Mathematical Programming for Simultaneous Feature Selection and Outlier Detection under l1 Norm

Michele Barbato[a,*], Alberto Ceselli[a]

[a]*Department of Computer Science, Università degli Studi di Milano*
18, via Celoria, 20133, Milano, Italy

Abstract

The goal of the simultaneous feature selection and outlier detection problem is to determine a sparse linear regression vector by fitting a dataset possibly affected by the presence of outliers. The problem is well-known in the literature, given its wide range of applications in data analysis, but has been tackled using mathematical programming only recently.

Existing methods have mostly focused on using the traditional least-squares criterion as an objective to determine optimal regression vectors. In this paper, instead, we study models optimizing the least absolute deviation criterion. This change of paradigm allows us to propose two mixed-integer linear programs, one adapted from existing studies on the least-squares setting, and the other one obtained from a disjunctive programming argument. We show theoretically and computationally that the disjunctive-based formulation exhibits better performances in terms of both continuous relaxation quality and integer optimality convergence. Furthermore, we show that both mixed-integer linear programs considered in our paper outperform alternative models based on indicator and special ordered set constraints offered in state-of-the-art mathematical programming solvers.

Finally, we experimentally benchmark these mathematical programming approaches against existing regression methodologies from the literature. We identify a contamination pattern in which mathematical programming is better than state-of-the-art algorithms in combining prediction quality, sparsity and robustness against outliers. Among mathematical programming approaches, those based on least absolute deviations perform best.

Keywords: Data science; Robust sparse regression; Least absolute deviation; Mathematical programming

1. Introduction

In the last two decades huge advances in data collection and digitization techniques have led high-dimensional problems becoming relevant in several fields, such as chemometrics (Bertsimas

*Corresponding author
Email addresses: michele.barbato@unimi.it (Michele Barbato), alberto.ceselli@unimi.it (Alberto Ceselli)

et al., 2020), biomedical research (Insolia et al., 2021), genomics and pattern recognition. Classical references from the literature like Greenshtein (2006) (and references therein) cover this domain.

A key application is *linear regression*. Given is a dataset of points in a $(d+1)$-dimension space: d dimensions represent features, and the last one represents a response.

Intuitively, each point in the dataset is assumed to represent a linearly related input-response pair for a certain phenomenon, whose measurement is affected by noise. With a geometric analogy, this linear relation can be modeled by a hyperplane in a $(d+1)$-dimension space: the regression problem consists in finding the slope and intercept which optimize a predefined notion of proximity between the hyperplane and the points. That is, searching for an optimal linear fitting of points.

Such a fitting hyperplane provides an approximation of the phenomenon measured by the dataset, and can therefore be used either as a descriptive model for it, or more commonly to perform *predictions* of the same phenomenon on new observations.

Linear regression has been studied for decades in statistical data analysis. However, the increase in the dimensionality and size of datasets has recently led to important methodological consequences. First, applications require an accurate feature selection. Such a need is not only computational. For instance, the interpretability of a model produced by fitting is preserved only if a restricted set of relevant explanatory variables are included in it. In the extreme case of *underdetermined regimes* (having more features than points) classical solution methods for linear regression problems do not even apply directly (Filzmoser and Nordhausen, 2021).

Second, dataset contamination chances are greater, yielding (possibly large) errors in either explanatory variables or responses for a few points. These, when errors are large, become *outliers*: their presence in the dataset may affect the fitting quality.

While outlier detection and feature selection have traditionally been treated separately, there is a very recent research trend, attempting to handle them together (Bottmer et al., 2022). At the same time, seminal papers like Bertsimas et al. (2016) opened up new perspectives, not only in the Operations Research community, showing mathematical programming to be a potential breakthrough in these applications. In fact, mixed-integer programming (MIP) formalizations of the regression with *simultaneous feature selection and outlier detection* (SFSOD) problem have been considered independently in a few studies (Insolia et al., 2021; Jammal, 2020; Jammal et al., 2020, 2021; Thompson, 2022). All these works focus on a common assumption: using the least-squares (LS) criterion to measure the proximity of a point to the hyperplane. This is not a coincidence: until recently, having differentiable proximity measures was seen as a necessary feature to operate state of the art regression algorithms.

In this paper we instead focus on the option of solving the SFSOD using a least absolute deviation (LAD) optimization criterion.

Motivations and Contributions. Two observations and one conjecture motivate a research interest on the LAD criterion. The first observation is straightforward: LAD regression models are easy to linearize and solve by means of optimization packages supporting the resolution of mixed-integer

linear programs. The second observation comes from the literature: LAD models tend to produce better solutions in terms of robustness, see *e.g.*, Arslan (2012); Dodge (1997); Wang and Zhu (2017). Our conjecture is instead the following: since the 1-norm upper bounds the 2-norm, the measurement of errors is amplified. Since outliers are those points having highest error, we expect their detection to be easier when minimizing the LAD, as they produce greater impact in the objective functions. This opens, among others, research questions concerning how different types of outliers affect the behaviour of the regression methods considered in our paper.

Despite the good properties of the LAD criterion, the SFSOD has been surprisingly treated only in the LS setting.

As first contribution we propose two MIP modelings of the SFSOD using the LAD optimization criterion. One modeling is the natural adaptation to the LAD setting of the model used in (Insolia et al., 2021; Jammal et al., 2020, 2021; Thompson, 2022) for the LS setting; it exploits auxiliary variables to cancel the residuals of the fitting hyperplane over the selected outliers. The other one elaborates over a canonical disjunctive linearization of the bilinear terms used to eliminate the residuals directly in the objective function. We provide a theoretical analysis of this latter approach, which proves to grant structural advantages with respect to linearization based on the well-known McCormick's envelopes (McCormick, 1976) and to yield stronger LP bounds with respect to the method adapted from the literature. All these modelings involve the use of big-M values bounding the residuals of the fitting hyperplane. We therefore analyze theoretically and computationally the relationships between their magnitude and the quality of their dual bounds.

Our second contribution is application-oriented. We conduct an extensive experimental analysis on synthetic datasets, comparing the MIP-based models, considering also benchmarks from the literature. Our study takes into account (a) computational efficiency of the fitting optimization process (b) quality of the results, when the corresponding regression solutions are used for prediction.

Our paper is organized as follows. In Section 2 we introduce our notation and revise relevant concepts and related approaches from the literature, in Section 3 we detail the MIP models for the SFSOD with LAD objective, in Section 4 we present our computational analysis, and in Section 5 we collect a few conclusions.

2. Notation and background

The problem of linear regression can be formalized as follows. Let

$$\mathcal{P} = \{p^i = (a^i, r_i) \in \mathbb{R}^{d+1}, i = 1 \dots k\}$$

be a set of k points satisfying conditions $r_i = \omega a^i + \zeta + \varepsilon_i$ for every $i = 1, 2, \dots, k$: $\omega \in \mathbb{R}^d$ is a vector of weights, $\zeta \in \mathbb{R}$ is a fixed scalar and $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_k)$ is a vector of random errors. The vector ω and the scalar ζ represent the slope and the intercept, respectively, of a hyperplane linking linearly a d-dimension space of *features* with scalar *responses*.

The goal of a regression problem is to find an estimate $(\hat{w}, \hat{z})$ of (ω, ζ) which optimizes a distance

measure between the hyperplane $\mathcal{H} = \{(x, y) \in \mathbb{R}^{d+1} : y = \hat{w}x + \hat{z}\}$ and the points $p^1, p^2, \ldots, p^k$. *Fitting* is therefore the process of searching for an optimal $(\hat{w}, \hat{z})$ given a dataset as input.

As discussed in the introduction, two improvements of traditional linear regression propose to combine it with (a) *feature selection*, that is, drop dimensions which are irrelevant and (b) *outlier detection*, that is, drop points which represent measurements strongly affected by errors.

Elaborating on the most general form provided in Insolia et al. (2021), we consider a *loss function* $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$ and express the SFSOD problem as the following MIP:

$$\min \sum_{i=1}^{k} \frac{1}{k} s_i \ell(r_i - a^i w - z) \tag{1}$$

$$||w||_0 \leq d_0 \tag{2}$$

$$||s||_0 \geq k - k_0 \tag{3}$$

$$w \in \mathbb{R}^d, z \in \mathbb{R} \tag{4}$$

$$s \in \{0, 1\}^k \tag{5}$$

where $|| \cdot ||_0$ is the 0-norm (counting the nonzero entries of its argument) and d_0 and k_0 are given "budget" values respectively for the number of features that can be selected in the final solution and for the number of points that can be marked as outliers. Constraints (2) and (3) are put as inequalities for flexibility and consistency with the literature. In fact, optimal solutions always exist in which constraint (3) is tight, and constraint (2) is likely to be tight due to numerical reasons. For the sake of generality, function $\ell(\cdot)$ is not given explicitly. A typical choice in the literature is to take $\ell(x) = x^2$, under which (1) corresponds to optimize the least-squares criterion; another common criterion is the least absolute deviation, corresponding to take $\ell(x) = |x|$. Variables $s_1, s_2, \ldots, s_k$ are called *switches* hereafter: given a solution $(\hat{w}, \hat{z}, \hat{s})$ to (1)–(5) point $p^i = (a^i, r_i)$ is selected as an *authentic* point (a non-outlier) if and only if $s_i = 1$.

Sparse and robust regression. Sparsity in regression problems consists in determining faithful predictors relying on small sets of features. The prototype of sparse regression problems is the *best subset selection* (BSS), obtained from (1)–(5) by taking $\ell(x) = x^2$ and $k_0 = 0$. BSS is **NP**-hard (Natarajan, 1995) and it has been considered computationally intractable for decades: an effective MIP-based resolution algorithm for the BSS has only been proposed in the recent work Bertsimas et al. (2016). Previously, sparse regression has been performed through relaxations of BSS. An efficiently solvable convex relaxation of the BSS is the *lasso* (Tibshirani, 1996), obtained by replacing the 0-norm constraint with its 1-norm counter-part, that is, $||w||_1 \leq d_0$. For a suitably chosen $\lambda \geq 0$, the lasso can be expressed in the equivalent "penalized ℓ_1" form

$$\min\{||r - Aw - z||_2^2 + \lambda ||w||_1 : w \in \mathbb{R}^d, z \in \mathbb{R}\},$$

highlighting that sparsity is enforced through the penalization of the term $||w||_1$. A similar penalization approach is used in the *Dantzig estimator* of Candes and Tao (2007) and in the non-convex

SCAD and *MC+ estimators* of Fan and Li (2001) and Zhang (2010); variations of the standard lasso method are also obtained by using multiple or adaptive penalization coefficients, see *e.g.*, Meinshausen (2007); Zou (2006); Zou and Hastie (2005).

