






MIP-BOOST: Efficient and Effective L_0 Feature Selection for Linear Regression

Ana Kenney , Francesca Chiaromonte & Giovanni Felici

To cite this article: Ana Kenney , Francesca Chiaromonte & Giovanni Felici (2020): MIP-BOOST: Efficient and Effective L_0 Feature Selection for Linear Regression, Journal of Computational and Graphical Statistics, DOI: [10.1080/10618600.2020.1845184](https://doi.org/10.1080/10618600.2020.1845184)

To link to this article: <https://doi.org/10.1080/10618600.2020.1845184>

 View supplementary material 

 Accepted author version posted online: 17 Nov 2020.

 Submit your article to this journal 

 Article views: 36

 View related articles 

 View Crossmark data 

有效率的

有效的

MIP-BOOST: Efficient and Effective L_0 Feature Selection for Linear Regression

Ana Kenney*

Dept. of Statistics, Penn State University. University Park PA, USA.

and

Francesca Chiaromonte

Dept. of Statistics, Penn State University. University Park PA, USA.

Inst. of Economics & EMbeDS, Sant'Anna School of Advanced Studies. Pisa, Italy.

and

Giovanni Felici

Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche. Rome, Italy.

*We thank Matthew Reimherr for useful discussions and comments. This work was partially funded by the NIH B2D2K training grant and the Huck Institutes of the Life Sciences of Penn State, and by NSF grant DMS-1407639. Computation was performed on the Roar Supercomputer at Penn State University.

Corresponding author Ana Kenney ajk5910@psu.edu

Abstract

Recent advances in mathematical programming have made Mixed Integer Optimization a competitive alternative to popular regularization methods for selecting features in regression problems. The approach exhibits unquestionable foundational appeal and versatility, but also poses important challenges. Here we propose MIP-BOOST, a revision of standard Mixed Integer Programming feature selection that reduces the computational burden of tuning the critical sparsity bound parameter and improves

performance in the presence of feature collinearity and of signals that vary in nature and strength. The final outcome is a more efficient and effective L_0 Feature Selection method for applications of realistic size and complexity, grounded on rigorous cross-validation tuning and exact optimization of the associated Mixed Integer Program. Computational viability and improved performance in realistic scenarios is achieved through three independent but synergistic proposals. Supplementary materials including additional results, pseudocode, and computer code are available online.

Keywords: Regression, feature selection, Mixed Integer Optimization, LASSO, cross-validation, whitening.

1 Introduction

Feature Selection methods are a key component of contemporary regression analyses, where they allow researchers to identify a subset of relevant or active predictors among an often very large number of available candidates.

LASSO-type methods ([Tibshirani \(1996\)](#), [Zou and Hastie \(2005\)](#), [Meier et al. \(2008\)](#)) select features solving a convex optimization problem that shrinks the L_1 norm of the coefficient estimates vector. Unlike prior best subset and sequential selection approaches (see [Hastie et al. \(2005\)](#) for an overview), these methods induce sparsity without a direct control on the number of features selected and gained enormous popularity due to the improvements they provided in terms of computational burden, accuracy, or both.

More recent proposals adopt Mixed Integer Programming (MIP) to regain direct control on the L_0 norm of the coefficient estimates vector, i.e. the size of the active set, while substantially reducing the computational burden on problems of realistic size through state-of-the-art optimization techniques ([Bertsimas et al. \(2016\)](#), [Bertsimas et al. \(2017\)](#)). These proposals engendered a lively debate ([Hastie et al. \(2017\)](#), [Mazumder et al. \(2017\)](#), [Bertsimas and Van Parys \(2017\)](#)), extensive comparisons with the LASSO, and the

introduction of *hybrids* – e.g., the relaxed LASSO ([Hastie et al. \(2017\)](#)), see also [Meinshausen \(2007\)](#)) and the penalized MIP ([Mazumder et al. \(2017\)](#)).

Notwithstanding their reduced computational cost, existing implementations of the MIP approach remain rather burdensome – and in fact often non-viable for problems comprising thousands or more features, or when performing a rigorous tuning of the sparsity bound (i.e. of the number of features to be retained). Moreover, whether MIP or LASSO approaches produce better prediction, coefficients estimation and active set identification depends on problem complexity. Signal strengths and heterogeneity across features, the degree of true underlying sparsity, and linear associations among features, can all have important effects on the relative performance of the two approaches.

According to theory, LASSO can separate active and inactive features only under the irrepresentable condition ([Zhao and Yu \(2006\)](#)), which bounds the correlations between them. This may require near-orthogonality between the two sets of features, which is rather unrealistic in many applications. In this regard, [Zhao and Yu \(2006\)](#) argued that for universal consistency of the LASSO (i.e. selection of the true active set) the amount of shrinkage needs to be reduced – leading to a regularization similar to that of the L_0 penalty. To date, the comparative performance of MIP and LASSO approaches in the presence of collinearity and signals that vary in nature and strength is only partially understood.

The lack of conclusive comparisons may be due to biases arising from computational limitations in tuning. Proper choices of the tuning parameter are critical for the performance of both LASSO and MIP. In LASSO this parameter is a non-negative scalar weighing the L_1 penalty. In MIP it is the sparsity bound, i.e. the integer size of the active set represented by the right hand side of a linear integer constraints. This intrinsically harder and more expensive tuning problem is handled with time caps (e.g., the three minute limit in [Hastie et al. \(2017\)](#)),

using a simple training/test regime instead of ν -fold cross-validation, and/or relying on validation results obtained in simplistic scenarios, where the optimal tuning parameter is easy to spot.

Best tuning practices are based on estimating out-of-sample performance, e.g., by cross-validation. But for a problem comprising p features, tuning the MIP by standard ν -fold cross-validation requires solving $\nu \times p$ instances of the optimization problem (or $\nu \times n$ when $p > n$) – with each instance carrying a still substantial computational cost.

Though theory-based metrics such as AIC or BIC (Akaike and Bayes Information Criteria) can be used to avoid cross-validation, they rely on distributional assumptions and asymptotic properties that may be unrealistic. Interestingly, [Miyashiro and Takano \(2015\)](#) treat information criteria as the objective in a reformulated MIP which must be solved only once, with the sparsity bound value internally determined – but this was found to be in fact slower than solving the original MIP over all possible values of the bound. Proposals also exist to improve the computational efficiency of each individual MIP run, usually adopting sub-optimal solutions based on linear relaxations and heuristics which do not carry a provable quality guarantee ([Willis and von Stosch \(2017\)](#)). In a recent proposal, [Hazimeh and Mazumder \(2018\)](#) solve multiple local MIPs but can still verify optimality under large local neighborhoods given enough computing time.

We contribute to this debate proposing MIP-BOOST, an enhanced method for L_0 Feature Selection based on extensions that significantly improve computational efficiency as well as effectiveness of the MIP approach. These extensions leverage specific characteristics of the class of optimization techniques used for solving L_0 Feature Selection problems.

First, we propose a *novel bisection procedure* designed specifically for tuning the sparsity bound, which significantly reduces the needed number of evaluations. While this modified bisection can in principle be employed in other optimization

settings, we focus on documenting its remarkable impact in cutting the computational burden of the MIP approach.

Second, we propose a *novel cross-validation scheme* that exploits the structure of the MIP and of the simplex algorithm used for its solution to significantly reduce the computational effort of repeating calculations across folds. We also employ warm starts and surrogate lower bounds within a state-of-the-art MIP solver to further cut running times.

Third, we complement the MIP with *whitening* (Kessy et al. (2018)). This is a pre-processing step that, by handling feature collinearities, can both reduce computational burden and improve solution quality. Whitening can be applied prior to any feature selection technique but it benefits MIP more than it does other approaches.

We test MIP-BOOST on an extensive body of synthetic data and on a Diabetes data set already analyzed in Efron et al. (2004), demonstrating a reduction of several orders of magnitude in the computational burden of MIP and a performance on par with or better than that of LASSO-type methods. Our results, in conjunction with other recent literature, establish L_0 Feature Selection as a solid and viable alternative for a wide spectrum of large and complex regression problems.

The remainder of the article is organized as follows: Section 2 provides technical background, Section 3 introduces our proposals, Section 4 presents our simulation study and two real data applications – one related to diabetes and another to body fat, and Section 5 contains final remarks and perspectives.

2 Background

Consider a linear regression of the form $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\mathbf{Y} \in \mathbb{R}^n$ is the vector of response values, $\mathbf{X} \in \mathbb{R}^{n \times p}$ the design matrix containing the values of p

predictors, $\beta \in \mathbb{R}^p$ the vector of regression coefficients, and $\epsilon \in \mathbb{R}^n$ the vector of errors. Throughout the article we assume that the data is centered and omit the intercept.

