (and restarting our simplex-based LP solvers). However, we see improvements when constraint generation is also introduced.

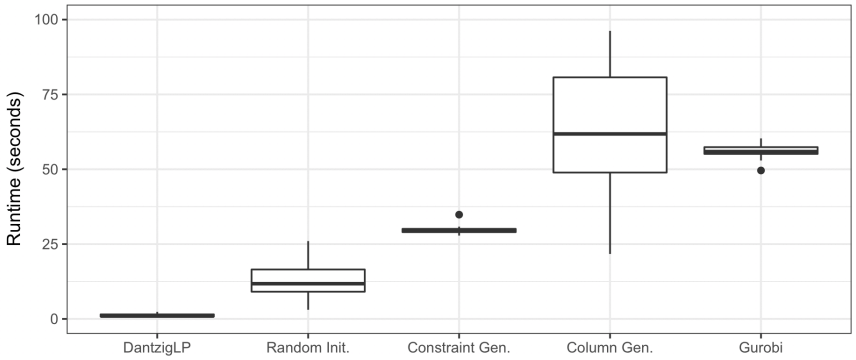

Figure 2: Comparison of 5 variants of DantzigLP ($n = 1,000$, $p = 10,000$), showing the importance of all components of the approach: Lasso initialization, column generation, constraint generation, and the simplex solver.

3.1.2 Experiments on real datasets

We demonstrate the performance of DantzigLP on real-datasets with $p \approx 10^6$ and $n \approx 10^3$. Due to memory constraints (our maximum was 64GB), only DantzigLP could solve these problems among our tested algorithms. We consider a path of 100 λ values log-spaced in the interval $[\lambda_{\max}, 10^{-3}\lambda_{\max}]$. Results are displayed in Table 4.

The "AmazonLarge" dataset from [19] has as its goal the prediction of helpfulness scores for product reviews based on data from Amazon's Grocery and Gourmet Food dataset. The "Boston1M" dataset consists of 104 covariates obtained from polynomial expansions of features in the Boston House Prices dataset ([18]), augmented with 1000 random permutations of each column.

Dataset	$\max \|\|\beta\|\|_0$	n	p	DantzigLP	Lasso
AmazonLarge	178	2,500	174,755	248.	70.7
Boston1M	56	200	1,000,103	3,702.	44.3

Table 4: DantzigLP (and associated Lasso) runtimes in seconds on real datasets. We solve for a path of 100 λ values. The column "$\max \|\beta\|_0$" indicates the size of the support of the densest solution obtained. The DantzigLP runtime includes the runtime of the Lasso initialization step.

3.2 Results for Basis Pursuit

We present some numerical results illustrating performance of the DantzigLP procedure for the basis pursuit problem (4). The matrix X is generated as in Section 3.1.1, with $\rho = 0$. We set $y = X\beta^0$, where β^0 is $0.2n$-sparse with nonzero elements chosen i.i.d. from $N(0, 1)$. We compare DantzigLP against two other algorithms: the ADMM approach of [7] as implemented in the ADMM R package, and solving the full LP model (`BP-Full`). In this example, we set a memory cap of 16GB of RAM for all experiments.

Table 5 shows runtimes in seconds for a number of instances. We see that DantzigLP performs much better than competing algorithms in terms of runtime, particularly when $p \gg n$. This also

n	p	DantzigLP	Lasso	Full LP (Gurobi)	ADMM	ADMM (% Converged)
200	10^3	1.9	0.38	1.3	> 1.1	90
200	10^5	8.8	7.8	NA	NA	NA
200	4×10^5	31.3	28.8	NA	NA	NA
500	10^3	16.8	1.4	4.3	6.3	100
500	10^5	25.3	19.1	NA	NA	NA
500	4×10^5	72.7	62.7	NA	NA	NA
1000	10^3	27.1	3.0	10.9	26.5	100
1000	10^5	88.7	42.1	NA	NA	NA
1000	4×10^5	209.	116.	NA	NA	NA