The above-mentioned BSS and lasso-like methods are suitable for sparse regression in both the underdetermined ($d < k$) and the overdetermined ($d \geq k$) regimes but they are sensitive to outliers. An approach to robustify these methods against outliers is to consider the loss function $\ell(x) = |x|$, leading to the so-called LAD methods. Several types of LAD-lasso methods are proposed *e.g.*, in Arslan (2012); Wang and Leng (2007); Wang et al. (2007); Xu and Ying (2010) and surveyed in Filzmoser and Nordhausen (2021). Similar alternatives involve robust convex loss functions such as the *Huber's loss function* and the *Tukey's biweight loss function*, see the literature survey of Thompson (2022) for these and additional robust loss functions. Standard regression methods can be also robustified by combining them with *least-trimmed squares* (LTS) paradigm (Rousseeuw, 1984), obtained from (1)–(5) by setting $\ell(x) = x^2$ and $d_0 = k$. The LTS is satisfactorily solvable through fast heuristic algorithms, see *e.g.*, the `R Studio` package `fastLTS` implementing the approach of Rousseeuw and Van Driessen (2006). The *sparseLTS* method of Alfons et al. (2013) (available from `R Studio` package `robustHD` Alfons (2021)) is an embedding of the LTS within the lasso framework and amounts to problem

$$\min\{\sum_{i=1}^{k}(1-s_i)\ell(r_i - a^i w - z) + k_0\lambda||w||_1 \colon ||s||_0 \leq k_0, s \in \{0,1\}^k, w \in \mathbb{R}^d, z \in \mathbb{R}, \}$$

for a suitably chosen $\lambda \geq 0$. A similar LTS embedding in the elastic net lasso of Zou and Hastie (2005) has been studied in Kurnaz et al. (2018a) and is available in the `R Studio` package `enetLTS` (Kurnaz et al., 2018b). Alternative approaches from the literature to get sparsity and robustness against outliers often work in a 2-stage fashion. For example, the *leverage least-trimmed absolute deviation* (LLTA) method of Sudermann-Merx and Rebennack (2021) imitates the LTS idea in the LAD setting but additionally removes potential bad leverage points from the dataset in a pre-processing step. Similarly, *robust principal components regression* and *robust partial least-squares regression* methods described in (Filzmoser and Nordhausen, 2021, Sect. 3.2-3.3) combine a robust dimension reduction with a subsequent robust regression on the set of selected explanatory variables. All above-mentioned robust sparse estimators lack direct control on either the selection of the features (*e.g.*, lasso-based methods) or the selection of the outliers (*e.g.,* BSS-derived methods).

In the present paper we elaborate on the SFSOD defined by (1)–(5), which can be seen as a fully robustified BSS method. Formalizations of the SFSOD as the optimization problem (1)–(5) are present in the literature, see *e.g.,* Chen et al. (2013), but only very recently some works have appeared in which the SFSOD problem has been formalized *and solved* as a MIP. To our knowledge, such an approach to tackle the SFSOD in the LS setting has been first proposed in Jammal et al. (2020, 2021). In Jammal et al. (2020) a lasso version is considered, that is, an ℓ_1-penalization term is included in the objective function. Instead, in Jammal et al. (2021) the SFSOD (1)–(5) in the LS setting is tackled by solving exactly a mixed-integer quadratic program (MIQP); to improve

convergence, the latter is supplied with a warm-start heuristic solution obtained through a *proximal alternating minimization* (PALM) algorithm. The results of Jammal et al. (2020, 2021), together with similar approaches in the domain of support vector machine, are also reported in Jammal (2020). The MIQP used in Jammal et al. (2020, 2021) is independently found and used in two subsequent works (Insolia et al., 2021; Thompson, 2022). Both those studies provide an analysis of the breakdown point of the proposed SFSOD models; moreover, Thompson (2022) provides a primal heuristic algorithm alternative to the above-mentioned PALM algorithm, which is based on a gradient descent technique.

The computational outcome of these works is consistent: when compared to classical methods in sparse or robust regression, the resolution of the SFSOD via MIP resolution provides solutions of high quality in terms of sparsity and robustness, at the price of a greater computational effort.

We also stress that all the works on the SFSOD mentioned above consider the LS setting for their experimental evaluation. There is therefore a clear uncovered research ground on the LAD setting, which in turn is the subject of investigation in our paper.

On handling bilinear terms in MIPs. When considering the SFSOD, a key role is taken by the handling of the product of terms in the objective function (1). In the LAD setting, in fact, they become bilinear terms: linearization techniques are known in the literature to work well in this context. One of them relies on McCormick's envelopes (McCormick, 1976). It replaces each single quadratic term $z = xy$ by a set of linear inequalities allowing the resolution of the initial MIP through standard mixed-integer linear programming (MILP) solvers. Such approach typically needs bounds on the variables appearing in the product hence it introduces big-M values in the resulting MILP. The *spatial branch-and-bound* paradigm tightens the McCormick's envelopes locally to each branch-and-bound node, by exploiting bounds implied by the branching steps. In the context of generic bilinear MIPs Fischetti and Monaci (2020) improves the spatial branch-and-bound scheme by means of non-standard branching rules and by separating valid intersection cuts. State-of-the-art solvers typically offer built-in functionalities to solve optimization problems with specific bilinear terms without relying on explicit linearizations of such terms. For example, Gurobi 9.5.2 Gurobi Optimization (2021) has native support for bilinear terms in the objective function. Moreover, bilinear terms with one binary and one generic bounded variable can be reformulated in Gurobi as well as CPLEX IBM ILOG (2020) through the so-called *Special Ordered Sets of type 1* (SOS-1), that is, expressions of the type $(x, y) \in$ SOS-1 imposing that at most one of variables x and y can have a nonzero value in the solution (Bertsimas and Weismantel, 2005). Indeed the identity $z = xy$ with $x \in \{0, 1\}$ and $y \in \mathbb{R}$ is equivalent to the pair of SOS-1 constraints $(x, z - y) \in$ SOS-1 and $(1 - x, z) \in$ SOS-1. Such constraints are managed by the supporting solvers through specific branching rules thus avoiding big-M terms. We also recall that SOS-1 constraints can be used to model 0-norm constraints as (2) and (3), see *e.g.,* Bertsimas et al. (2016). In particular the LAD version of MIP (1)–(5) can be modeled in both CPLEX and Gurobi using SOS-1 constraints without using big-M values.

In the same setting of products between one binary and one generic bounded variable another

reformulation approach is through the so-called *indicator constraints*: these latter enforce conditions linked to the value of the binary variable in an "on/off" fashion, so that the relation $z = xy$ with $x \in \{0, 1\}$ amounts to specify the two indicator constraints $(x = 0 \Rightarrow z = 0)$ and $(x = 1 \Rightarrow z = y)$. When, as in this case, the conditions amount to satisfy linear constraints, solvers internally reformulate indicator constraints in an extended space and linearize them following McCormick's technique, and the involved variable bounds are automatically tightened during branch-and-bound algorithms, see the study in Belotti et al. (2016).

Finally, we report that the use of disjunctive programming Balas (1998) to linearize specific indicator constraints in the space of natural variables, also when the underlying condition is nonlinear, showed great potential in the literature (Bonami et al., 2015; Hijazi et al., 2012).

3. Formulations and properties

Formally, the SFSOD problem in the LAD setting is obtained by taking $\ell(x) = |x|$ in (1) and thus it amounts to solve

$$\min \left\{ \frac{1}{k} \sum_{i=1}^{k} s_i |r_i - a^i w - z| \colon s, w, z \text{ satisfy (2)–(5)} \right\} \tag{LAD-SFSOD}$$

Given points $p^1 = (a^1, r_1), p^2 = (a^2, r_2), \ldots, p^k = (a^k, r_k)$ as in Sect. 2, we define A as the so-called *design matrix* (whose rows are $a^1, a^2, \ldots, a^k$), and $r = (r_1, r_2, \ldots, r_k)$ in $\mathbb{R}^k$ as the (column) *response vector*.

Let $(\hat{s}, \hat{w}, \hat{z})$ be a feasible solution to (LAD-SFSOD) and let $\hat{A}$ and $\hat{r}$ be obtained from A and r by removing all k_0 rows indexed by $h \in \{1, 2, \ldots, k\}$ such that $\hat{s}_h = 0$. Then the value of $(\hat{s}, \hat{w}, \hat{z})$ in (LAD-SFSOD) corresponds to $||\hat{r} - \hat{A}w - \hat{z}\mathbf{1}||_1$ where $||\cdot||_1$ is the 1-norm and $\mathbf{1}$ is the all 1's column vector of the conforming dimension. For every $i \in \{1, 2, \ldots, k\}$, a point $p^i = (a^i, r_i)$ is labelled as an outlier when $s_i = 0$ while it is labelled as authentic if $s_i = 1$. With this nomenclature we conclude that an optimal solution to (LAD-SFSOD) amounts to select, among all possible combinations, at most k_0 outliers such that the hyperplane $\hat{\mathcal{H}} = \{(x, y) \in \mathbb{R}^{d+1} \colon y = \hat{w}x + \hat{z}\}$ provides the minimum least absolute deviation from the set of authentic points. Moreover, the hyperplane $\hat{\mathcal{H}}$ must be sparse in the sense that $\hat{w}$ must contain at most d_0 nonzero entries, corresponding to features which are *retained*. The remaining features are *discarded*.

In order to solve (LAD-SFSOD) with commercial MILP solvers like CPLEX it is convenient to linearize its objective function and the 0-norm constraints (2) and (3). We proceed in two steps. We first linearize the 0-norm constraints and the absolute value terms in the objective function using the same approach considered in Insolia et al. (2021); Jammal et al. (2021); Thompson (2022); next, we focus on the linearization of the products in the objective function. To perform the first step we assume to know upper- and lower-bound vectors $W^U = (W_1^U, W_2^U, \ldots, W_d^U)$ and $W^L = (W_1^L, W_2^L, \ldots, W_d^L)$ such that $W_j^L \leq \hat{w}_j \leq W_j^U$ for every $j = 1, 2, \ldots, d$ and for every optimal solution $(\hat{s}, \hat{w}, \hat{z})$ to (LAD-SFSOD).

Then, we introduce binary variables $f \in \{0,1\}^d$, each f_j taking value 1 if feature j is retained, 0 if it is discarded, and the continuous variables $p \geq \mathbf{0}$, each p_i upper-bounding the value of the residual on point p^i. Consider the following MIP:

$$(\mathcal{F}) \ \min \frac{1}{k} \sum_{i=1}^{k} s_i p_i \tag{6}$$

$$p_i \geq r_i - a^i w - z \qquad \forall i = 1, 2, \ldots, k \tag{7}$$

$$p_i \geq z + a^i w - r_i \qquad \forall i = 1, 2, \ldots, k \tag{8}$$

$$w_j \geq W_j^L f_j \qquad \forall j = 1, 2, \ldots, d \tag{9}$$

$$w_j \leq W_j^U f_j \qquad \forall j = 1, 2, \ldots, d \tag{10}$$

$$\sum_{j=1}^{d} f_j \leq d_0 \tag{11}$$

$$\sum_{i=1}^{k} s_i \geq k - k_0 \tag{12}$$

$$s \in \{0,1\}^k \tag{13}$$

$$f \in \{0,1\}^d \tag{14}$$

$$p \geq \mathbf{0}, w \in \mathbb{R}^d, z \in \mathbb{R} \tag{15}$$

Proposition 1. *The MIP ($\mathcal{F}$) is equivalent to* (LAD-SFSOD).

Indeed, let $(p^\star, f^\star, s^\star, w^\star, z^\star)$ be a solution to (6)–(15). Since it is a minimization problem, constraints (7)–(8) imply that $p_i^\star = |r_i - w^\star a^i - z^\star|$ for every $i = 1, 2, \ldots, k$ such that $s_i^\star = 1$; moreover, by (9)–(10) we get that $w_j^\star \neq 0$ only if $f_j^\star = 1$, hence (11) guarantees that $w^\star$ has at most d_0 nonzero entries, that is, $||w^\star||_0 \leq d_0$. Since $s^\star \in \{0,1\}^k$ we also immediately get $||s^\star||_0 \geq k - k_0$ from (12). Therefore $(s^\star, w^\star, z^\star)$ is a feasible solution to (LAD-SFSOD) of value $\sum_{i=1}^{k} s_i^\star |r_i - w^\star a^i - z^\star|$; analogously, if $(\hat{s}, \hat{w}, \hat{z})$ is a solution to (LAD-SFSOD) we can define $\hat{f}_j = 0$ if and only if $\hat{w}_j = 0$ for every $j = 1, 2, \ldots, d$ and $\hat{p}_i = |r_i - \hat{w} a^i - \hat{z}|$ for every $i = 1, 2, \ldots, k$. Since W^L and W^U provide valid lower- and upper-bounds on the entries of $\hat{w}$ we get that $(\hat{p}, \hat{f}, \hat{s}, \hat{w}, \hat{z})$ satisfies (9)–(15), showing the equivalence.