In the following, we describe the standard LASSO approach and set the stage for our proposals introducing Mixed Integer Programs (MIP) and their use to formulate the best subset selection problem. As will appear clearly, the key difference between the MIP and LASSO formulations is the choice of penalization: While MIP uses the L_0 norm, bounding the size of the subset (the number of selected features), LASSO uses the L_1 norm, bounding the size of the coefficient vector.

2.1 Feature Selection with the LASSO

LASSO ([Tibshirani \(1996\)](#)) is a regularization method commonly used for feature selection due to its computational speed and ability to induce shrinkage and fit parameters simultaneously. It solves the constrained optimization

$$\min_{\beta} \frac{1}{n} \|Y - X\beta\|_2^2 \quad (1)$$

$$\|\beta\|_1 \leq \lambda.$$

where the L_1 norm is used to bound the size of coefficient vector by the scalar $\lambda \geq 0$. LASSO does not explicitly solve best subset selection nor directly control sparsity; it leverages a simple optimization problem comprising continuous variables and convex constraints.

Many fast algorithms have been proposed for solving (1); the most commonly used is a variation of coordinate descent from [Friedman et al. \(2010\)](#) (a very efficient implementation is provided in the `glmnet` R-package). Solution quality depends on the choice of the tuning parameter λ , which is often selected minimizing cross-validation error. Another choice, which induces more shrinkage,

is the largest λ with cross-validation error within one standard deviation from the minimum (*parsimonious* LASSO; Hastie et al. (2005)).

Overall, notwithstanding the efficient tuning made possible by computational speed, LASSO is known for producing solutions with a high number of false positives. In fact, Tibshirani (2011) recommended it as a *screen* (instead of a selection procedure), because under certain conditions it achieves the sure screening property (Fan and Lv (2008)).

2.2 Best Subset Selection as a Mixed Integer Quadratic Program

For any given integer $k \leq p$, best subset selection minimizes the regression error using at most k features. This problem has the same structure of (1), namely

$$\min_{\beta} \frac{1}{n} \|Y - \mathbf{X}\beta\|_2^2 \quad (2)$$

$$\|\beta\|_0 \leq k$$

but it restricts the subset size directly through the L_0 constraint

$\|\beta\|_0 = \sum_{i=1}^p I(\beta_i \neq 0) \leq k$. This constraint naturally leads to a Mixed Integer Program

(MIP). In MIPs, an objective function is optimized under a collection of constraints – including constraints that restrict some of the variables to integer values. A well studied class of MIPs are Mixed Integer Quadratic Programs (MIQP), where a quadratic objective function is optimized under linear constraints, lower and upper bounds on the feasible values of the solution, and integrality constraints.

Bertsimas et al. (2016) adopted a reformulation of (2) based on a MIQP

introducing a binary vector $z \in \{0,1\}^p$ that indicates whether each feature is selected or not ($z_j = 1$ or 0):

$$\begin{aligned}
& \min_{\beta, z} \frac{1}{n} \|Y - X\beta\|_2^2 \\
& -\mathcal{M}z_j \leq \beta_j \leq \mathcal{M}z_j \quad j=1, \dots, p \quad ; \quad \sum_{j=1}^p z_j \leq k \quad (3) \\
& \beta_j \in \mathbb{R} \quad , \quad z_j \in \{0,1\} \quad j=1, \dots, p.
\end{aligned}$$

The integrality constraints in (3) (and all MIQPs in general) pose serious computational challenges that have been tackled with dramatic improvements in optimization solvers (see [Bertsimas et al. \(2016\)](#) for an overview). At present, the most effective way to solve a MIQP is to relax the integrality constraints and then combine implicit enumeration methods, such as Branch & Bound, with the addition of constraints that approximate integrality in the relaxed problem, such as Branch & Cut (see [Schrijver \(1986\)](#)) for an introduction). The overarching goal of recent contributions and of our own work is indeed to harness this extremely powerful machinery for the purpose of feature selection.

The parameters \mathcal{M} and k also play very important roles. Constraining each $\beta_j \neq 0$ to be within $[-\mathcal{M}; \mathcal{M}]$ is referred to as the “bigM” trick, and the proper choice of \mathcal{M} is an art. A sufficiently large \mathcal{M} guarantees that solutions to (3) are also solutions to (2), but an excessively large \mathcal{M} can affect solvability – mainly for numerical reasons. Using separate bounds $\mathcal{M} = (\mathcal{M}_1, \dots, \mathcal{M}_p)'$ for each coefficient increases flexibility in the estimation of β .

[Bertsimas et al. \(2016\)](#) provide a few options for selecting \mathcal{M} , as well as a clever formulation that does not require its *a priori* choice. In our implementation we set separate and fairly wide bounds utilizing Ordinary Least Squares (OLS) estimates.

The parameter k has a less technical and rather crucial role for guaranteeing accurate feature selection. It bounds the size of the subset in (3), directly controlling sparsity. If k is too small, we miss important signals; if it is too large, we overfit the regression. For large p , choosing k by standard tuning methods

such as 10-fold cross-validation across its entire range $\{1, \dots, p\}$ is computationally very demanding, if not prohibitive – even with state-of-the-art solvers. Assuming that the regression is very sparse, this issue can be mitigated by restricting the range of k at the outset (e.g., exploring only $k \in \{0, \dots, 10\}$ even though p is in the range of thousands). In many applications though this may not be appropriate.

3 Our Proposals

Here we present MIP-BOOST, which provides an efficient and effective way to deploy the MIP machinery for L_0 feature selection. It is based on three extensions to the classic MIP that together contribute to large improvements in execution time and solution quality.

The first is a modified bisection procedure that limits the grid search required for tuning; the second is an integrated scheme that reduces the computational burden of individual MIP solutions in a ν -fold cross-validation; and the third is a whitening pre-processing step that handles feature collinearities. Each targets a different aspect of the computational demands of feature selection.

3.1 Bisection with Feelers

Let k_0 be the true and unknown number of active features. Fixing an appropriate bound, i.e. a good estimate of k_0 based on the data, is critical for the effectiveness of feature selection. This can be done by evaluating out-of-sample regression performance via ν -fold cross-validation, and comparing it across all k values from 1 to p (or n when $p > n$). For example, the black curve in Figure 1(a) shows 10-fold cross-validation Mean Squared Errors (CVMSE) obtained solving MIPs at $k = 1, \dots, p$ on simulated data with $n = 100$, $p = 50$ and $k_0 = 15$ (see Section 4 for details). Ideally, the CVMSE curve would decrease (as one adds more active features) and then steadily increase (as one overfits adding inactive features), identifying \hat{k}_0 as a *sweet spot*. But in many practical applications,

especially when features are highly correlated and/or signals are weak, the CVMSE curve can have an *elbow* shape as in Figure 1(a) (or yet more complicated, fluctuating behaviors as in Section 3, Figure 3 of the Supplement). Note that non-ideal behaviors of the CVMSE curve can be starker when plotting against discrete sparsity bound values, as opposed to plotting against a fine grid of λ values when tuning the LASSO. Our objective is thus to reliably identify a global minimum (a sweet spot) or the smallest, most parsimonious among a stretch of approximately equivalent bound values (an elbow) in a non-ideal CVMSE curve. And the problem is the computational burden of cross-validating across the whole range of k : if p is large, this may be unviable because we need to solve as many as $v \times p$ MIPs (or $v \times n$ for $p > n$). For very sparse regressions, say $k_0 \leq 10$, it may still be feasible to solve v MIPs for $k = 1, 2, \dots$ until it becomes clear that we are past a sweet spot or an elbow in the CVMSE curve. However, if the regression is not very sparse, or if it is only *proportionally* sparse but p is very large, this is still unviable; e.g., if $p = 5000$ and $k_0 = 10\% \times p = 500$, we need to cross-validate more than 500 values of k (solve more than $v \times 500$ MIPs) to identify a good estimate of the bound.

If we could count on the CVMSE curve having a single global minimum and a clear quasi-convex shape, we could reduce computational burden with standard 1D search methods that do not require first or second order information. These search the whole range $\{1, \dots, p\}$ (or $\{1, \dots, n\}$ if $p > n$) of k but evaluate only a small portion of its values. However, they can be rather ineffective for identifying the elbow of a curve shaped as in Figure 1(a) since gains in solution quality, no matter how negligible, push the search towards adding inactive variables. Thus, we propose a procedure that modifies a standard bisection search (Cormen et al. (2009)) as to avoid chasing minor gains. We call this **Bisection with Feelers (BF)**. BF can effectively hone into an elbow, as well as a sweet spot, and it does so efficiently; as a bisection, it limits the number of MIPs to be solved to around $v \times \log_2(p)$ (or $v \times \log_2(n)$ when $p > n$).