Table 5: Mean runtimes in seconds for simulated basis pursuit instances (20 instances per setting of n and p), comparing DantzigLP, the Lasso initialization step, the Full LP implementation in Gurobi, and ADMM. The "NA" values indicate cases where the algorithm was terminated due to memory constraints, with a maximum memory allocation of 16GB. "ADMM (% Converged)" indicates the percentage of instances in which ADMM was able to converge. The "ADMM" column reports the minimum of time to termination and time to convergence, with a ">" symbol indicating cases where fewer than 100% of instances converged.

comes with large savings in memory; the other algorithms were unable to solve problems of size $p \geq 10^5$ due to violation of the 16GB memory bound. DantzigLP can solve problems an order of magnitude larger than this. For instances with smaller p, the overhead of generating columns can cause DantzigLP to underperform the baselines in certain cases. The DantzigLP runtimes in Table 5 include the runtime of the Lasso initialization step, which is the main performance bottleneck; the Lasso accounts for 80% of the runtime for these instances.

While DantzigLP can obtain solutions of high accuracy, ADMM often has difficulty in doing so. The ADMM column in Table 5 reports the minimum of time to converge to within 10^{-4} of the true objective and time to complete 10000 ADMM iterations (after which we terminate the algorithm). As the "ADMM (% Converged)" column shows, for larger problem sizes none of the instances converge to that tolerance within the allotted iteration bound.

3.3 Computational experience with Fused Dantzig Selector

Signal estimation. We first consider the Fused Dantzig Selector with $X = \mathbb{I}$, that is, the signal estimation case of Section 2.4.1. We generate a piecewise constant signal, with discontinuities / knots chosen at random from $[n]$. At each knot, the jump is chosen from $N(0, 1)$. We add noise with SNR=10 to the signal. We solve (17) at a single value $\lambda = \|H_B^T(y - H_A\alpha_A^0 - H_B\alpha_B^0)\|_\infty$, where (α_A^0, α_B^0) corresponds to the true signal.

In Table 6, we compare our DantzigLP framework[10] to directly solving the full version (17). Gurobi's dual simplex solver is used in both cases. The formulation (17) allows solution of problems several orders of magnitude larger than the Dantzig Selector problem (9), when column/constraint

[10]We solve each instance to within a tolerance of $\epsilon = 10^{-4}$ and add up to 40 columns (constraints) per iteration of column (constraint) generation.

n	# knots	DantzigLP	Full LP
5×10^4	100	3.5	3.3
	200	3.5	3.2
	1,000	4.2	3.1
10^5	100	5.7	7.9
	200	6.2	8.0
	1,000	8.2	7.9

n	# knots	DantzigLP	Full LP
2×10^5	100	10.8	24.0
	200	11.9	24.1
	1,000	16.4	23.6
5×10^5	100	48.7	120.3
	200	54.1	121.4
	1,000	80.9	120.6

Table 6: [Signal Estimation] Runtimes (seconds) for the Fused Dantzig Selector, comparing DantzigLP with solution of the Full LP formulation (17) using Gurobi. Because of memory requirements, it would not be possible directly solve (16) without using our proposed reformulation (17) for the instances considered here.

n	p	DantzigLP	Gurobi
500	5,000	33.3	82.8
500	10,000	117.3	341.7
1,000	5,000	67.0	221.3
1,000	10,000	207.7	925.2

Table 7: [Regression] Runtimes (in seconds, averaged over 20 replications) for the fused Dantzig Selector with $X \neq I$. We compare our DantzigLP framework with Full LP (Gurobi).

generation is not used. The DantzigLP framework improves modestly on this enhancement, by factors of 2-3 for problems with large n and a small number of knots. The runtime for the full LP formulation is insensitive to the number of knots, while the runtime of DantzigLP increases with the number of knots with a large number of knots. This is not surprising, as more knots implies a denser solution. Solving the Fused Lasso (required for initialization) takes less than one second across all instances.