3.1. Objective Function Linearizations and Theoretical Analysis

All constraints of the MIP ($\mathcal{F}$) are linear. However, its objective function is bilinear and, in general, non-convex. Therefore in this section we consider two linearizations for the bilinear terms in (1). Both linearizations involve the use of big-M values. For notational convenience, we define $\text{res}_i(w, z) = |r_i - wa^i - z|$ for every $i = 1, 2, \ldots, k$; that is, $\text{res}_i(w, z)$ is the residual of hyperplane $\mathcal{H} = \{(x, y) \in \mathbb{R}^{d+1} \colon y = wz + z\}$ with respect to point p^i. We assume to know a vector $R^U = (R_1^U, R_2^U, \ldots, R_k^U)$ of *valid* upper bounds on the residuals of an optimal solution $(s^\star, w^\star, z^\star)$ to (LAD-SFSOD), that is, $\text{res}_i(w^\star, z^\star) \leq R_i^U$ for every $i = 1, 2, \ldots, k$.

The first linearization of (6) we consider is a refinement of a standard technique based on disjunctive programming. Let us fix $i \in \{1, 2, \ldots, k\}$ and consider an auxiliary variable $x_i \geq 0$ defined by $x_i = s_i p_i$; since $s_i \in \{0, 1\}$ we may restrict to consider the solutions $(\hat{x}, \hat{p}, \hat{f}, \hat{s}, \hat{w}, \hat{z})$ such that one of the two cases holds: either $\hat{s}_i = 1$ and hence $0 \leq \hat{x}_i = \hat{p}_i \leq R_i^U$; or $\hat{s}_i = 0$ and hence $\hat{x}_i = 0$ and $\hat{p}_i = R_i^U$. The latter holds because when $\hat{s}_i = \hat{x}_i = 0$ defining $\hat{p}_i = R_i^U$ necessarily satisfies all constraints (7)–(15) and does not have any impact on the objective function. Equivalently, this disjunction is expressed by $(\hat{x}_i, \hat{p}_i, \hat{s}_i) \in P_0^i \cup P_1^i$ where $P_0^i = \{(0, R_i^U, 0)\}$ and $P_1^i = \{(\pi, \pi, 1) \colon 0 \leq \pi \leq R_i^U\}$ for every $i = 1, 2, \ldots, k$; since P_0^i is a single point in $\mathbb{R}^3$, while P_1^i is a segment, the convex hull of their union is easily determined as:

$$\mathrm{conv}(P_0^i \cup P_1^i) = \{(x_i, p_i, s_i) \colon x_i = p_i - R_i^U(1 - s_i), 0 \leq x_i \leq p_i \leq R_i^U, 0 \leq s_i \leq 1\}.$$

Projecting out the x variables we get the following disjunctive-based linearization of (LAD-SFSOD):

$$(\mathcal{D}) \quad \min \frac{1}{k} \sum_{i=1}^{k} (p_i - (1 - s_i) R_i^U) \tag{16}$$

$$p_i \geq R_i^U (1 - s_i) \qquad \forall i = 1, 2, \ldots, k \tag{17}$$

$$p_i \leq R_i^U \qquad \forall i = 1, 2, \ldots, k \tag{18}$$

$$(p, f, s, w, z) \text{ satisfy (7)–(15).} \tag{19}$$

Observation 1. *Since $(\mathcal{D})$ is a minimization problem, an optimal solution will automatically satisfy constraints* (18)*. Therefore these latter may be omitted from $(\mathcal{D})$ without affecting the correctness of the model. We include them since they appear explicitly in the description of conv$(P_0^i \cup P_1^i)$ given above.*

We point out the following relation to exist between the disjunctive-based formulation $(\mathcal{D})$ and the linearization of (6) via McCormick's envelopes.

Proposition 2. *The polyhedron corresponding to the continuous relaxation of $(\mathcal{D})$ is a face of that corresponding to the continuous relaxation of McCormick's reformulation of $(\mathcal{F})$.*

For the sake of conciseness, here we do not report the McCormick's reformulation of $(\mathcal{F})$ nor the formal derivation of the latter statement. They can be found in Appendix A. Another important result, is the following.

Observation 2. *The optimal continuous relaxation value of $(\mathcal{D})$ and that of McCormick's reformulation coincide,*

even if the set of feasible solutions does not. The proof is reported in Appendix A; it exploits the fact that the variables appearing in McCormick's reformulation are binding in constraints where they appear.

At the same time, the disjunctive-based formulation has two advantages. First, it is of smaller size than McCormick's reformulation. The second advantage is the following.

Observation 3. *By fixing to 0 a switch variable s_i for some $i \in \{1, 2 \dots, k\}$ formulation $(\mathcal{D})$ automatically implies the constraint $p_i = R_i^U$,*

which does not hold for McCormick's reformulation (for which, in general $\mathbf{0} \leq p \leq R^U$ independently of the value of the s variables). Since such a fixing is a typical operation in several MIP resolution techniques (*e.g.*, branching in branch-and-bound methods), this property offers computational advantage to the disjunctive-based formulation. In fact, as reported in (Vielma, 2015, Sect. 2.2), effective MIP formulations combine small size with strong LP bounds and with good behaviour under branching. In this regard the disjunctive-based reformulation of LAD-SFSOD is superior to McCormick's one in all these aspects, hence we will focus on $(\mathcal{D})$ from now on.

We now move to the second linearization of the objective function of $(\mathcal{F})$. It arises from an adaptation to the LAD setting of the SFSOD with LS objective (LS-SFSOD), considered independently in Insolia et al. (2021); Jammal et al. (2021); Thompson (2022). All these three works introduce variables which cancel the residuals in correspondence of outliers. We are not aware of alternative exact modelling of the LS-SFSOD in the literature. For the sake of structural comparison with the literature, we consider the corresponding LAD version, which can be obtained by using the loss function $\ell(x) = |x|$ instead of $\ell(x) = x^2$ in the model of Insolia et al. (2021); Jammal et al. (2021); Thompson (2022). This leads to the following linearization of (LAD-SFSOD):

$$(\mathcal{L}) \ \min \frac{1}{k} \sum_{i=1}^{k} q_i \tag{20}$$

$$q_i + r_i - wa^i - z - t_i \geq 0 \qquad \forall i = 1, 2, \dots, k \tag{21}$$

$$r_i - wa^i - z - t_i - q_i \leq 0 \qquad \forall i = 1, 2, \dots, k \tag{22}$$

$$t_i \geq -(1 - s_i)R_i^U \qquad \forall i = 1, 2, \dots, k \tag{23}$$

$$t_i \leq (1 - s_i)R_i^U \qquad \forall i = 1, 2, \dots, k \tag{24}$$

$$(f, s, w, z) \text{ satisfy } (9)\text{–}(15) \tag{25}$$

$$q \geq \mathbf{0}, t \in \mathbb{R}^k. \tag{26}$$

Variables t, appearing in (20)–(26), are used to cancel the residual in correspondence of outliers, which in turn are selected by the switch variables s, while in $(\mathcal{D})$ this is done directly with the binary switch variables.

Our finding is that the disjunctive-based linearization is inherently different from the linearization $(\mathcal{L})$ proposed in the literature. Namely, we are able to theoretically prove that the continuous relaxation value of the disjunctive-based formulation is at least as strong as the continuous relaxation value of the literature MILP; next, we derive sufficient conditions on the bound vector R^U for the two continuous relaxation values to coincide; finally we experimentally show that by violating such a condition we can construct instances on which the continuous relaxation of the disjunctive-based formulation is strictly stronger. To fix notation let $\bar{\mathcal{D}}_{R^U}$ and $\bar{\mathcal{L}}_{R^U}$ be the polytopes arising from the continuous relaxations of $(\mathcal{D})$ and $(\mathcal{L})$ respectively and let $v(\bar{\mathcal{D}}_{R^U})$ and $v(\bar{\mathcal{L}}_{R^U})$ indicate

the corresponding continuous relaxation values. The following result holds for every big-M vector R^U used in the above formulations (also non-valid ones).

Proposition 3. *$v(\bar{\mathcal{D}}_{R^U}) \geq v(\bar{\mathcal{L}}_{R^U})$ for every $R^U \geq \mathbf{0}$.*

Proof. Let $(\tilde{p}, \tilde{f}, \tilde{s}, \tilde{w}, \tilde{z}) \in \bar{\mathcal{D}}_{R^U}$; we determine $\tilde{q}, \tilde{t}$ such that $(\tilde{q}, \tilde{t}, \tilde{f}, \tilde{s}, \tilde{w}, \tilde{z}) \in \bar{\mathcal{L}}_{R^U}$ and $\sum_{i=1}^{k}(\tilde{p}_i - (1-\tilde{s}_i)R_i^U) = \sum_{i=1}^{k} \tilde{q}_i$, which therefore proves the claim.

To this end, we define $\tilde{q}_i = \tilde{p}_i - (1-\tilde{s}_i)R_i^U$. Then the equivalence of the objective functions is immediate. Moreover, from (17) it follows that $\tilde{q}_i \geq 0$. Finally, let $I_i = [r_i - \tilde{w}a^i - \tilde{z} - \tilde{q}_i, r_i - \tilde{w}a^i - \tilde{z} + \tilde{q}_i] \cap [-(1-\tilde{s}_i)R_i^U, (1-\tilde{s}_i)R_i^U]$ for every $i = 1, 2, \ldots, k$. Note that $I \neq \emptyset$: from (7) we get $r_i - \tilde{w}a^i - \tilde{z} - \tilde{q}_i \leq (1-\tilde{s}_i)R_i^U$ while (8) implies $r_i - \tilde{w}a^i - \tilde{z} + \tilde{q}_i \geq -(1-\tilde{s}_i)R_i^U$. Let us pick $\tilde{t}_i \in I_i$ for every $i = 1, 2, \ldots, k$. Then $(\tilde{q}, \tilde{t}, \tilde{f}, \tilde{s}, \tilde{w}, \tilde{z}) \in \bar{\mathcal{L}}_{R^U}$.

□

We now present a sufficient condition for $v(\bar{\mathcal{D}}_{R^U}) = v(\bar{\mathcal{L}}_{R^U})$.

Proposition 4. *If $R_i^U \geq \max\{|r_i - wa^i - z| \colon (q, t, f, s, w, z)$ attains $v(\bar{\mathcal{L}}_{R^U})\}$ for $i = 1, 2, \ldots, k$ then $v(\bar{\mathcal{D}}_{R^U}) = v(\bar{\mathcal{L}}_{R^U})$.*

Proof. In view of Prop. 3 we only need to prove that $v(\bar{\mathcal{D}}_{R^U}) \leq v(\bar{\mathcal{L}}_{R^U})$ when R^U is as in statement of Prop. 4. Let $(\bar{q}, \bar{t}, \bar{f}, \bar{s}, \bar{w}, \bar{z}) \in \bar{\mathcal{L}}_{R^U}$ attaining value $v(\bar{\mathcal{L}}_{R^U})$. Then $\bar{q}_j \leq \bar{s}_j R_j^U$ for every $j = 1, 2, \ldots, k$. Indeed, suppose by contradiction that $\bar{q}_i > \bar{s}_i R_i^U$ for some $i \in \{1, 2, \ldots, k\}$. Since $(\bar{q}, \bar{t}, \bar{f}, \bar{s}, \bar{w}, \bar{z})$ attains the minimum of (20) on $\bar{\mathcal{L}}_{R^U}$, constraints (21) and (22) imply $\bar{q}_i = |r_i - \bar{w}a^i - \bar{z} - \bar{t}_i|$; then $|r_i - \bar{w}a^i - \bar{z} - \bar{t}_i| > s_i R_i^U$. Let first $r_i - \bar{w}a^i - \bar{z} - \bar{t}_i \geq 0$. Then $R_i^U \geq r_i - \bar{w}a^i - \bar{z} > s_i R_i^U + \bar{t}_i$ since R_i^U is a valid upper bound on $|r_i - \bar{w}a^i - \bar{z}|$ by the hypothesis. Then $\bar{t}_i < (1-\bar{s}_i)R_i^U$. Let $\varepsilon = \min\{(1-\bar{s}_i)R_i^U - \bar{t}_i, \bar{q}_i - \bar{s}_i R_i^U\}$. We define $q'_i := \bar{q}_i - \varepsilon$, $t'_i := \bar{t}_i + \varepsilon$ and $q'_j = \bar{q}_j$, $t'_j = \bar{t}_j$ for every $j \neq i$. By the definitions, $\bar{q}_i > q'_i \geq 0$ and $r_i - \bar{w}a^i - \bar{z} - t'_i = q'_i$. Hence $(q', t', \bar{f}, \bar{s}, \bar{w}, \bar{z}) \in \bar{\mathcal{L}}_{R^U}$ and its value is strictly less than $v(\bar{\mathcal{L}}_{R^U})$, a contradiction. The case $r_i - \bar{w}a^i - \bar{z} - \bar{t}_i < 0$ is analogous. To conclude we define $\bar{p}_j = \bar{q}_j + (1-\bar{s}_j)R_j^U$ for every $j = 1, 2, \ldots, k$. The fact that $\bar{q}_j \leq \bar{s}_j R_j^U$ for every $j = 1, 2, \ldots, k$ implies that $\bar{p}_j \leq R_i^U$. All other constraints of (17)–(19) are easily seen to be satisfied by $(\bar{p}, \bar{s}, \bar{w}, \bar{z})$ and the latter obviously has value $v(\bar{\mathcal{L}}_{R^U})$. Thus $v(\bar{\mathcal{D}}_{R^U}) \leq v(\bar{\mathcal{L}}_{R^U})$. □

We recall the following.