The overall idea of BF is to bisect the search interval until convergence, choosing each subsequent interval based on which continues to provide a substantial decrease in CVMSE. We begin by initializing $a = a_0$ and $c = c_0$, the two end points of the search interval and splitting at the midpoint b to get $[a, b]$ and $[b, c]$. We can naively set $a_0 = 1$ and $c_0 = p$ (or $c_0 = n$ when $p > n$). A better option is to utilize an inexpensive method to set a more informed upper bound. For our experiments in Section 4 we used the LASSO which is known to provide a good upper bound, retaining true positives (albeit often along with false ones; see Section 2.1). Other advanced heuristics for solving best subset selection, such as that proposed in [Hazimeh and Mazumder \(2018\)](#), can be applied instead of LASSO to provide a range of values, though LASSO is expected to be the most generous, and thus conservative, as it favors larger solution sets.

Our criterion to decide which of the two intervals to search next is a standardized marginal change in cross-validation error (a slope). Let M_k be the set of features selected by the MIP with a generic k , and $f(k)$ the corresponding cross-validation error; the improvement that occurs between, say, x and y is

$$\Delta f(x, y) = -\frac{1}{f(x)} \frac{f(y) - f(x)}{(y - x)},$$

which represents the percentage change per

feature added in $[x, y]$ (the negative sign makes positive and higher values of $\Delta f(x, y)$ preferable). We solve for M_a , M_b , and M_c and evaluate the criterion for $[a, b]$ and $[b, c]$. With a given improvement threshold $\delta > 0$, $\Delta f(b, c) \leq \delta$ suggests that b is already past the elbow of the curve, so we search $[a, b]$. Conversely, if $\Delta f(b, c) > \delta$, we search $[b, c]$. To fix a reasonable δ it helps to think of it as a baseline percent improvement to the starting point of each interval; in our experiments in Section 4 we use 1 or 5% – meaning a gain of less than 1 or 5% is considered negligible.

In Section 6 of the Supplement we provide a proof of convergence for this procedure under some regularity assumptions. Notably, the proof covers the case of a CVMSE curve with multiple elbows – where convergence occurs to the one with the lowest CVMSE. This is also the case when there are multiple local

minima, unless there are substantial increases in CVMSE between them. In such scenarios, BF will initially tend to select the minimum with a sparser solution to avoid overfitting – but the feelers portion can in fact correct for this, provided it operates with a wide enough radius (see below).

The process continues until the bisection converges and outputs a tentative estimate \hat{k}_0 . This though is not necessarily our end result: again to help tackle irregularities in the CVMSE curve, we next send feelers around \hat{k}_0 , computing $\Delta f(\hat{k}_0 - l_f, \hat{k}_0)$ and $\Delta f(\hat{k}_0, \hat{k}_0 + l_f)$, where l_f is the feelers' radius (in our experiments in Section 4 we fix $l_f = 1$ and check only immediately to the left and right of \hat{k}_0). If both are $\leq \delta$, suggesting that \hat{k}_0 does not provide a sufficient improvement relative to its surroundings, we restart the search setting $a = 1$ and $c = \hat{k}_0 - l_f$. But if $\Delta f(\hat{k}_0, \hat{k}_0 + l_f) > \delta$, suggesting that improvement is possible to the right of \hat{k}_0 , we restart the search setting $a = \hat{k}_0 + l_f$ and leaving c to its initial value.

Due to the feelers portion, the number of evaluations in BF may exceed $\log_2(c_0)$, depending on how many restarts we allow and how far past the elbow we are each time. In our experiments we typically restart at most once; nevertheless, we set a bound *itermax* on the number of iterations to avoid excessive computing (Algorithm 1 in Section 1 of the Supplement provides pseudocode for BF).

3.2 Integrated Cross-Validation

The red curve in Figure 1(a) shows average computing times (over cross-validation folds) across k values. Some are much higher than others. As expected, very small or very large k 's result in faster convergence rates for the optimization solver (see Section 4.1). Convergence slows down as we move inward and speeds up again around the true k_0 – which is of course unknown in applications. What if, in addition to reducing the number of MIP runs required to tune k (Section 3.1), we could make the most expensive ones cheaper?

For any given fold, solving the MIP amounts to solving a relaxation of the complete problem where constraints on withheld observations are ignored (i.e. relaxed). Thus, when the focus shifts from one fold to another, it may be very easy to recover feasibility of the relaxed solution using a dual approach – a step typically integrated in MIP solvers. We call this **Integrated Cross-Validation** (ICV). For each fold \mathcal{F} , we modify (3) relaxing constraints on withheld observations through a parameter $\tilde{\mathcal{M}}$ whose use is similar to that of \mathcal{M} in Section 2. In symbols, we set $m_i = \tilde{\mathcal{M}} \times I(i \in \mathcal{F}), i = 1, \dots, n$ and consider

$$\begin{aligned}
& \min_{\beta, z, e} \sum_{i=1}^n e_i^2 \\
& -e_i - m_i \leq Y_i - \sum_{j=1}^p \beta_j X_{ij} \leq e_i + m_i \quad i = 1, \dots, n \\
& -\mathcal{M}z_j \leq \beta_j \leq \mathcal{M}z_j \quad j = 1, \dots, p \quad , \quad \sum_{j=1}^p z_j \leq k \\
& \beta_j \in \mathbb{R} \quad , \quad z_j \in \{0, 1\} \quad j = 1, \dots, p \quad , \quad m_i = \tilde{\mathcal{M}} \times I(i \in \mathcal{F}).
\end{aligned} \tag{4}$$

In this formulation subsequent folds are bound to be computationally cheaper than the first, that serves as a starting point, or warm start, for the others. However, for very expensive runs even the first fold can be computationally taxing. Following well established practice, we further reduce computing time by generating good starting points for the Branch & Cut procedure. Various approaches have been proposed to produce high quality warm starts (e.g., [Beck and Eldar \(2013\)](#), [Patrascu and Necoara \(2015\)](#), [Hazimeh and Mazumder \(2018\)](#)) – any of which could be used here – but we opt for a simple and very inexpensive Forward Selection (FS) ([Hastie et al. \(2005\)](#)). FS can be solved for exactly k features, producing selection indicators z_{FS} and coefficient estimates β_{FS} . Running MIP with these as warm starts substantially cuts computational burden. An example of the resulting gains is shown in Figure 1(b), where Integrated cross-validation is consistently faster than standard cross-validation across folds (the other two approaches shown in the Figure, Integrated+ and Integrated++ are discussed next).

From a practical standpoint, when solving for a generic k , termination is triggered either by reaching a sufficiently small integrality gap between lower and upper bound (Schrijver (1986), Chen et al. (2011)), or by surpassing a maximum computing time (both rules can be customized in state-of-the-art solvers). In our implementation, as customary, we set a gap threshold ϵ_G as well as a *maxtime*. For the most expensive runs, *maxtime* may be reached before the current gap is at or below ϵ_G . Notably, the solver often reaches an optimal or near optimal point earlier than the total solution time may indicate (Bertsimas et al. (2016), Hastie et al. (2017)) due to a weak lower bound that is slow to improve. On the other hand, Hastie et al. (2017) noted that the setting of a bound on computation time (3 minutes, in the specific case) may have contributed to the poorer performance of MIP compared to LASSO approaches in their more complex experiments. To mitigate these issues we resort again to FS, this time to create a surrogate lower bound which allows us to lessen the dependence on *maxtime*.

We rerun FS, now for $k + 1$ features, and take the resulting in-sample MSE \mathcal{E}_{FS^+} as an *error bound*. We continue solving our MIP and terminate when the surrogate gap between the current objective and \mathcal{E}_{FS^+} is $\leq \epsilon_{FS^+}$ (surrogate gap threshold). This way, instead of relying on a potentially overly stringent *maxtime* parameter (or setting an arbitrarily larger value to be more conservative), we continue the Branch & Cut procedure until it generates a solution that is more parsimonious than that of FS (one fewer feature) and has comparable error (a gap matching ϵ_{FS^+}).

To be clear, the goal of MIP-BOOST is to reduce the computational burden while solving each MIP to optimality. A *maxtime* parameter is still needed in practice (see also Bertsimas et al. (2016) and Hastie et al. (2017)), but the surrogate lower bound allows us to enforce some solution quality guarantee should the *maxtime* be reached without verified convergence. For instance, the Integrated+ and Integrated++ computing times shown across folds in Figure 1(b) refer to triggering the surrogate lower bound after a *maxtime* of 3 and 0 minutes,

respectively. These are lower than the times of Integrated and standard cross-validation ran until convergence. Clearly, Integrated++ will be the fastest in practice, but it may worsen solution quality since it completely relies on the surrogate lower bound. We consider this approach as a way to measure the accuracy of the bounds in Section 4.1.2, but we do not recommend its use when optimality is the goal; a good heuristic may be yet faster and possibly better in terms of solution quality. Conversely, Integrated+ may result in slightly higher computing times than the *maxtime* set, but can only improve solution quality. Our numerical experiments show that, when used in combination with BF, the “+” version of ICV is still much more efficient than standard cross-validation – with negligible additional cost compared to the “++” version.