Regression. We illustrate the performance of our DantzigLP framework for (7) for a general model matrix $X \neq \mathbb{I}$. Here, entries of X are drawn from a standard Gaussian ensemble: i.e., $X_{ij} \overset{\text{iid}}{\sim} N(0,1)$. The underlying β^0 is piecewise constant and is drawn as before with 20 knots, with jumps chosen i.i.d. from $N(0,1)$. We generate $y = X\beta^0 + e$, with $e_i \overset{\text{iid}}{\sim} N(0,\sigma^2)$ and σ chosen to produce an SNR of 10.

Table 7 shows that the Fused Dantzig Selector modestly outperforms solving the full LP with Gurobi. The runtime improvements are not as pronounced as in the case of ℓ_1-DS because the initial solution obtained from the Lasso problem (5) is not as accurate as for ℓ_1-DS. Thus, additional columns/constraints need to be generated to achieve optimality, leading to increased runtimes.

4 Acknowledgements

Rahul Mazumder acknowledges research support from the Office Naval Research ONR-N000141512342, ONR-N000141812298 (Young Investigator Award) and the National Science Foundation (NSF-IIS-1718258).

References

[1] M. S. Asif and J. Romberg. On the lasso and dantzig selector equivalence. In *2010 44th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 3 2010.

[2] M. S. Asif and J. Romberg. Dantzig selector homotopy with dynamic measurements. *Proc. SPIE 7246, Computational Imaging VII, 72460E*, 7246, 2009.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] S. R. Becker, E. J. Candès, and M. C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165, 7 2011.

[5] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena, 1997.

[6] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[8] C. Brauer, D. A. Lorenz, and A. M. Tillmann. A primal-dual homotopy algorithm for ℓ_1-minimization with ℓ_∞-constraints. *Computational Optimization and Applications*, 70(2):443–478, 2018.

[9] E. Candes and T. Tao. The dantzig selector: Statistical estimation when p is much larger than n. *Ann. Statist.*, 35(6):2313–2351, 12 2006.

[10] S. Chen and D. Donoho. Basis pursuit. In *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44 vol.1, 10 1994.

[11] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[12] A. Dedieu and R. Mazumder. Solving large-scale l1-regularized svms and cousins: the surprising effectiveness of column and constraint generation. *arXiv preprint arXiv:1901.01585*, 2019.

[13] D. L. Donoho, A. Maleki, and A. Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[15] B. Efron, T. Hastie, and R. Tibshirani. Discussion: The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2358–2364, 2007.

[16] L. R. Ford Jr and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958.

[17] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[18] D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81 – 102, 1978.

[19] H. Hazimeh and R. Mazumder. Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. *ArXiv e-prints*, March 2018.

[20] G. M. James, P. Radchenko, and J. Lv. Dasso: connections between the dantzig selector and lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(1): 127–142, 2009.

[21] N. A. Johnson. A dynamic programming algorithm for the fused lasso and l 0-segmentation. *Journal of Computational and Graphical Statistics*, 22(2):246–260, 2013.

[22] X. Li, T. Zhao, X. Yuan, and H. Liu. The flare package for high dimensional linear regression and precision matrix estimation in r. *Journal of Machine Learning Research*, 16:553–557, 2015.

[23] Z. Lu, T. K. Pong, and Y. Zhang. An alternating direction method for finding dantzig selectors. *Computational Statistics & Data Analysis*, 56(12):4037 – 4046, 2012.

[24] E. Mammen and S. Geer. Locally adaptive regression splines. *The Annals of Statistics*, 25(1): 387–413, 1997.

[25] H. Pang, H. Liu, and R. Vanderbei. The fastclime package for linear programming and large-scale precision matrix estimation in r. *Journal of Machine Learning Research*, 15(1):489–493, January 2014.

[26] H. Pang, H. Liu, R. J. Vanderbei, and T. Zhao. Parametric simplex method for sparse learning. In *Advances in Neural Information Processing Systems*, pages 188–197, 2017.

[27] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259 – 268, 1992.

[28] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[29] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, pages 91–108, 2005.

[30] X. Wang and X. Yuan. The linearized alternating direction method of multipliers for dantzig selector. *SIAM Journal on Scientific Computing*, 34(5):A2792–A2811, 2012.

[31] S. J. Wright, R. D. Nowak, and M. A. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.