Observation 4. *When R^U is a vector of valid upper bounds on the residuals of an optimal solution of* (LAD-SFSOD)*, MILPs ($\mathcal{D}$) and ($\mathcal{L}$) are both valid reformulations of* (LAD-SFSOD).

In fact, if for some $i \in \{1, 2, \ldots, k\}$ the value R_i^U does not satisfy the assumption in Prop. 4, inequality $q_i \leq s_i R_i^U$ is not implied by the constraints defining $\bar{\mathcal{L}}_{R^U}$. Therefore using them may lead to $v(\bar{\mathcal{L}}_{R^U}) < v(\bar{\mathcal{D}}_{R^U})$. In particular, this is the case of the bounds vectors R^U as those considered in Obs. 4. We have actually observed experimentally this phenomenon to occur frequently, see Sect. 4.1.

3.2. Big-M Computation

As reported in Thompson (2022) it "remains an open research question how to estimate provably correct parameters in the absence of any assumptions on" the design matrix. In fact, for the SFSOD problem, it is nontrivial to determine big-M values (even loose ones) which do not cut off optimal solutions.

As a consequence, the works on the SFSOD resort to the following heuristic method: first a primal solution $(\omega^\star, \zeta^\star)$ and the corresponding residuals $\text{res}_1^\star, \text{res}_2^\star, \ldots, \text{res}_k^\star$ are computed; then (arbitrary) multipliers $m_1, m_2 \geq 1$ are selected, allowing to set $W_j^L = -m_1|\omega_j^\star|$ and $W_j^U = m_1|\omega_j^\star|$ for every $j = 1, 2, \ldots, d$ and $R_i^U = m_2 \text{res}_i^\star$ for every $i = 1, 2, \ldots, k$. A refinement of this technique exploiting multiple primal solutions is given in Insolia et al. (2021).

The approach just described is not exact since it could cut off the global minimum of (LAD-SFSOD) (or its LS counter-part), although it maximally improves the starting primal solutions without worsening its residuals nor changing its sparsity: if $w_j^\star = 0$ for some $j = 1, 2, \ldots, d$ then also $W_j^L = W_j^U = 0$. More flexibility on the sparsity of the resulting vector is obtained in Jammal et al. (2021); Thompson (2022) by setting $W_j^L = -m_1||w^\star||_\infty$ and $W_j^U = m_1||w^\star||_\infty$, but in this case all entries of W^L, as well as those of W^U, coincide.

Building on alternative ideas from the literature, and in particular on the heuristic approach presented in Bertsimas et al. (2016) to calculate W^L, W^U for the BSS in the overdetermined regime ($k > d$ with at least d points in general position), we proceed as follows. Let $(w^\star, z^\star)$ be hyperplane coefficients of a feasible solution to (LAD-SFSOD) and let $v^\star = ||r - Aw^\star - z^\star||_2^2$; for every $j = 1, 2, \ldots, d$ we get W_j^U and W_j^L by solving the following pair of convex optimization programs:

$$
\begin{aligned}
W_j^L = \min\ & w_j \\
& ||r - Aw - z||_2^2 \leq v^\star \\
& w \in \mathbb{R}^d, z \in \mathbb{R},
\end{aligned}
\qquad\qquad
\begin{aligned}
W_j^U = \max\ & w_j \\
& ||r - Aw - z||_2^2 \leq v^\star \\
& w \in \mathbb{R}^d, z \in \mathbb{R}.
\end{aligned}
$$

Since the domain of the above problems is a compact set of $\mathbb{R}^{d+1}$, they both admit finite optimal values. These latter can be computed analytically exploiting the convexity of the problem domain, see the formulas in the supplementary material of Bertsimas et al. (2016).

We remark that while the approach of Bertsimas et al. (2016) to the BSS allows an exact computation of valid big-M values, our adaptation to the LAD-SFSOD turns out to be heuristic: the 2-norm of the total residual of an optimal solution to the (LAD-SFSOD) could be larger than the one of a sub-optimal solution. We point out that, with this approach, $w_j^\star = 0$ does not necessarily implies $W_j^U = 0$ nor $W_j^L = 0$ for any $j \in \{1, 2, \ldots, d\}$ and the bounds are not necessarily coinciding.

We heuristically compute a bound vector R^U on the residuals in a similar manner. Namely, by

first solving the following convex optimization programs for every $i = 1, 2, \ldots, k$:

$$\begin{aligned} S_i^L = \min\, & a^i w \\ & ||r - Aw - z||_2^2 \leq v^\star \\ & w \in \mathbb{R}^d, z \in \mathbb{R}, \end{aligned} \qquad \begin{aligned} S_i^U = \max\, & a^i w \\ & ||r - Aw - z||_2^2 \leq v^\star \\ & w \in \mathbb{R}^d, z \in \mathbb{R}. \end{aligned}$$

and then using $|r^i - a^i w - z| \leq |r^i| + |z| + \max\{|S_i^L|, |S_i^U|\} =: R_i^U$; in this case the convex programs defining S_i^L and S_i^U admit analytical solutions in both the underdetermined and overdetermined regimes (see the supplementary material of Bertsimas et al. (2016)).

4. Computational Results

In this section we evaluate branch-and-bound algorithms based on the formulations presented in Sect. 3. The comparison is performed from two standpoints. On one hand we consider the computational performance of the formulations, measured as continuous relaxation quality, CPU time needed to reach integer optimality and optimality gap for unsolved instances. We also compare the performance of the models based on big-M values with models based on indicator and SOS-1 constraints supported by CPLEX and Gurobi. On the other hand we consider the prediction quality of the regression hyperplanes found as solutions to (LAD-SFSOD) by means of the formulations of Sect. 3.

Instances. In order to analyse the dependency of both computational and quality performances of the considered MILPs on the type of outliers, we define synthetic instances, so to have full control on their structure. Let d and k be fixed. Each instance in our experiments is composed by 5 replications of two datasets, each containing k points in $\mathbb{R}^{d+1}$. In each replication the first dataset is for training, the other is for testing. Moreover, each instance depends on a feature vector $\omega \in \mathbb{R}^d$, on a signal-to-noise ratio (SNR) value α, on an outlier ratio $0 < \pi < 1/2$ and on response and design error means μ_r and μ_A. The design matrices of both datasets are initially constructed row-wise, drawing each row from a multi-variate normal distribution $\mathcal{N}(0, \mathbb{I})$ with $\mathbb{I}$ being the identity matrix. We observe that this specific generation criterion yields normalized data. Therefore no further standardization techniques to increase numerical stability and improve bounds computation is needed. Given also $\sigma^2 = ||\omega||_2^2/\alpha$ the response vector is defined by $r_i = \omega a^i + \zeta + \varepsilon_i$ for every $i = 1, 2, \ldots, k$ with ζ extracted from a $\mathcal{N}(0, 1)$ distribution and ε_i extracted from a $\mathcal{N}(0, \sigma^2)$ distribution. In a subsequent step we consider one of the two corruption schemes, only for the training dataset:

1. corruption only in the responses (*vertical outliers*);
2. corruption in both design matrix and responses (*bad leverage points*).

In both schemes, each point is independently chosen as outlier with a given probability π. For outlier points, the corresponding response entry is corrupted, changing it by a value extracted from a $\mathcal{N}(\mu_r, 1)$ distribution. Only in scheme 2, also the design matrix is corrupted by changing the entries in the rows corresponding to outliers by a value extracted from a $\mathcal{N}(\mu_A, 1)$ distribution.

Only entries corresponding to nonzero coefficients in the feature vector are corrupted in this way. For both corruption schemes, we consider the following parameters:

- $d = 50$;
- $k = 100, 150, 200$;
- $\omega \in \mathbb{R}^d$ with its first 5 entries equal to 1, and all others equal to 0;
- $\alpha = 5$;
- $\mu_r = -10$ and $\mu_A = 10$;
- $\pi = 0.1$.

The parameters above are essentially those used in Insolia et al. (2021) for the tests on synthetic instances. The only differences are in the definition of ω, whose first 5 entries are set to 2 in Insolia et al. (2021), and in the fact that here we focus on the overdetermined regime (having $k > d$). As reported in Dodge (1997), the LAD fitting is automatically unaffected by vertical outliers, when the corruption noise on the response has the same sign of the measurement noise. However, our models do not rely on this assumption, allowing us to handle arbitrary adversarial corruptions as well. Our training instances, in fact, cover also this general case (including, in particular, cases where μ_r and the ε_i's have opposite sign).

Computation of W^L, W^U and R^U. Vectors W^L and W^U are computed via the resolution of convex optimization programs reported in Sect. 3.2. The feasible solution needed for the computation of W^L and W^U is obtained by the PALM algorithm of Jammal et al. (2020). The PALM algorithm was designed to produce a primal solution for the SFSOD problem in the LS setting but such a solution is obviously also feasible for (LAD-SFSOD). Pseudo-code and convergence guarantees of such an algorithm can be found in Jammal et al. (2020). The convex programming-based approach for getting R^U turned out to produce very large bounds, resulting in numerical instability and poor performance of the branch-and-bound algorithms used in this section. Since we are interested in studying the strength of the formulations of Sect. 3 depending on the magnitude of the bounds in R^U, we proceed differently as follows. For each instance replication we define $E_i = |r_i - \omega a^i - \zeta|$ for every $i = 1, 2, \ldots, k$, where r_i and a^i are the response and design row of the i-th point in the instance after applying the corruption step. Vector $E = (E_1, E_2, \ldots, E_k)$ represents the smallest bounds on the residuals. In our experiments we will set $R^U = \phi E$ with $\phi \geq 1$ being a parameter allowing us to control the tightness of the big-M used in the MIP formulations of Sect. 3. We point out that using such big-M values could cut off the global optimum of (LAD-SFSOD).