As with the warm start, procedures other than FS can be used to create the surrogate lower bound. We compared FS with the advanced heuristic proposed in [Hazimeh and Mazumder \(2018\)](#) and implemented in the `L0Learn` package on some of our numerical experiments (see Section 4.6 Figure 26 of the Supplement). In the settings we tested, we observed no substantial change in statistical quality or computational speed – however, we expect advanced heuristics to improve quality in general, since they are likely to produce more accurate bounds than FS.

We conclude noting that, to account for cases where the surrogate lower bound is still too hard to match within a realistic time frame, we set a second, much more generous *totaltime* parameter (Algorithm 2 in Section 1 of the Supplement provides pseudocode).

3.3 Whitening

With strong collinearities and/or weak signals, an accurate estimate of the size of the active set does not guarantee that solving for $M_{\hat{k}_o}$ will select the relevant features. Figure 2 and Figure 2 in Section 3 of the Supplement show results obtained solving with the true $k_o = 10$ on data simulated for a range of Signal-to-

Noise ratios (SNR; from 0.1 to 10) and highly collinear features. We considered an autoregressive correlation structure with $\text{corr}(\mathbf{X}_\ell, \mathbf{X}_j) = \alpha^{|\ell-j|}$, and a block structure with $\text{corr}(\mathbf{X}_\ell, \mathbf{X}_j) = \rho$ within the active and inactive sets, and $\text{corr}(\mathbf{X}_\ell, \mathbf{X}_j) = \omega$ across the two sets (see Section 4). For $\alpha = 0.9$, or $\rho = 0.5$ and $\omega = 0.4$, MIP solutions lack sensitivity even with the true k_0 , especially at low SNRs. However, we find that a *whitening* data pre-processing procedure can greatly improve MIP recovery.

In full generality, whitening involves switching from the original \mathbf{X} ($n \times p$) to a new design matrix $\mathbf{Z} = \mathbf{X}\mathbf{W}$ ($n \times p$), where the whitening matrix \mathbf{W} ($p \times p$) is chosen so that the transformed features are uncorrelated. We use the sample covariance matrix and set $\mathbf{W} = \boldsymbol{\Sigma}^{-1/2}$, which is known as ZCA or ZCA-Mahalanobis whitening. ZCA minimizes the distance between original and whitened features (Kessy et al. (2018)), and creates a direct feature pairing considering distances between the j th feature in \mathbf{X} and \mathbf{Z} . MIP, or any other selection method, can then be applied to select whitened features in \mathbf{Z} and translate the selection back to the original features in \mathbf{X} ; for instance, if Z_1 and Z_3 are selected, this will translate in selecting X_1 and X_3 . Of course the efficacy of selecting original features based on this simple pairing depends on how small is the minimal distance, i.e. on how close \mathbf{Z} can be made to \mathbf{X} . Whitening approaches other than ZCA, capable of controlling and possibly further reducing this minimal distance, represent an interesting area of research beyond the scope of this article. We note though that, under the settings we considered in our numerical experiments, ZCA does produce a marked improvement – as shown in Figure 2, where it allows one to reach almost 100% true positives at SNRs ≥ 0.75 also under high collinearity.

Post selection, one can use any estimator on the feature subset; in our experiments we simply use the OLS (Algorithm 3 in Section 1 of the Supplement provides pseudocode for whitening).

ZCA whitening has a marked positive effect also on the CVMSE curve (see Supplement, Section 3, Figure 3); it produces a more distinct elbow and solutions reasonably close to the true $k_0 = 10$. Interestingly, LASSO procedures do not seem to benefit from whitening as much as MIP, as they are prone to specificity rather than sensitivity issues (see Sections 2.3 and 4). To implement whitening, we used the MLE of the covariance matrix in low-dimensional experiments (Figure 2, Section 3 of the Supplement), where it is reasonably accurate, and the true covariance matrix in high-dimensional experiments. Investigating the use of effective high-dimensional covariance estimators in whitening (see [Fan et al. \(2016\)](#) for an overview), likewise investigating whitening approaches more effective than ZCA, exceeds the scope of this article and is left for future work.

4 Numerical Studies

To assess statistical quality, we compare MIP-BOOST to LASSO procedures and FS. To assess computational efficiency, we compare the burden of 10-fold cross-validation with MIP-BOOST with that of a standard MIP implementation (LASSO procedures are not considered here as they are orders of magnitude faster).

4.1 Simulation Study

We consider a variety of scenarios defined in terms of problem size p , sparsity level k_0 (the true number of active features), signal patterns in β (the regression coefficients), SNR levels, and nature and strength of feature collinearities.

We draw the n rows of the data matrix \mathbf{X} independently from a p -variate Gaussian distribution $\mathbf{X} \sim N_p(\mathbf{0}, R)$ – without loss of generality, means are set to 0 and variances to $r_{j,j} = 1$. For the correlations, we consider two regimes. The first is an *autoregressive structure* with $r_{\ell,j} = \alpha^{|\ell-j|}$ for $\ell \neq j$. Here each feature is strongly associated with a few neighboring ones – and only weakly or almost not at all to features further away. If we list the active features as the first k_0 , followed by the $p - k_0$ inactive ones, potential confounding across the former and the

latter occurs for those within a certain radius of the k_0 boundary – depending on the size of $\alpha \in [0,1]$. The second regime is a *block structure* where all active features are pair-wise correlated at level $r_{\ell,j} = \rho, \ell, j \in A, \ell \neq j$, all inactive features are too, $r_{\ell,j} = \rho, \ell, j \in A^C, \ell \neq j$, and across the two sets one also has pair-wise correlations at level $r_{\ell,j} = \omega, \ell \in A, j \in A^C$. Here the sizes of $\rho \in [0,1]$ and $\omega \in [0,1]$ control, respectively, the degree of potential confounding among active features, and between active and inactive ones. Notably, each active feature has the same strength of linear association with *all* inactive features; sizable values of ω induce strong violations of the irrerepresentable condition (Zhao and Yu (2006)), which in turn induce a loss of selection consistency for LASSO-type methods. These structures proxy different types of real data settings and have different eigenvalue decay profiles (see Figure 1, Section 2 in the Supplement); with appropriately high parameter values, they can both represent strong collinearity.

The response vector is generated as $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N_n(0, \theta^2 \mathbf{I})$. Values of the scalar θ are used to create scenarios with varying SNRs, defined as $\text{SNR} = \frac{\text{var}(\boldsymbol{\beta}^T \mathbf{X})}{\theta^2}$. A summary of all parameter settings is given in Table 1. Realistic SNRs may vary depending on scientific fields and class of problems. In recent literature, the MIP approach was shown to perform well at very high SNRs (Bertsimas and Van Parys (2017)) and, conversely, to perform worse than LASSO procedures at very low SNRs (Hastie et al. (2017)). Both extremes may characterize a limited number of real applications; we explore a range of SNR values from 0.1 to 10, corresponding to R^2 values from $\approx 1\%$ to $\approx 90\%$ (results below are for $\text{SNR} = 0.1, 0.75, 1, 10$, full results are provided in the Supplement, Section 4).

4.1.1 Statistical Quality

Fixing the number of cross-validation folds to 10, we compare our MIP-BOOST against the *standard LASSO* with the minimizing or the parsimonious choice of λ (LMN and LSD, see Section 2.3); the *relaxed LASSO* of Hastie et al. (2017) with

the minimizing choice of λ (RL); and *standard forward selection* with the minimizing choice of subset size (FS). For LMN, LSD, RL and FS we use the R package `bestsubset` from [Hastie et al. \(2017\)](#). For MIP-BOOST we use Julia 0.6.1 to interface with the MIP optimization solver Gurobi 7.5.1. *maxtime* is set to the recommended 3 minutes for all scenarios, but *totaltime* is set to 10 and 20 minutes for scenarios with $p = 100$ and $p = 1000$, respectively¹. Optimality and surrogate gaps are set to $\epsilon_G = 0.05$ and $\epsilon_{FS+} = 0.05$ (see Section 3.1).

Concerning the *big M* parameters (see Section 2.3), we set $M_j = c \|\hat{\beta}_{j,OLS}\|$ where β_{OLS} is the vector of Ordinary Least Squares (OLS) estimates and we set the scaling constant to $c = 5$ (inflating by c does not give rise to solvability issues).