Implementation details. The instance generation script is implemented in `R`. The MIP formulations presented in Sect. 3 are implemented using the `C++` API of CPLEX 20.10 and solved with the branch-and-bound framework offered by the same optimization suite. Unlike formulations ($\mathcal{D}$) and ($\mathcal{L}$) given above, in our implementation we omit the term $1/k$ in the objective function, since in preliminary

tests we observed that it did not improve the computational performance. The corresponding stability effect is probably already given by the rescaling of CPLEX. The redundant constraints (18) discussed in Obs. 1 are implemented as upper bounds of variables p_i's so that CPLEX manages them automatically. Each run is performed in sequential mode (1 thread per run) on a Linux machine (Ubuntu 20.04.4) equipped with 32 GB of RAM and with a 4.10 GHz CPU (model Intel(R) Xeon(R) W-1250P). The whole test session is performed in parallel using the `parallel` package of `R`. We use the default setting of CPLEX except for the parameters `LocalImplied` and `Implied` that are set to their maximum value (3 and 2 respectively). These values enforce local and global implied bound constraints in the most aggressive way allowed by CPLEX. Such cuts may substantially speed-up the convergence of branch-and-bound algorithms, especially when big-M values are involved (Belotti et al., 2016). The primal solution produced by the PALM algorithm in the computation of the bound vectors W^L and W^U is also used as warm start of CPLEX branch-and-bound procedure.

4.1. Comparison of continuous relaxations

In this section we experimentally compare the continuous relaxation of formulations ($\mathcal{D}$) and ($\mathcal{L}$). Prop. 3 states the former to never be weaker than the latter. Our main finding is the following.

Experimental Observation 1. *The continuous relaxation of the disjunctive-based formulation ($\mathcal{D}$) is, in general,* strictly *stronger than the continuous relaxation of ($\mathcal{L}$).*

We recall that according to Prop. 4 a necessary condition to guarantee such a result is that the value of R_i^U is small enough for at least one $i \in \{1, 2, \ldots, k\}$. Experimentally, we compare the relative increase of the LP bound obtained by the disjunctive-based approach with respect to the literature approach, by considering sequences of vectors R^U having increasing entries. Let $v(\mathcal{D}_{R^U})$ and $v(\mathcal{L}_{R^U})$ be the LP bound values of ($\mathcal{D}$) and ($\mathcal{L}$) respectively. The relative increase of the LP bound is computed as $100 \cdot \frac{v(\mathcal{D}_{R^U}) - v(\mathcal{L}_{R^U})}{v(\mathcal{L}_{R^U})}$. In this experiment we use $R^U = \phi E$ with $\phi = 1, 1.2, 1.5$; that is, we consider the vector with the smallest bounds and two additional bound vectors whose entries are respectively increased by 20% and 50% with respect to the smallest ones. We consider datasets polluted with bad leverage points, that is, following scheme 2. For each instance replication we solve with CPLEX the continuous relaxations of both formulations ($\mathcal{D}$) and ($\mathcal{L}$), where we consider all combinations of $d_0 = 5, 6, \ldots, 10$ and $k_0 = \lceil 0.1k \rceil, \lceil 0.13k \rceil, \lceil 0.17k \rceil, \lceil 0.2k \rceil$ (corresponding to 10%, 13%, 17% and 20% of points selectable as outliers). This gives a grid of 24 parameter combinations for each considered formulation; since each instance consists of 5 replications we get a total of 120 runs for each dataset size. Averaging over all the runs executed in this section, the continuous relaxation of each formulation is computed in less than 0.06 CPU seconds.

Detailed computational results are provided in Table 1. Interestingly, the continuous relaxation values do not depend on d_0 for any formulation. Therefore in Table 1 we report results according to the dataset size (column "k") and to the percentage of points selectable as outliers (column "k_0 (%)"). The remaining three columns give the LP bound values and the relative increase depending on the value of ϕ. An additional line ("Avg.") reports the average of the relative increases for each dataset size. We immediately observe that the continuous relaxation values of the disjunctive-based

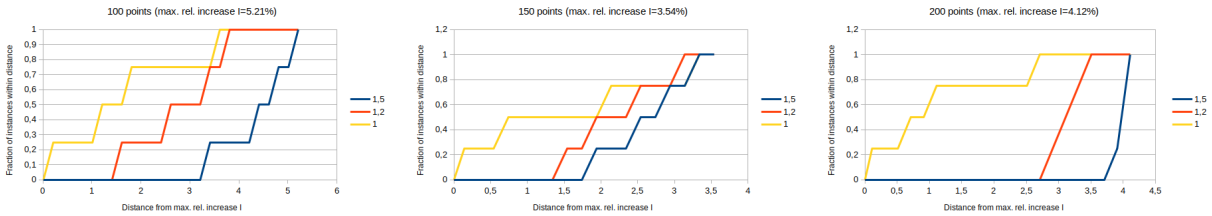

Figure 1: Performance profiles of the LP-bound increase of ($\mathcal{D}$) over ($\mathcal{L}$).

formulation are strictly larger than those of the literature formulation and that larger relative increases correspond to smaller values of ϕ. This is consistent with the theoretical findings of Sect. 3.1. The table also clearly shows that the relative increase improves when k_0 is larger. In general, the average relative increase worsens by increasing the dataset size, although some exceptions are present ($k = 200$ with $\phi = 1$ and $k = 150$ with $\phi = 1.5$).

To better display how the strength of the disjunctive-based formulation depends on the big-M magnitude, in Figure 1 we provide an aggregated overview of the same experiment. The pictures in Figure 1 are constructed according to the principle of cumulative distribution functions as follows. For each dataset size and each value of ϕ, we take the maximum relative increase I in the corresponding column of Table 1; then for every value λ on the x-axis we report the proportion of runs (y-axis) yielding a given LP bound relative increase of at least $I - \lambda$. In other words, a value on the x-axis of Figure 1 represents the distance from the best relative increase registered in our experiments. For a given curve of the figure, the corresponding value on the y-axis is the proportion of runs within such a distance. In particular, upper curves correspond to better performance than lower curves. In each picture of Figure 1 we have three curves, corresponding to the three values of ϕ. With this interpretation, the three pictures clearly show that, for every dataset size, vectors R^U with smaller bounds (*i.e.*, smaller values of ϕ) correspond to larger relative increases.

4.2. Comparison of MIP Performances

In this section we analyse the computational performance of the branch-and-bound algorithms relying on formulations ($\mathcal{D}$) and ($\mathcal{L}$) of Sect. 3 (for short, we will indicate the algorithms by ($\mathcal{D}$) and ($\mathcal{L}$) as well). We use the same grid of combinations for k_0 and d_0 employed in Sect. 4.1. That is, for each corruption scheme, each algorithm is executed over a total of 360 runs (120 for each dataset size). Each run has a CPU time limit of 3600 seconds. We define $R^U = 2E$ for both MILPs. This corresponds to increase by 100% the smallest residuals computed at instance generation.

Results are presented in terms of CPU time to reach optimality and relative optimality gap upon termination; this latter is obtained by using the formula $(UB - LB)/UB$ where UB and LB are the best upper- and lower-bound obtained when the optimization process ends (either by optimality or because of time limit). A broad computational campaign is provided in the companion technical

Table 1: LP bound values of formulations of Sect. 3 and relative increase of $v(\mathcal{D})$ over $v(\mathcal{L})$. We consider LAD-SFSOD on instances corrupted with bad leverage points.

k	k_0 (%)	$\phi = 1$			$\phi = 1.2$			$\phi = 1.5$		
		$v(\mathcal{L})$	$v(\mathcal{D})$	Incr.(%)	$v(\mathcal{L})$	$v(\mathcal{D})$	Incr.(%)	$v(\mathcal{L})$	$v(\mathcal{D})$	Incr.(%)
100	10	37.45	38.09	1.69	30.58	31.04	1.49	23.81	23.86	0.19
	13	26.78	27.73	3.56	20.73	21.11	1.84	13.09	13.17	0.60
	17	20.91	21.75	4.03	14.34	14.74	2.78	7.83	7.90	0.84
	20	15.55	16.33	5.06	9.61	9.96	3.67	3.33	3.40	1.98
	Avg.			3.58			2.45			0.90
150	10	130.78	131.18	0.30	80.13	80.54	0.50	65.79	66.04	0.39
	13	76.29	77.44	1.51	63.59	64.26	1.05	49.16	49.51	0.70
	17	61.22	62.96	2.85	47.92	48.72	1.66	31.24	31.58	1.07
	20	52.39	54.24	3.54	38.32	39.12	2.07	23.01	23.41	1.72
	Avg.			2.05			1.32			0.97
200	10	119.55	121.46	1.60	105.27	105.97	0.67	90.91	90.92	0.01
	13	99.77	102.78	3.02	87.00	87.76	0.87	69.91	69.92	0.02
	17	85.34	88.36	3.53	70.20	70.96	1.08	49.24	49.26	0.05
	20	73.34	76.36	4.11	55.80	56.57	1.38	36.97	37.09	0.32
	Avg.			3.06			1.00			0.10

Table 2: Average CPU time on tests solved within the time limit by both formulations of Sect. 3 and average relative optimality gap on tests unsolved by both formulation upon reaching the time limit.

		k	Total	Avg. CPU Time(s)		Avg. Gap(%)	
				$(\mathcal{D})$	$(\mathcal{L})$	$(\mathcal{D})$	$(\mathcal{L})$
Vertical outliers	Solved by both	100	54	488.11	565.93	-	
		150	39	623.90	726.79	-	
		200	16	628.34	739.63	-	
	Unsolved by both	100	57	-		26.16	23.80
		150	79	-		21.37	21.52
		200	102	-		22.58	23.57
Bad leverage points	Solved by both	100	67	396.08	438.04	-	
		150	36	326.06	484.22	-	
		200	34	405.55	526.04	-	
	Unsolved by both	100	48	-		24.59	24.95
		150	81	-		19.80	20.97
		200	85	-		22.30	22.84

report (Barbato and Ceselli, 2022b). In the following we give a synthetic report of the obtained results.

In Table 2 we present a quantitative overview of the CPU time and of the relative optimality gap obtained by $(\mathcal{D})$ and $(\mathcal{L})$. The table contains two horizontal blocks; each refers to the corruption scheme indicated in the first column. Each block is split in two parts: one containing the average CPU time on the runs in which both algorithms proved optimality within the time limit, the other containing the average optimality gap on the runs in which neither algorithm terminates before the time limit. There remain just 22 runs over 720, in which only one algorithm terminates; roughly half of them is solved by each algorithm.

First, from Table 2 we observe that, independently of the corruption scheme, the instances become more difficult to solve to optimality when the size of the dataset increases. This is not

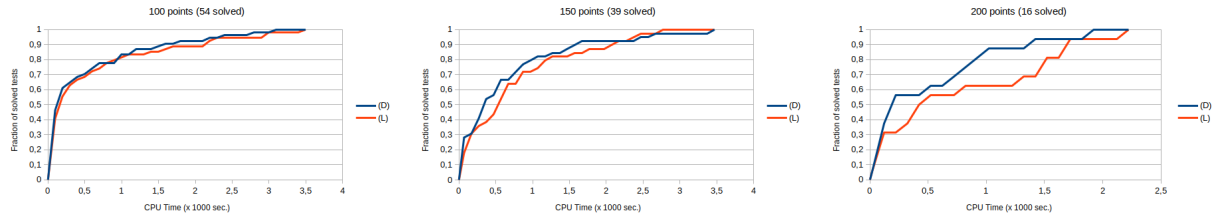

Figure 2: Performance profiles of MILPs ($\mathcal{D}$) and ($\mathcal{L}$): fraction of solved instances within a given CPU time. (LAD-SFSOD tests with vertical outliers.)

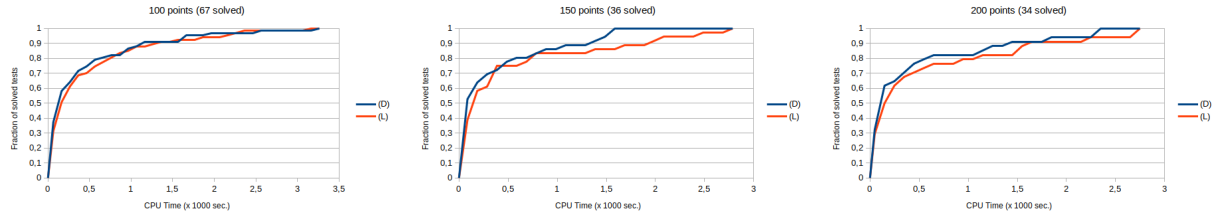

Figure 3: Performance profiles of MILPs ($\mathcal{D}$) and ($\mathcal{L}$): fraction of solved instances within a given CPU time. (LAD-SFSOD tests with bad leverage points.)

surprising because k_0 depends proportionally on k, thus larger instances correspond to more points selectable as outliers, that is, to more combinatorial decisions to be determined. We mention that instances tend to get harder also for a fixed k, when k_0 increases.

A more important outcome is summarized in the following

Experimental Observation 2. *Independently of the corruption scheme, algorithm ($\mathcal{D}$) is more effective than algorithm ($\mathcal{L}$).*

The latter observation is clear from the comparison of the average CPU times obtained by ($\mathcal{D}$) and ($\mathcal{L}$) on the *affordable* instances, which are those solved by both algorithms. It actually corresponds to structurally better performances of ($\mathcal{D}$).

To better display such computational advantage of ($\mathcal{D}$) over ($\mathcal{L}$), in Figure 2 (resp. Figure 3) we present performance profiles for the case of vertical outliers (resp. bad leverage points). These are constructed as follows: for a given value $\bar{t}$ on the x-axis we consider the fraction of runs solved by the corresponding algorithm within $\bar{t}$ seconds, so that higher profiles correspond to better performance (more precisely, to more runs solved within a given time limit). We see that, while for 100 points datasets the dominance of ($\mathcal{D}$) is marginal, such a dominance becomes more clear as the number of points increases to 150 and 200.

Another interesting conclusion that stems from our experiments is the following:

Experimental Observation 3. *Instances with vertical outliers appear harder to solve to optimality than instances with bad leverage points, for both algorithms.*

In detail, affordable instances are less in the vertical outliers case than in the bad leverage points case. Moreover, they are more time-consuming, independently of the algorithm. We explain this behaviour by noting that the objective of (LAD-SFSOD) penalizes the residuals, which are higher in the case of corrupted explanatory variables *and* response values. This in turn makes each binary decision choosing a specific point as outlier to have a stronger impact in the definition of an optimal solution, ultimately helping MIP solvers to perform better.

To better understand this phenomenon, we performed a similar experiment using the 2-norm in the objective (LS-SFSOD setting). Detailed results can be found in the companion technical report (Barbato and Ceselli, 2022b). Our main finding is the following.

Experimental Observation 4. *Within the LS-SFSOD setting, instances affected by bad leverage points are more difficult than those with vertical outliers.*

That is, a reverse phenomenon is observed. Our explanation is the following. Due to the Cauchy-Schwarz inequality, the 2-norm leads to smaller penalization of residuals than the 1-norm of the LAD setting. Therefore, the effect for binary decisions concerning outliers does not happen at the same scale when the 2-norm is used, leaving to the MIP solver a higher computational burden for exploring potentially good solutions. Summarizing, this corroborates our hypothesis that the 1-norm in the objective tends to detect bad leverage points more easily than vertical outliers, thanks to higher penalization of the residuals.

4.3. Comparison with built-in functionalities of MIP solvers

In this section we compare algorithm ($\mathcal{D}$) with two algorithms for the LAD-SFSOD based on the built-in functionalities of CPLEX 20.1 and Gurobi 9.5.2. More precisely we have exploited:

- indicator and SOS-1 constraints of CPLEX 20.1 to model respectively the bilinear objective function and the 0-norm constraints of the LAD-SFSOD;
- native support for 0-norm constraints and bilinear objective functions of the LAD-SFSOD.

This allows us to avoid the usage of big-M values in implementing LAD-SFSOD with those solvers. The comparison is conducted on the testing datasets with $k = 100$ and affected by vertical outliers, using all combinations of $d_0 \in \{5, 6, \ldots, 10\}$ and $k_0 \in \{10\%, 13\%, 17\%, 20\%\}$.

Results are provided in Table 3: there we report for each tested algorithm the relative optimality gap in percentage and the CPU time needed to reach optimality. Results are averaged over the 5 replications of each testing dataset. Boldface indicates the best values of relative optimality gap and CPU time for each tested combination of d_0 and k_0. We see that Gurobi 9.5.2 does not solve any instance to optimality. Moreover, while it is able to find incumbents for each instance it always reports a primal bound of 0.0, thus explaining the 100% of gap obtained at each run. The algorithm based on the SOS-1 and the indicator constraints of CPLEX is more effective than Gurobi 9.5.2 but it is outperformed by ($\mathcal{D}$), as it is apparent by looking at the boldface values: the disjunctive-based formulation always gets smaller CPU times and reaches optimality in runs corresponding to $d_0 = 6$

and $k_0 = 20\%$, to $d_0 = 7$ and $k_0 = 17\%$, to $d_0 = 8, 9$ and $k_0 = 13\%$ and to $k_0 = 10$ and $d_0 = 10\%$ which are always unsolved by CPLEX 20.1.0; moreover, it obtains smaller relative optimality gaps with only one exception ($d_0 = 8$ and $k_0 = 10\%$). The last line of Table 3 reports the total number of runs (out of 120) in which each algorithm reached optimality; algorithm ($\mathcal{D}$) reaches optimality in 58 runs, while the CPLEX-based version in 42 runs.

Table 3: Comparison of ($\mathcal{D}$) with CPLEX and Gurobi built-in functionalities. Instances with $k = 100$ and vertical outliers. Results averaged over 5 replications.

		($\mathcal{D}$)		CPLEX 20.1		Gurobi 9.5.2	
d_0	k_0	Gap (%)	CPU Time (sec.)	Gap (%)	CPU Time (sec.)	Gap (%)	CPU Time (sec.)
5	0.10	0.00	**4.65**	0.00	22.06	100	3600.00
	0.13	0.01	**99.35**	0.01	155.70	100	3600.00
	0.17	0.01	**879.43**	0.28	1026.65	100	3600.00
	0.20	**1.90**	**1936.30**	13.37	2165.98	100	3600.00
6	0.10	0.00	**29.29**	0.01	47.76	100	3600.00
	0.13	**0.01**	**1104.33**	5.20	2057.02	100	3600.00
	0.17	**3.87**	**2391.91**	12.47	2644.80	100	3600.00
	0.20	**10.18**	**3161.23**	16.33	3600.00	100	3600.00
7	0.10	0.01	**155.95**	0.01	539.82	100	3600.00
	0.13	**2.08**	**1724.14**	10.12	2577.00	100	3600.00
	0.17	**14.08**	**3069.84**	24.04	3600.00	100	3600.00
	0.20	**22.69**	3600.00	51.30	3600.00	100	3600.00
8	0.10	0.59	**838.06**	**0.01**	2361.68	100	3600.00
	0.13	**6.67**	**2579.57**	20.54	3600.00	100	3600.00
	0.17	**19.25**	3600.00	35.81	3600.00	100	3600.00
	0.20	**35.82**	3600.00	46.75	3600.00	100	3600.00
9	0.10	**1.32**	**1012.75**	5.28	2279.11	100	3600.00
	0.13	**12.16**	**3188.53**	24.44	3600.00	100	3600.00
	0.17	**30.57**	3600.00	40.29	3600.00	100	3600.00
	0.20	**42.38**	3600.00	49.35	3600.00	100	3600.00
10	0.10	**1.85**	**1403.76**	17.19	3600.00	100	3600.00
	0.13	**19.20**	3600.00	28.19	3600.00	100	3600.00
	0.17	**34.62**	3600.00	43.06	3600.00	100	3600.00
	0.20	**47.17**	3600.00	55.78	3600.00	100	3600.00
Solved/ Tot. Runs			58/120		42/120		0/120

In summary algorithm ($\mathcal{D}$) consistently outperforms the versions based on built-in functionalities of both CPLEX and Gurobi. This latter reports the worse performance with no instance solved to optimality and an optimality gap of 100%, due to its inability of obtaining positive lower bounds; CPLEX-based algorithm seldom obtains optimal solutions within the time limit of 3600 seconds, and always for small values of d_0 and k_0. Finally, ($\mathcal{D}$) reports smaller optimality gaps on runs not reaching optimality.

4.4. Comparison of Prediction Quality

In this section we study the quality of the solutions produced by solving (LAD-SFSOD) (and the LS-SFSOD) in previous sections. We follow to a large extent the evaluation setting proposed in Insolia et al. (2021). We compare the solutions obtained from formulations of Sect. 3 with those produced by the `R` packages `enetLTS` (v. 1.1.0) and `sparseLTS` (v. 0.7.2) mentioned in Sect. 2 (the complete parameter specification of these methods is provided in Appendix B). Both are much

faster than MIP approaches (see Appendix B for details). In the comparison we denote by $(\mathcal{D})_2$ and $(\mathcal{L})_2$ the methods respectively based on formulations $(\mathcal{D})$ and $(\mathcal{L})$ adapted for the LS setting. The comparison is made evaluating the prediction of the solutions over the testing dataset which, we recall, is not corrupted. Let $p^1 = (a^1, r_1), p^2 = (a^2, r_2), \ldots, p^k = (a^k, r_k)$ be the points in a testing dataset and let $(\hat{\omega}, \hat{\zeta})$ be a solution obtained from one of the above-mentioned methods. The first measures we define are:

- *rooted mean squared prediction error* (RMSPE) defined by $\text{RMSPE} = \left(\frac{\sum_{i=1}^{k}(r_i - \hat{\omega}a^i - \hat{\zeta})^2}{k}\right)^{1/2}$
- *mean absolute error* (MAE) defined by $\text{MAE} = \frac{\sum_{i=1}^{k}|r_i - \hat{\omega}a^i - \hat{\zeta}|}{k}$
- *false positive* (FP) and *false negative* (FN) ratios for the feature selection defined by $\text{FP} = \frac{|\{j=1,2,\ldots,d:\ \hat{\omega}\neq 0 \text{ and } \omega = 0\}|}{|\{j=1,2,\ldots,d:\ \omega = 0\}|}$ and $\text{FN} = \frac{|\{j=1,2,\ldots,d:\ \hat{\omega}= 0 \text{ and } \omega\neq 0\}|}{|\{j=1,2,\ldots,d:\ \omega \neq 0\}|}$ where ω is the authentic vector of hyperplane coefficients.
- *F1-score* for the feature selection: $\text{F1} = (1 - \text{FN})/(1 - \text{FN} + (\text{FP} + \text{FN})/2)$

In the tables the measures above are reported for each instance after averaging their values over the corresponding 5 replications. Let also $(\hat{\omega}^1, \hat{\zeta}^1), \ldots, (\hat{\omega}^5, \hat{\zeta}^5)$ the solutions obtained on the 5 replications of the same instance. Their average is defined by $(\bar{\omega}, \bar{\zeta}) = \sum_{h=1}^{5}(\hat{\omega}^h, \hat{\zeta}^h)/5$. Given the authentic hyperplane ω, in the tables we additionally report the *mean squared error* (MSE) subdivided in *MSE bias* defined by $\frac{\sum_{j=1}^{d}(\omega_j - \bar{\omega}_j)^2}{d}$ and *MSE variance* defined by $\frac{\sum_{j=1}^{d}\sum_{h=1}^{5}(\hat{\omega}_j^h - \bar{\omega}_j)^2}{5d}$.

Before presenting the results, we recall that our branch-and-bound algorithms for the SFSOD is run on each instance by varying d_0 and k_0 over a grid of parameters, thus producing a grid of solutions; however, we need to select only one of them which is subsequently used to perform the prediction on the testing dataset. We use the same method based on the Bayesian Information Criterion (BIC) presented in Insolia et al. (2021) to select a single solution out of the solution grid generated for each instance.

In Table 4 we present the results in the setting involving vertical outliers. For each quality measure and each dataset size we highlight in boldface the best corresponding result.