MIP-BOOST (Julia) and data generation (R) code is available as supplementary material.

Simulation scenarios are replicated 5 or 10 independent times. As performance metrics, we consider (i) average number of true and false positives (*sensitivity* and lack of *specificity*); (ii) average MSE computed on separately drawn validation data for each scenario and simulation (*prediction accuracy*); and (iii) accuracy in estimating β , partitioned into *variance* and *squared bias*. Results for (i) and (ii) are shown below and those for (iii) in Section 4 of the Supplement. We report the metrics with/without whitening depending on whether it helps or hurts each competing approach; in the scenarios considered LMN, LSD and RL perform better without whitening, and results for FS vary depending on SNR, covariance structure and problem size.

Scenarios with $p = 100$:

Here the dimension $p = 100$ is one fifth of the sample size $n = 500$, and the number of active features is $k_0 = 10$. Figure 3 summarizes results with autoregressive correlations ($\alpha = 0.9$) and SNRs around 0.1, 0.75, 1 and 10 (plots

for all SNRs and both correlation structures are shown in the Supplement, Section 4).

Figure 3(a) shows that all procedures capture a limited number of true positives when the SNR is low. LMN and RL select a few more true positives than other procedures, but are still both too sparse and unspecific. The lack of specificity of LMN and RL does not improve (it worsens) at higher SNRs. In contrast, the parsimonious LSD is rather specific – but it, too, struggles to capture true positives unless signals are strong. MIP-BOOST shows some lack of specificity, but in comparison to LMN and RL this is milder and less dependent on signal strength. In addition, it already captures all true positives at an SNR as small as 0.5 (Figure 5 in the Supplement, Section 4). FS, while appearing rather specific like LSD, catches up more slowly than all others in capturing true positives. Given the strong collinearities, these differences in sensitivity and specificity do not translate in substantial differences in prediction accuracy, as shown in Figure 3(b).

Figure 3(c) shows true and false positive results when all procedures are applied to whitened data. Comparing this to Figure 3(a), the already unspecific LASSO procedures become even more unspecific. FS, too sparse but very specific on the original data, becomes even sparser at low SNRs and loses its specificity at high SNRs. This loss of specificity is yet more pronounced under the block correlation structure (Figure 3 in the Supplement, Section 3; FS selects all 100 features at $\text{SNR} \geq 2$).

Figures 3(e) and 3(f) provide insight on why whitening affects LASSO and MIP-BOOST differently. With whitening the LASSO CVMSE curves give little information on the location of the optimal λ . As the SNR increases, the λ minimizing the error gets further and further away from the λ inducing the right level of sparsity (i.e. corresponding to $k \approx 10$). At $\text{SNR} = 0.1$ the two λ 's are close, but this still corresponds to a solution where more than half of the selected

features are false positives (Figure 3(c)). In contrast, the MIP-BOOST CVMSE curves point more clearly to the right level of sparsity as the SNR increases. Curves shown here are generated by solving on 10 folds across an entire grid of k values ranging from 1 to 100 ².

Scenarios with $p = 1000$:

the dimension $p = 1000$ is twice the sample size $n = 500$ (undersampling). The number of active features remains $k_0 = 10$. Figure 3(d) summarizes true and false positive results with autoregressive correlations ($\alpha = 0.9$) and SNRs 0.1, 0.75, 1 and 10 (complete results are again shown in the Supplement, Section 4). For LASSO procedures and MIP-BOOST, results are remarkably and reassuringly similar to those for $p = 100$. However, the behavior of FS (which here performs best on whitened data) changes notably with respect to scenarios with $p = 100$.

Other scenarios (see Supplement):

We end noting that results with the block structure are consistent with those with the autoregressive structure. However, as the correlation between active and inactive features increases all methods become less effective, even under whitening – likely due to violations of irrepresentability (Zhao and Yu (2006)). To test a lower dimensional but sparser scenario we also simulated data with the block structure for $p = 100$ and $k_0 = 2$. MIP-BOOST is still more specific than LMS and RL and, surprisingly, only LSD benefits from increased sparsity (see Supplement, Section 4.5, Figure 25).

Mazumder et al. (2017) and Hazimeh and Mazumder (2018) proposed the addition of a ridge regularization to help improve solution quality in lower SNR regimes. To explore this, we added an L_2 penalty to MIP-BOOST, and compared solution quality under the $p = 100$ scenario with SNR = 0.1 and AR covariance structure ($\alpha = 0.9$). Results and details of our implementation utilizing the `L0Learn` package in combination with MIP-BOOST are provided in the Supplement (pseudo code in Section 1, Alg. 4 and results in Section 4,

Figure 26). Overall, the statistical quality was very similar with sparse solutions. Adding the L_2 penalty did help reduce the number of false positives, while only slightly reducing the true positives.

4.1.2 Computational Burden

Next, we document the dramatic cut in computing time afforded by MIP-BOOST vs. a standard, or naive, MIP implementation – which selects \hat{k}_0 searching the entire sparsity bound range with traditional cross-validation. This cut renders a rigorously tuned MIP viable in a broader variety of realistic scenarios. We measure the total time required for a 10-fold cross-validation search of the interval between 1 and a maximum sparsity bound c , comparing naive MIP, MIP-BOOST with no surrogate lower bound stopping (Section 3.2), MIP-BOOST+ where the surrogate lower bound is triggered after reaching a 3 minutes *maxtime*, and MIP-BOOST++ where it is applied immediately. We fix again *totaltime* = 10 and 20 minutes for $p = 100$ and $p = 1000$, respectively, and allow 1 thread per instance to mimic settings in previous results (Hastie et al. (2017)). Here, MIP-BOOST serves as a comparison when one can simply afford a higher computational budget on all scenarios, while MIP-BOOST+ has a more realistic *maxtime* but is permitted to continue to at least match the surrogate lower bound. As mentioned in Section 3.2, we do not recommend MIP-BOOST++ when trying to solve each MIP to optimality because of its reliance on surrogate lower bounds which are only meant to assure some level of quality when a realistic *maxtime* is reached without convergence. We provided it as a comparison to measure the accuracy of such surrogate lower bounds.

A conservative estimate of the speedup expected from MIP-BOOST if the computing time required for each MIP evaluation were equivalent to that of naive MIP and equal across the entire search interval is obtained comparing the total number of sparsity bounds explored in traditional cross-validation (say, c) and in the proposed bisection (at most $\log_2(c) + 3$) – see Supplement, Section 5, Figure 26. The benefits of MIP-BOOST become more dramatic the larger is c ,

which is important because the sparsity bound range to be explored increases with the dimension p of the problem. Moreover, actual MIP-BOOST speedups (see below) vastly surpass this conservative estimate.

Figures 4(a)- 4(c) show results for a scenario with $n = 500$, $p = 100$, $k_0 = 10$, autoregressive correlations ($\alpha = 0.9$), and $\text{SNR} = 1$. The range considered extends to $c = 100$. The naive MIP uses traditional 10-fold cross-validation, with a maximum computing time per MIP evaluation set to the more generous 10 (CV10m) minutes as in MIP-BOOST or 3 (CV3m) minutes like MIP-BOOST+ (note again, that MIP-BOOST+ is allowed to go over the 3 minutes if there is still no convergence and surrogate lower bounds are not met). On average, MIP-BOOST is 11.5x faster than CV10m and 5x faster than CV3m. MIP-BOOST+ and MIP-BOOST++ are 24.3x and 199.6x faster than CV10m, respectively. Note that, in spite of potentially exceeding the 3 minute *maxtime*, MIP-BOOST+ is still faster than CV3m.

Importantly, the best upper bound (in-sample error objective) found per instance is essentially the same across procedures (see Supplement, Section 5, Figure 27(a)). In fact, if we omit the bisection and solve across all 100 sparsity bounds, the average upper bound of MIP-BOOST is equal or stronger than that of CV10m and CV3m for 75% and 81% of the values between 1 and $c = 100$, respectively (64% and 77% for MIP-BOOST+, 28% in both cases for MIP-BOOST++). Across the sparsity bound range, the drop in quality of MIP-BOOST++ averages $\sim 0.5\%$ and is never higher than $\sim 2\%$ (see Supplement, Section 5, Table 1 and Figure 28(a)). Notably, Figure 4(b) shows that the cut in computational burden does *not* come at the expense of statistical quality; in fact, all MIP-BOOST variants favor sparser solutions with fewer false positives than CV10m and CV3m.