Experimental Observation 5. *In presence of vertical outliers,* `sparseLTS` *is best in combining prediction quality and sparsity.*

In detail, we measure prediction quality by the values of "RMSPE" and "MAE"; the performance gap between LTS methods and MIP approaches tends to decrease as the size of the dataset increases: on the largest datasets ($k = 200$ points) all methods report similar prediction errors. Algorithm `enetLTS` reports the best prediction errors as highlighted by the boldface values of Table 4, followed by `sparseLTS` method. On the other hand `enetLTS` yields high false positive ratios on all three types of datasets (28% for $k = 100$, 33% for $k = 150$ and 37% for $k = 150$) while `sparseLTS` and the LAD-SFSOD solutions yield the lowest false positive ratios (always between 4% and 9%). Essentially, the `enetLTS` method does not produce sparse solutions (remember that the authentic vector of hyperplane coefficients has 45 zero entries). This is due to the presence of vertical outliers.

We also point out that the false positive ratio of the `enetLTS` method increases by increasing k (and hence the number of outliers) while it is more stable for LAD- and LS-SFSOD methods and for `sparseLTS`.

An analogous analysis is made in Table 5 for datasets corrupted with bad leverage points.

Experimental Observation 6. *In presence of bad leverage points, LAD-SFSOD is the best method in combining prediction quality and feature selection performance.*

The LAD-SFSOD solutions produced by the formulations of Sect. 3 yield essentially identical results for all quality measures. The best prediction error of LTS methods is at least 1.57 times worse than the best prediction error of the LAD-SFSOD method. The false positive ratio of `enetLTS` method is 6 up to more than 10 times worse than the best LAD-SFOSD method in each instance group; the false positive ratio of `sparseLTS` is comparable to that of LAD- and LS-SFSOD methods but its false negative ratio is larger, thus leading to a worse F1-score, interpretable as a less accurate feature selection. Instead, the false negative ratio of SFSOD methods is always null. Also, the lowest MSE bias and variance is always obtained by LAD- and LS-SFSOD methods.

The comparison between LAD-SFSOD methods and LS-SFSOD methods highlights that in terms of prediction error, the former yield the best solutions, but the latter stay in comparable ranges; however, LS-SFSOD solutions tend to be less sparse with FP being slightly higher than the LAD-SFSOD solutions for all three types of datasets. Finally, we remark that, according to the results reported in companion technical report Barbato and Ceselli (2022b), LAD-SFSOD models yield also faster computations.

Table 4: Prediction quality of the formulations of Sect. 3 in the LAD and LS setting, and comparison with methods from the literature. (Instances with vertical outliers.)

k	Method	RMSPE	MAE	FP	FN	F1	MSE bias	MSE variance
100	LAD-SFSOD ($\mathcal{D}$)	1.25	1.00	**0.09**	0.00	**0.96**	0.00	**0.01**
	LAD-SFSOD ($\mathcal{L}$)	1.28	1.01	0.10	0.00	0.95	0.00	0.02
	LS-SFSOD ($\mathcal{D}$)$_2$	1.26	1.00	0.11	0.00	0.95	0.00	**0.01**
	LS-SFSOD ($\mathcal{L}$)$_2$	1.34	1.06	0.11	0.00	0.95	0.00	0.02
	`enetLTS`	**1.18**	**0.95**	0.28	0.00	0.88	0.00	**0.01**
	`sparseLTS`	1.21	0.97	**0.09**	0.00	**0.96**	0.00	**0.01**
150	LAD-SFSOD ($\mathcal{D}$)	1.11	0.89	0.05	0.00	**0.98**	0.00	0.02
	LAD-SFSOD ($\mathcal{L}$)	1.15	0.92	**0.04**	0.00	**0.98**	0.00	0.02
	LS-SFSOD ($\mathcal{D}$)$_2$	1.16	0.93	0.09	0.00	0.96	0.00	0.02
	LS-SFSOD ($\mathcal{L}$)$_2$	1.18	0.95	0.10	0.00	0.95	0.00	0.02
	`enetLTS`	**1.07**	**0.85**	0.33	0.00	0.86	0.00	0.02
	`sparseLTS`	1.09	0.88	0.05	0.00	**0.98**	0.00	0.02
200	LAD-SFSOD ($\mathcal{D}$)	1.06	**0.85**	**0.06**	0.00	**0.97**	0.00	0.01
	LAD-SFSOD ($\mathcal{L}$)	1.08	0.86	0.08	0.00	0.96	0.00	0.01
	LS-SFSOD ($\mathcal{D}$)$_2$	1.07	0.86	0.10	0.00	0.95	0.00	0.01
	LS-SFSOD ($\mathcal{L}$)$_2$	1.08	0.86	0.09	0.00	0.96	0.00	0.01
	`enetLTS`	**1.05**	**0.85**	0.37	0.00	0.85	0.00	**0.00**
	`sparseLTS`	1.07	0.86	**0.06**	0.00	**0.97**	0.00	**0.00**

Table 5: Prediction quality of the formulations of Sect. 3 in the LAD and LS setting, and comparison with methods from the literature. (Instances with bad leverage points.)

k	Method	RMSPE	MAE	FP	FN	F1	MSE bias	MSE variance
100	LAD-SFSOD ($\mathcal{D}$)	1.21	0.97	0.09	**0.00**	**0.96**	**0.00**	**0.04**
	LAD-SFSOD ($\mathcal{L}$)	**1.20**	**0.96**	0.09	**0.00**	**0.96**	**0.00**	0.05
	LS-SFSOD $(\mathcal{D})_2$	1.22	0.98	0.11	**0.00**	0.95	**0.00**	**0.04**
	LS-SFSOD $(\mathcal{L})_2$	1.24	1.00	0.11	**0.00**	0.95	**0.00**	**0.04**
	`enetLTS`	2.38	1.92	0.53	0.13	0.72	0.06	0.08
	`sparseLTS`	2.26	1.82	**0.08**	0.23	0.81	0.07	0.07
150	LAD-SFSOD ($\mathcal{D}$)	1.15	**0.92**	**0.05**	**0.00**	**0.97**	**0.00**	**0.01**
	LAD-SFSOD ($\mathcal{L}$)	**1.14**	**0.92**	**0.05**	**0.00**	**0.97**	**0.00**	**0.01**
	LS-SFSOD $(\mathcal{D})_2$	1.17	0.94	0.07	**0.00**	0.97	**0.00**	**0.01**
	LS-SFSOD $(\mathcal{L})_2$	1.17	0.94	0.06	**0.00**	0.97	**0.00**	**0.01**
	`enetLTS`	1.83	1.49	0.33	0.03	0.85	0.03	0.04
	`sparseLTS`	1.81	1.48	0.07	0.07	0.93	0.04	0.03
200	LAD-SFSOD ($\mathcal{D}$)	**1.09**	**0.86**	0.06	**0.00**	0.97	**0.00**	**0.02**
	LAD-SFSOD ($\mathcal{L}$)	1.10	0.87	**0.05**	**0.00**	**0.98**	**0.00**	**0.02**
	LS-SFSOD $(\mathcal{D})_2$	1.11	0.88	0.09	**0.00**	0.96	**0.00**	**0.02**
	LS-SFSOD $(\mathcal{L})_2$	1.12	0.88	0.09	**0.00**	0.96	**0.00**	**0.02**
	`enetLTS`	1.84	1.48	0.51	0.03	0.78	0.03	0.05
	`sparseLTS`	1.81	1.45	0.08	0.27	0.77	0.04	0.03

5. Conclusions

Our paper revolves around the idea of using a least absolute deviation (LAD) criterion for the SFSOD, instead of the least-squares (LS) criterion which is more commonly used in the literature.

We rely on a mathematical programming approach: we model and solve the LAD-SFSOD by means of two MILP formulations, one adapted from the literature dealing with the LS-SFSOD and the other one based on a disjunctive argument. Our theoretical analysis proves the disjunctive-based formulation to offer, in terms of polyhedral structure, advantages with respect to MILP adapted from the literature. These yield computational advantages. For instance, the quality of the continuous relaxation of the disjunctive-based formulation is never worse than those of models adapted from the literature, and it becomes strictly better when a proper choice of model parameters can be made.

For what concerns the specific application to the SFSOD, we are able to obtain several insights. First, the performance of LAD and LS methods are strongly affected by the type of outliers in the dataset. When vertical outliers are involved LTS methods provide the best combination of prediction error, sparsity and computational efficiency. However, when bad leverage points pollute the dataset, our experiments show that the mathematical programming approaches using LAD-SFSOD models perform best (and especially the disjunctive-based formulation). In particular, in this case, LTS methods are not able to combine sparsity and robustness satisfactorily.

This leads to another interesting insight: mathematical programming models using least squares appear to be dominated for the SFSOD, being inferior to dedicated LTS methods in presence of

vertical outliers, and inferior to LAD-SFSOD models in presence of bad leverage points.

Finally, we found it interesting to note that our results were obtained by changing the classical approach in computational statistics, in favor of one which is mathematical programming oriented. This highlights the potential impact that Operations Research techniques can still bring in computational statistics. In this spirit, we foresee two directions for future investigation. One is the design of dedicated algorithms for optimizing the LAD-SFSOD MILPs of this paper. To this end decomposition approaches are promising to improve computational performances, as shown in related applications of MILPs to computational statistics (see *e.g.*, Warwicker and Rebennack (2022) for a Benders' decomposition algorithm for the resolution of robust piecewise linear regression based on a previous work of Rebennack and Krasko (2020)). Another direction is to investigate if the application of kernels to the dual of stronger formulations (as the disjunctive-based one) would improve the performance of classic methods.

Data and Code Availability

The instances generated for this study are publicly available at the online repository of Barbato and Ceselli (2022a). The code implemented for this study are available from the corresponding author upon request.

Declarations

Funding. The work was partially funded by Università degli Studi di Milano, Piano Sostegno alla Ricerca (PSR).

Conflict of interest. The authors declare that they have no conflict of interest.

References

Alfons, A. (2021). `robustHD` v. 0.7.2: Robust methods for high-dimensional data. `https://cran.r-project.org/web/packages/robustHD/robustHD.pdf`. Accessed: 2022-31-03.

Alfons, A., Croux, C., and Gelper, S. (2013). Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *The Annals of Applied Statistics*, pages 226–248.

Arslan, O. (2012). Weighted LAD-LASSO method for robust parameter estimation and variable selection in regression. *Computational Statistics & Data Analysis*, 56(6):1952–1965.

Balas, E. (1998). Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3–44.

Barbato, M. and Ceselli, A. (2022a). Replication Data for: "Mathematical Programming for Simultaneous Feature Selection and Outlier Detection under l1 Norm". `https://doi.org/10.13130/RD_UNIMI/1MZNNS`.

Barbato, M. and Ceselli, A. (2022b). Technical Report of "Mathematical Programming for Simultaneous Feature Selection and Outlier Detection under l1 Norm". https://doi.org/10.13130/RD_UNIMI/UA8PFI.

Belotti, P., Bonami, P., Fischetti, M., Lodi, A., Monaci, M., Nogales-Gómez, A., and Salvagnin, D. (2016). On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications*, 65(3):545–566.

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

Bertsimas, D., Kitane, D. L., Azami, N., and Doucet, F. (2020). Novel mixed integer optimization sparse regression approach in chemometrics. *Analytica Chimica Acta*, 1137:115–124.

Bertsimas, D. and Weismantel, R. (2005). *Optimization over integers*, volume 13. Dynamic Ideas, Belmont.

Bonami, P., Lodi, A., Tramontani, A., and Wiese, S. (2015). On mathematical programming with indicator constraints. *Mathematical Programming*, 151(1):191–223.

Bottmer, L., Croux, C., and Wilms, I. (2022). Sparse regression for large data sets with outliers. *European Journal of Operational Research*, 297(2):782–794.

Candes, E. and Tao, T. (2007). The Dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2313–2351.

Chen, Y., Caramanis, C., and Mannor, S. (2013). Robust sparse regression under adversarial corruption. In *International conference on machine learning*, pages 774–782. PMLR.