In Figures 4(c) and 4(d) we raise the dimension to $p = 1000$, leaving all other simulation parameters untouched. We limit the range considered to $c = 50$ to

keep traditional cross-validation viable, and increase the maximum computing time per MIP evaluation to 20 minutes (CV20). Here MIP-BOOST variants are as much as 14.3x faster than CV20m. The average upper bound of all three MIP-BOOST variants is equal or stronger than that of CV20m and CV3m for 72-96% and 98% of the sparsity bounds, respectively (see Supplement, Section 5, Table 1 and Figure 28(b)). For larger sparsity bounds, CV3m often fails to find an integer solution within the time limit; the missing CVMSE values make CV3m difficult to tune. We can still tally true and false positives based on the sparsity bound with minimum CVMSE among those that produced solutions in each fold, but remembering that the search options are biased towards smaller values. With this caveat, Figure 4(d) shows the poor statistical quality of CV3m, with a large and highly variable number of false positives even when choosing a bound close to the true $k_0 = 10$. Here MIP-BOOST is the most effective, followed by MIP-BOOST+; MIP-BOOST++ selects marginally more false positives than CV20m while being 14.3x faster. As expected, MIP-BOOST is best in terms of accuracy; solution quality can only improve if more time is permitted. However, this also holds compared against CV20m – which had the same amount of time but did not utilize our ICV. It was also only marginally more expensive than setting a strict 3 minute run time per instance as in [Hastie et al. \(2017\)](#), with much stronger results. The favorable results of MIP-BOOST+ demonstrate the benefit of allowing additional computing time after the *maxtime* is reached. Finally, MIP-BOOST++ is only slightly cheaper than MIP-BOOST+, but with similar quality to CV20m. Thus, our surrogate lower bounds appear to be an effective strategy.

The results above compare computing time and upper bounds under a restricted maximum computing time per MIP evaluation (20, 10, or 3 minutes). With $p = 1000$, this maximum was reached in the majority of cases and by all procedures. To further demonstrate the effectiveness of MIP-BOOST vs. naive MIP, we compared the time to reach an optimality gap of 5% between traditional cross-validation and ICV at the true $k_0 = 10$, across the 10 cross-validation folds. Here the SNR was raised to 10. This was repeated on 10 different simulated data sets,

totaling 100 MIP runs for the 2 procedures. ICV converged faster in 75 of the 100 cases -providing a median time reduction of 13%.

Finally, to gauge whether the cuts we documented in computing time would hold also in non sparse problems, we repeated the analysis performed for $p = 100$ fixing $k_0 = 50$. We maintained similar computational gains as well as solution quality. Details and results can be found in the Supplement, Section 5.3, Figures 31-32.

4.2 Applications to Diabetes and Bodyfat Data

To further assess our approach, we analyze two real, publicly available, data sets. The first data set was used in [Efron et al. \(2004\)](#). It contains $n = 442$ diabetes patients and 10 baseline predictors (age, body mass index, average blood pressure, six blood serum measurements and sex encoded as $\{0, 1\}$). The response is a measure of disease progression one year after the baseline. The second data set was used to produce predictive equations for lean body weight in [Penrose et al. \(1985\)](#). It contains $n = 252$ men and 13 predictors (age, weight, height, neck, chest, abdomen, hip, thigh, knee, ankle, biceps, forearm, and wrist circumference). The response is the percent of body fat calculated using body density, which is also reported in the data set. We increased problem complexity by including interactions and polynomial terms. For the diabetes data set, following [Efron et al. \(2004\)](#), we included all pair-wise interactions (products) and squared terms for all predictors except for sex – leading to a total $p = 64$ features ($< n$). For the bodyfat data set, we included all polynomial terms up to degree 3 – leading to a total of $p = 559$ features ($> n$). Polynomial terms not only increase the problem size, but also the level of collinearity, mimicking the settings considered in our simulations. Additionally, we again expanded the bodyfat data set with polynomial terms up to degree 3, but including density as a predictor, obtaining $p = 679$ features. This exemplifies a problem comprising one predictor with a strong, though non-linear, relationship with the response; ideally, a selection procedure ought to identify this and avoid redundant features.

We split both data sets into training and test sets with a 75/25 breakdown. As before, we standardized all features to have mean 0 and variance 1. For whitening, we used the MLE for the diabetes regression, where $p < n$, and the sparse covariance estimator proposed in [Bien and Tibshirani \(2011\)](#) through the `spcov` function in R for the bodyfat regressions, where $p > n$. All procedures were fit on both un-whitened and whitened data; Table 2 reports the best results in terms of validation/prediction error. Aside from FS applied to the diabetes regression, all competing procedures performed best on un-whitened data.

MIP-BOOST generated sparser solutions than LMN and RL, however unlike in previous studies ([Hastie et al. \(2017\)](#)) this did not come at a cost in terms of prediction error. For the diabetes regression, it selected Body Mass Index (BMI), Mean Arterial Pressure (MAP) and Lamotrigine (LTG). FS with whitening produced the minimum prediction error, but with as many as 7 features – the three identified by MIP-BOOST, plus HDL cholesterol (HDL), total cholesterol (TCH), blood glucose (GLU) and the interaction between BMI and MAP. Compared to MIP-BOOST, these additional features only reduced the prediction error by 3.8%. LASSO-type methods selected many more features, but again only reducing the prediction error by 1.6% compared to MIP-BOOST (or increasing by 1% for LSD). Notably, LMN and the RL had the same performance; relaxing the LASSO ([Hastie et al. \(2017\)](#)) did not improve it in the diabetes regression.

MIP-BOOST produced the minimum prediction error for both bodyfat regressions. In the regression not including density, the reduction in error vs. other procedures ranged from 6.4 to 26%, and in that including density from 20 to as much as 82%. In the latter case, MIP-BOOST was the only method that solely selected density when given the option. All others included density along with some additional features – which actually led to an increase in prediction error on the test set. In the regression without density, all procedures performed worse, as expected, but MIP-BOOST better exploited combinations of other

measurements to reduce prediction error. The features selected across methods include several higher order and interaction terms (see Table 2, Section 6 of the Supplement). Interestingly, only FS failed to select a term involving height, which – along with weight, abdomen, and wrist circumference – were all included in the equation originally proposed by Penrose et al. (1985) and in the terms selected by the other procedures.

Of course, on real data problems we have no conclusive way to confirm that we have captured all active features, but we have shown that MIP-BOOST can produce sparse, interpretable and competitive solutions (and within reasonable amounts of time, 13 minutes for diabetes and 1-3.5 hours for bodyfat) in both low ($p < n$) and high ($p > n$) dimensional regressions with strongly correlated features.

5 Concluding Remarks and Future Work

Mixed Integer Optimization for L_0 feature selection is receiving much attention due to its foundational appeal and versatility. The effectiveness of this approach, like that of others, depends critically on the selection of a tuning parameter. Pursuing this through cross-validation requires the solution of a large number of MIP instances. Notwithstanding recent improvements in algorithmic efficiency, this can hinder computational viability in problems of realistic size and complexity. MIP-BOOST cuts computational burden by several orders of magnitude, with increasing gains as the dimension of the problem, and thus the tuning parameter range to explore, increase.

Importantly, though our proposals were described for linear models, they could easily be extended to generalized linear models and other classes of statistical models. Also importantly, with MIP as well as other methods, strong feature collinearity and/or weak regression signals can deteriorate statistical quality even when the optimization is well tuned. MIP-BOOST addresses collinearity through ZCA whitening. This is not a new concept, but to our knowledge we are the first

to demonstrate empirically its impact on feature selection. Whitening is attracting increasing attention; [Kessy et al. \(2018\)](#) show how different forms of whitening can benefit different statistical problems. While our numerical studies show that whitening improves recovery in MIP procedures, they do not show a positive effect on LASSO performance. This may be due to the fact that LASSO suffers from poor specificity, not poor sensitivity. The effects of whitening on features selection methods deserve further investigation. In particular, with regards to the MIP approach, it will be critical to investigate whether and how fast an increase in the distance between original and ZCA whitened features erodes the gains in recovery. We also noted how covariance estimation accuracy can affect the effectiveness of whitening, something that is of special concern in high-dimensional settings. In this respect, we plan to explore different covariance and precision estimation approaches ([Fan et al. \(2016\)](#)).

We envision several other avenues for future work. In our simulations, strong signals induced false positives for MIP-BOOST which were easily prevented with stricter improvement thresholds in the bisection – suggesting that this may benefit from data-driven adaptive thresholding. Moreover, we could increase efficiency incorporating new techniques to generate cuts within the Branch & Bound algorithm. [Bertsimas and Van Parys \(2017\)](#) proposed a cutting plane method for sparse ridge regression which, under mild feature collinearity and fairly strong signals, showed good feature selection performance and low computational burden. Also, [Bertsimas et al. \(2017\)](#) proposed adding constraints to balance specific, competing goals of the modeler. Along similar lines, structure in the data could be used to improve solution quality and reduce computation.