Dodge, Y. (1997). Lad regression for detecting outliers in response and explanatory variables. *Journal of Multivariate Analysis*, 61(1):144–158.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360.

Filzmoser, P. and Nordhausen, K. (2021). Robust linear regression for high-dimensional data: An overview. *Wiley Interdisciplinary Reviews: Computational Statistics*, 13(4):e1524.

Fischetti, M. and Monaci, M. (2020). A branch-and-cut algorithm for mixed-integer bilinear programming. *European Journal of Operational Research*, 282(2):506–514.

Greenshtein, E. (2006). Best subset selection, persistence in high-dimensional statistical learning and optimization under $l1$ constraint. *The Annals of Statistics*, 34(5):2367–2386.

Gurobi Optimization (2021). Gurobi 9.5 optimizer reference manual. https://www.gurobi.com/documentation/9.5/refman/index.html. Accessed September 28, 2022.

Hijazi, H., Bonami, P., Cornuéjols, G., and Ouorou, A. (2012). Mixed-integer nonlinear programs featuring "on/off" constraints. *Computational Optimization and Applications*, 52(2):537–558.

IBM ILOG (2020). CPLEX Optimization Studio Version 20 Release 1.0. `https://www.ibm.com/docs/en/icos/20.1.0?topic=documentation-introducing-ilog-cplex-optimization-studio-2010`. Accessed August 26, 2022.

Insolia, L., Kenney, A., Chiaromonte, F., and Felici, G. (2021). Simultaneous feature selection and outlier detection with optimality guarantees. *Biometrics*.

Jammal, M. (2020). *Variable selection and outlier detection via mixed integer programming*. PhD thesis, Normandie Université; Université Libanaise.

Jammal, M., Canu, S., and Abdallah, M. (2020). ℓ_1 Regularized Robust and Sparse Linear Modeling Using Discrete Optimization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 645–661. Springer.

Jammal, M., Canu, S., and Abdallah, M. (2021). Joint outlier detection and variable selection using discrete optimization. *SORT-Statistics and Operations Research Transactions*, pages 47–66.

Kurnaz, F. S., Hoffmann, I., and Filzmoser, P. (2018a). Robust and sparse estimation methods for high-dimensional linear and logistic regression. *Chemometrics and Intelligent Laboratory Systems*, 172:211–222.

Kurnaz, F. S., Hoffmann, I., and Filzmoser, P. (2018b). `enetLTS` v. 0.1.1: Robust and sparse methods for high dimensional linear and logistic regression. `https://cran.r-project.org/web/packages/enetLTS/enetLTS.pdf`. Accessed: 2022-31-03.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I – Convex underestimating problems. *Mathematical Programming*, 10(1):147–175.

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

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234.

Rebennack, S. and Krasko, V. (2020). Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing*, 32(2):507–530.

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880.

Rousseeuw, P. J. and Van Driessen, K. (2006). Computing LTS regression for large data sets. *Data Mining and Knowledge Discovery*, 12(1):29–45.

Sudermann-Merx, N. and Rebennack, S. (2021). Leveraged least trimmed absolute deviations. *OR Spectrum*, 43(3):809–834.

Thompson, R. (2022). Robust subset selection. *Computational Statistics & Data Analysis*, 169:107415.

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

Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. *Siam Review*, 57(1):3–57.

Wang, H. and Leng, C. (2007). Unified LASSO estimation by least squares approximation. *Journal of the American Statistical Association*, 102(479):1039–1048.

Wang, H., Li, G., and Jiang, G. (2007). Robust regression shrinkage and consistent variable selection through the LAD-Lasso. *Journal of Business & Economic Statistics*, 25(3):347–355.

Wang, Y. and Zhu, L. (2017). Variable selection and parameter estimation via wlad–scad with a diverging number of parameters. *Journal of the Korean Statistical Society*, 46(3):390–403.

Warwicker, J. A. and Rebennack, S. (2022). Generating optimal robust continuous piecewise linear regression with outliers through combinatorial benders decomposition. *IISE Transactions*, pages 1–13.

Xu, J. and Ying, Z. (2010). Simultaneous estimation and variable selection in median regression using lasso-type penalty. *Annals of the Institute of Statistical Mathematics*, 62(3):487–514.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429.

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.

Appendix A. Relation between the Disjunctive-Based and McCormick's Linearizations

In this section we prove the results of Prop. 2 and Obs. 2 in the main paper. The objective function of (LAD-SFSOD) is $\sum_{i=1}^{k} s_i p_i$ where $s \in \{0,1\}^k$ is the set of switch variables and $\mathbf{0} \leq p \leq R^U$ is the set of variables upper bounding the residuals of an optimal solution. The objective can be linearized by describing $\text{conv}(\{(x_i, p_i, s_i)\colon x_i = p_i s_i,\ s_i \in \{0,1\},\ 0 \leq p_i \leq R_i^U\})$ for every

$i = 1, 2, \ldots, k$ through the linear inequalities provided by McCormick's envelope McCormick (1976). Applying such linearization technique yields the following valid reformulation of (LAD-SFSOD):

$$\min \sum_{i=1}^{k} x_i \tag{A.1}$$

$$x_i \geq p_i + R_i^U s_i - R_i^U \qquad \forall i = 1, 2, \ldots, k \tag{A.2}$$

$$x_i \leq p_i \qquad \forall i = 1, 2, \ldots, k \tag{A.3}$$

$$x_i \leq R_i^U s_i \qquad \forall i = 1, 2, \ldots, k \tag{A.4}$$

$$p_i \geq r_i - a^i w - z \qquad \forall i = 1, 2, \ldots, k \tag{A.5}$$

$$p_i \leq z + a^i w - r_i \qquad \forall i = 1, 2, \ldots, k \tag{A.6}$$

$$w_j \geq W_j^L f_j \qquad \forall j = 1, 2, \ldots, d \tag{A.7}$$

$$w_j \leq W_j^U f_j \qquad \forall j = 1, 2, \ldots, d \tag{A.8}$$

$$\sum_{j=1}^{d} f_j \leq d_0 \tag{A.9}$$

$$\sum_{i=1}^{k} s_i \geq k - k_0 \tag{A.10}$$

$$s \in \{0, 1\}^k \tag{A.11}$$

$$f \in \{0, 1\}^d \tag{A.12}$$

$$p \geq \mathbf{0}, w \in \mathbb{R}^d, z \in \mathbb{R} \tag{A.13}$$

$$x \geq \mathbf{0}. \tag{A.14}$$

For every vector $R^U \geq \mathbf{0}$ (also not valid) we denote by $\mathcal{M}_{R^U}$ and $\mathcal{D}_{R^U}$ the polytopes describing the domain of the linear relaxations of (A.1)–(A.14) and of the disjunctive-based formulation, respectively; then $\mathcal{D}_{R^U}$ corresponds to a face of $\mathcal{M}_{R^U}$, as stated in Prop. 2. Indeed, it is immediate that $\mathcal{D}_{R^U}$ is obtained from the linear relaxation of (A.1)–(A.14) by setting all constraints (A.2) to equality. In order to prove Obs. 2 we first observe that every optimal solution $(p^\star, f^\star, s^\star, w^\star, z^\star)$ to the linear relaxation of the disjunctive-based formulation admits a solution $(x^\star, p^\star, f^\star, s^\star, w^\star, z^\star)$ of (A.1)–(A.14) of the same value, just by defining $x_i^\star := p_i^\star - (1 - s_i^\star) R_i^U$. Then, denoting by $v(\mathcal{M}_{R^U})$ and $v(\mathcal{D}_{R^U})$ the optimal values of the linear relaxations of $(\mathcal{D})$ and of (A.1)–(A.14) we obtain that $v(\mathcal{D}_{R^U}) \geq v(\mathcal{M}_{R^U})$. Conversely, let $(x^\star, p^\star, f^\star, s^\star, w^\star, z^\star)$ a point in $\mathcal{M}_{R^U}$ of value $v(\mathcal{M}_{R^U})$. We observe that the point $(x^\star, \bar{p}, f^\star, s^\star, w^\star, z^\star)$, defined by $\bar{p}_i = R_i^U (1 - s_i^\star)$ if $p_i^\star + R_i^U s_i^\star - R_i^U < 0$ and $\bar{p}_i = p_i^\star$ otherwise, is also an optimal solution. Moreover, this latter solution satisfies (A.2) with equality, so that $(\bar{p}, f^\star, s^\star, w^\star, z^\star)$ belongs to $\mathcal{D}_{R^U}$, thus proving $v(\mathcal{D}_{R^U}) \leq v(\mathcal{M}_{R^U})$.

Appendix B. Details for `LTS` methods

We provide a list of non-default values for the parameters of algorithms `enetLTS` and `sparseLTS` used in Sect. 4.4.

Algorithm `enetLTS`*.*

- `hsize` (proportion of points labelled as authentic in the final solution; default 0.75): $1 - p_0$ where $p_0 \in \{0.1, 0.13, 0.17, 0.2\}$. Note that these choices of p_0 correspond to the values of k_0 used to test the MI(L)P approach;
- `nsamp` (number of initial subsamples on which C-steps are performed; default 500, see Kurnaz et al. (2018b,a) for details on the C-steps of `enetLTS`): 1000;
- `nfold` (fold number for k-cross validation; default 5): 10.

Except for `hsize`, the other non-default values above were also used in Insolia et al. (2021). Moreover, we specify that we evaluate the `enetLTS` solution obtained from the reweighted fit (see Kurnaz et al. (2018a) for details).

Algorithm `sparseLTS`*..*

- `mode` (types of penalty term; default ["lambda", "fractional"], see Alfons (2021); Alfons et al. (2013) for details): "fractional";
- `lambda` (penalty term; default 0.05): 0.15 for evaluation on datasets affected by vertical outliers and 0.5 for evaluation on datasets affected by bad leverage points;
- `alpha` (proportion of authentic points in the final solution; default 0.75): $1 - p_0$ where $p_0 \in \{0.1, 0.13, 0.17, 0.2\}$. Note that these choices of p_0 correspond to the values of k_0 used to test the MI(L)P approach;
- `nsamp` (subsamples to use at the beginning and to keep after the C-steps; default $[500, 10]$): $[1000, 20]$;
- `crit` (criterion to select the solution; default ["BIC", "PE"]): "BIC", *i.e.*, a Bayesian Information Criterion.

As for `enetLTS`, we evaluate the solution obtained from the reweighted fit of `sparseLTS`. We point out that the value `lambda` of the penalty term is relevant to obtain solutions of good quality. The values listed above have been found after searching for a good tradeoff between sparsity and robustness. As reported in Sect. 4.4, those choices yield very good results when vertical outliers are involved, but poor results (in terms of both prediction error and F1 score when compared to the LAD-SFSOD solutions) when bad leverage points are present in the dataset. In our experience, lowering the value of `lambda` in the evaluation of `sparseLTS` on datasets affected by bad leverage points may improve the prediction quality but produces much denser hyperplanes (thus worsening the F1 score), while increasing its value has the opposite effect.

Computational times. As explained in Sect. 4.4, the MI(L)P solutions used to evaluate the prediction quality have been generated by solving the corresponding models with CPLEX, by imposing a time limit of 3600 CPU seconds in the LAD setting and of 5400 CPU seconds in the LS setting. We have used the machine indicated at paragraph "Implementation details" of Sect. 4 which is equipped with a 4.10 GHz processor. Although in several cases the corresponding MI(L)Ps are solved in a shorter time, a considerable amount of tests reached the time limit without having proved optimality of the best incumbent. As a consequence, the MI(L)P approach turned out to be very time consuming: even running tests in parallel at least 3 days of computation were needed to complete an entire test session.

On the contrary, `enetLTS` and `sparseLTS` algorithms are much faster: they have been run over a desktop machine running under Ubuntu 18.04 and equipped with 4 GB of RAM and with a 1.70 GHz CPU (model Intel(R) Core(TM) i5-3317U) and have completed always within 1 hour (`enetLTS`) or within seconds (`sparseLTS`) the entire test session.