6 Supplementary Materials

Supplement-MIP-BOOST.pdf pdf containing pseudo code, additional simulation results, and a convergence proof sketch for Bisection with Feelers

MIPBOOSTex.jl: Julia code running a simulated example of the methods described in the article. Contains functions for each component for general use.

Notes

¹Calculations were performed on the Institute for Computational and Data Sciences Advanced CyberInfrastructure (ICDS-ACI; Penn State University), using the basic memory option on the ACI-B cluster with an Intel Xeon 24 core processor at 2.2 GHz and 128 GB of RAM. The multi-thread option in Gurobi was limited to a maximum of 5 threads for consistency across settings.

²For SNR 10, MIP-BOOST was run with a more conservative threshold set for BF; a quick run of the LASSO selecting a very large number of features can inexpensively diagnose the need to implement a restrictive BF. However, we stress that BF used in MIP-BOOST does *not* suffer from the same lack of specificity that hinders LASSO at high SNRs. When the SNR increases the true k_0 becomes in fact easier to identify for MIP; it is simply that the BF search may be pushed towards the right of the k range if seemingly large differences in CVMSE are not “ignored” by setting a stricter threshold. In Figure 3(e), the elbow at 10 is clearly denoted when SNR=10, but the strong signal leads to distinct drops in the CVMSE also at large k values. A more restrictive threshold easily improves solution quality. In contrast, using a more conservative tuning for LASSO (e.g., the parsimonious LSD), still fails to increase specificity.

References

Beck, A. and Eldar, Y. C. (2013), ‘Sparsity constrained nonlinear optimization: Optimality conditions and algorithms’, *SIAM Journal on Optimization* **23**(3), 1480–1509.

Bertsimas, D., King, A., Mazumder, R. et al. (2016), ‘Best subset selection via a modern optimization lens’, *The Annals of Statistics* **44**(2), 813–852.

Bertsimas, D., King, A. et al. (2017), 'Logistic regression: From art to science', *Statistical Science* **32**(3), 367–384.

Bertsimas, D. and Van Parys, B. (2017), 'Sparse high-dimensional regression: Exact scalable algorithms and phase transitions', *arXiv preprint arXiv:1709.10029*.

Bien, J. and Tibshirani, R. J. (2011), 'Sparse estimation of a covariance matrix', *Biometrika* **98**(4), 807–820.

Chen, D.-S., Batson, R. G. and Dang, Y. (2011), *Applied Integer Programming: Modeling and Solution*, John Wiley & Sons.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2009), *Introduction to algorithms*, MIT press.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R. et al. (2004), 'Least angle regression', *The Annals of statistics* **32**(2), 407–499.

Fan, J., Liao, Y. and Liu, H. (2016), 'An overview of the estimation of large covariance and precision matrices', *The Econometrics Journal* **19**(1), C1–C32.

Fan, J. and Lv, J. (2008), 'Sure independence screening for ultrahigh dimensional feature space', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70**(5), 849–911.

Friedman, J., Hastie, T. and Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of statistical software* **33**(1), 1.

Hastie, T., Tibshirani, R., Friedman, J. and Franklin, J. (2005), 'The elements of statistical learning: data mining, inference and prediction', *The Mathematical Intelligencer* **27**(2), 83–85.

Hastie, T., Tibshirani, R. and Tibshirani, R. J. (2017), 'Extended comparisons of best subset selection, forward stepwise selection, and the lasso', *arXiv preprint arXiv:1707.08692*.

Hazimeh, H. and Mazumder, R. (2018), 'Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms', *arXiv preprint arXiv:1803.01454*.

Kessy, A., Lewin, A. and Strimmer, K. (2018), 'Optimal whitening and decorrelation', *The American Statistician* pp. 1–6.

Mazumder, R., Radchenko, P. and Dedieu, A. (2017), 'Subset selection with shrinkage: Sparse linear modeling when the snr is low', *arXiv preprint arXiv:1708.03288*.

Meier, L., Van De Geer, S. and Bühlmann, P. (2008), 'The group lasso for logistic regression', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70**(1), 53–71.

Meinshausen, N. (2007), 'Relaxed lasso', *Computational Statistics & Data Analysis* **52**(1), 374–393.

Miyashiro, R. and Takano, Y. (2015), 'Mixed integer second-order cone programming formulations for variable selection in linear regression', *European Journal of Operational Research* **247**(3), 721–731.

Patrascu, A. and Necoara, I. (2015), 'Random coordinate descent methods for l_0 regularized convex optimization', *IEEE Transactions on Automatic Control* **60**(7), 1811–1824.

Penrose, K. W., Nelson, A. and Fisher, A. (1985), 'Generalized body composition prediction equation for men using simple measurement techniques', *Medicine & Science in Sports & Exercise* **17**(2), 189.

Schrijver, A. (1986), *Theory of linear and integer programming*, John Wiley & Sons, Inc. New York, NY, USA.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 267–288.

Tibshirani, R. (2011), 'Regression shrinkage and selection via the lasso: a retrospective', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73**(3), 273–282.

Willis, M. J. and von Stosch, M. (2017), 'L0-constrained regression using mixed integer linear programming', *Chemometrics and Intelligent Laboratory Systems* **165**, 29–37.

Zhao, P. and Yu, B. (2006), 'On model selection consistency of lasso', *Journal of Machine learning research* **7**(Nov), 2541–2563.

Zou, H. and Hastie, T. (2005), 'Regularization and variable selection via the elastic net', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **67**(2), 301–320.

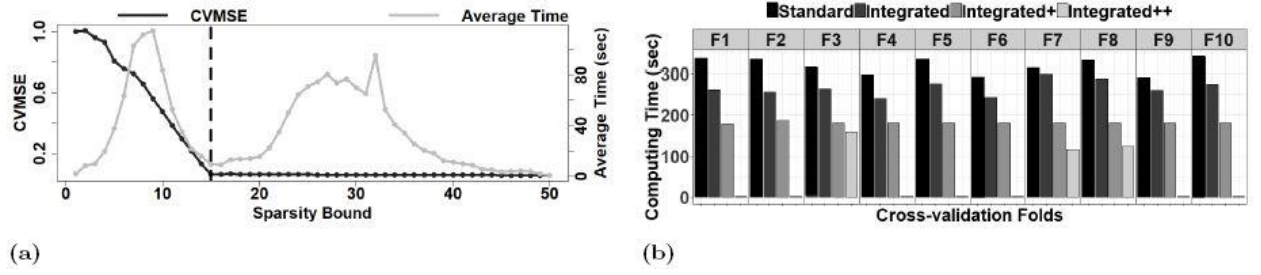


Fig. 1 (a) Cross-validation Mean Squared Error (CVMSE) and average computing time (over the 10 folds) across all values of k , for a simulation scenario with $n = 500$ observations, $p = 50$ features, and $k_0 = 15$. The dashed line marks the true bound. (b) Bar chart of computing times across the 10 folds when applying integrated, integrated+, integrated++, and standard cross-validation (see Section 3.2) at $k = 5$ for a simulation scenario with $n = 500$ observations, $p = 1000$ features, and $k_0 = 10$.

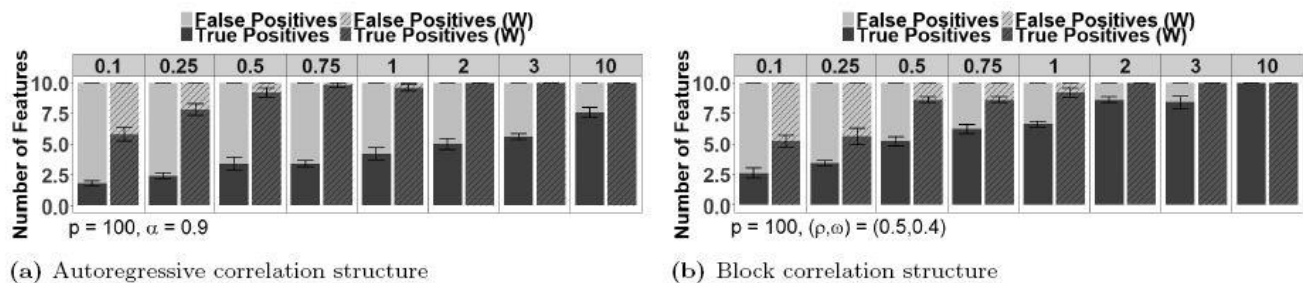


Fig. 2 Breakdown in average numbers of true and false positives selected solving MIPs across a range of SNR values (displayed along the top x -axis, error bars at $\pm 1SD$). Data are simulated from scenarios with $p = 100$, $k_0 = 10$, and highly correlated features under two correlation structures.

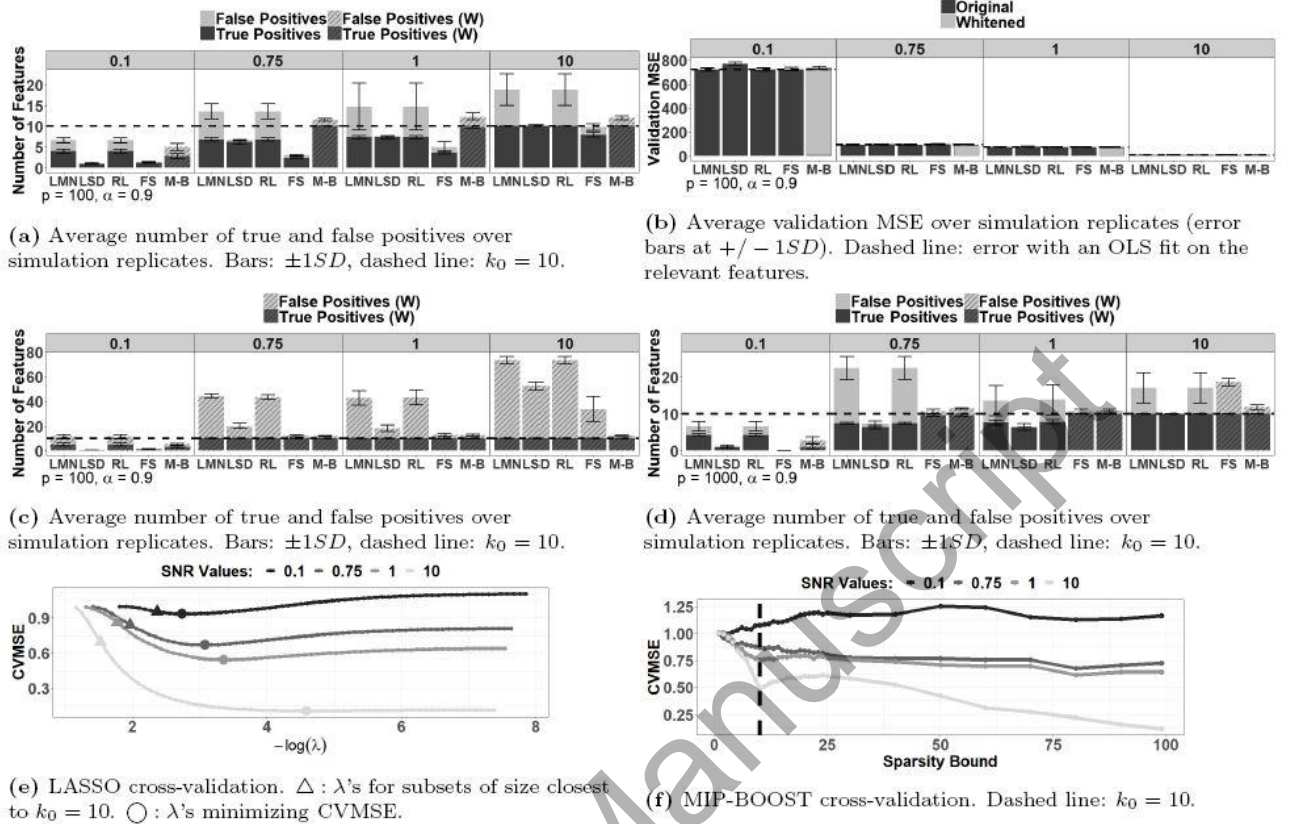
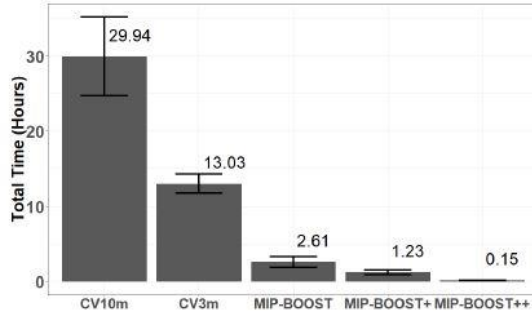
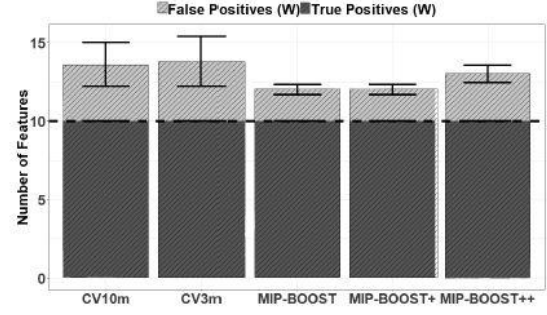


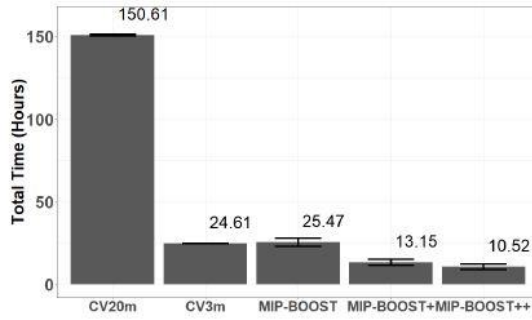
Fig. 3 Summary results in scenarios with $n = 500$, $p = 100$ (panels (a)-(c)), $p = 1000$ (panel (d)), $k_0 = 10$, various SNR values, and autoregressive correlation structure with $\alpha = 0.9$. M-B stands for MIP-BOOST in the bar charts. Gray: original data. Gray with dashes: whitened data (MIP-BOOST for $p = 100$ in (a) and (b), all procedures in (c), MIP-BOOST and FS in (d)). We used 10 replicates for $\text{SNR} \leq 0.75$ and $p = 100$, and 5 for all other scenarios. Cross-validation behavior: LASSO in (e) and MIP-BOOST in (f).



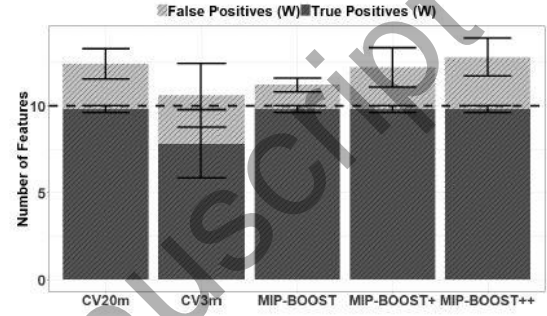
(a) Average total time (in hours). Bars: $\pm 1SD$.



(b) Average no. true and false positives. Bars: $\pm 1SD$, dashed: $k_0 = 10$.



(c) Average total time (in hours). Bars: $\pm 1SD$.



(d) Average no. true and false positives. Bars: $\pm 1SD$, dashed: $k_0 = 10$.

Fig. 4 Summary results in scenarios with $n = 500$, $p = 100$ (panels (a)-(b)); $p = 1000$ (panels (c)-(d)); $k_0 = 10$, autoregressive correlations ($\alpha = 0.9$) and $SNR = 1$. We used 5 replicates.

Table 1 Summary of parameter settings used in the simulation study

Parameter	Values
$n : p : k_0$	500:100,1000:10
α , AUTOREG	*0.8, 0.9
(ρ, ω) , BLOCK	*(0.5,0.1), *(0.5,0.3), *(0.5,0.4)
β	$(1,1,1,1,1,1,1,1,1,0,\dots,0)^T$, ** $(10,10,10,10,10,10,10,10,5,5,5,0,\dots,0)^T$
SNR (R^2)	0.1 (0.09), *0.25 (0.20), *0.50 (0.33), 0.75 (0.43), 1 (0.50) , *2 (0.67), *3 (0.75), 10 (0.91)

*: results in Supplement, Section 4. **: 7 stronger and 3 weaker active signals ($\beta_j = 10$ and 5). Scenarios are replicated 5 or 10 times depending on the SNR. R^2 based on the SNR using the formulation in [Hastie et al. \(2017\)](#).

Table 2 Number of selected features and Validation Mean Squared Error (computed on the test set) for procedures applied to the diabetes and bodyfat regressions.

Procedure	Diabetes		Bodyfat		Bodyfat with Density	
	\hat{k}_0	VALMSE	\hat{k}_0	VALMSE	\hat{k}_0	VALMSE
LMN	19	0.488	15	16.052	6	0.696
LSD	9	0.501	4	20.141	2	2.172
RL	19	0.488	15	16.052	6	0.696
FS	7	*0.477	2	15.857	2	0.488
MIP-BOOST	3	0.496	13	*14.838	1	*0.388

*: minimum Validation MSE.