Minimization of Akaike’s information criterion in linear regression analysis via mixed integer nonlinear program

Keiji Kimura\textsuperscript{a} and Hayato Waki\textsuperscript{b}

\textsuperscript{a}Graduate School of Mathematics, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan; \textsuperscript{b}Institute of Mathematics for Industry, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan

(Received 12 August 2016; accepted 8 May 2017)

Akaike’s information criterion (AIC) is a measure of evaluating statistical models for a given data set. We can determine the best statistical model for a particular data set by finding the model with the smallest AIC value. Since there are exponentially many candidates of the best model, the computation of the AIC values for all the models is impractical. Instead, stepwise methods, which are local search algorithms, are commonly used to find a better statistical model, though it may not be the best model. We propose a branch-and-bound search algorithm for a mixed integer nonlinear programming formulation of the AIC minimization presented by Miyashiro and Takano [Mixed integer second-order cone programming formulations for variable selection, Eur. J. Oper. Res. 247 (2015), pp. 721–731]. More concretely, we propose procedures to find lower and upper bounds, and branching rules for this minimization. We then combine such procedures and branching rules with SCIP, a mathematical optimization software and the branch-and-bound framework. We show that the proposed method can provide the best AIC-based statistical model for small- or medium-sized benchmark data sets in the UCI Machine Learning Repository. Furthermore, the proposed method finds high-quality solutions for large-sized benchmark data sets.

Keywords: Mixed integer nonlinear program; branch-and-bound; SCIP and Akaike’s information criterion

1. Introduction

Selecting the best statistical model from a number of possible statistical models for a given data set is one of the most important problems in statistical applications, e.g. regression analysis. This is called variable selection. The purposes of variable selection are to provide the simplest statistical model for a given data set, and to improve the prediction performance, while keeping the goodness of fit for the given data set. See [8] for more details on variable selection.

In variable selection based on an information criterion, all the candidates are evaluated by the information criterion. We select the best statistical model by using those evaluations. Akaike’s information criterion (AIC) is one of the information criteria and proposed in [3]. The AIC value is computed for each candidate, and the model with the smallest AIC value is selected as the best statistical model. Since we often handle exponentially many candidates for the best model in practical applications, the global minimization based on AIC is not practical. Instead, stepwise methods, which are local search algorithms, are commonly used to find a statistical model which has as small AIC value as possible. However, stepwise methods may not be the smallest.
The contribution of our study is to propose a branch-and-bound search algorithm for a mixed integer nonlinear programming (MINLP) formulation of the minimization of AIC for linear regression [12]. We propose procedures to find lower and upper bounds of the MINLP problems, and define branching rules for efficient computation. In addition, we provide implementation to solve the MINLP problems efficiently via SCIP [2,13,17], which is mathematical optimization software and a branch-and-bound framework. SCIP has high flexibility of user plugin and control on various parameters in the branch-and-bound framework for efficient computation. We also propose efficient computation for data sets with linear dependence. By applying the proposed method to such benchmark data sets in [16], we can obtain the best statistical models. Our implementation is available in [18].

We introduce some related work. Miyashiro and Takano [12] propose an MISOCP formulation for variable selection based on some information criteria in linear regression. Bertsimas and Shioda [6] and Bertsimas, King and Mazumder [5] provide a mixed integer quadratic programming (MIQP) formulation for linear regression with a cardinality constraint. The minimization of AIC can be divided into some MIQP problems by fixing the number of explanatory variables. These MIQP problems are formulated as their formulation. We compare the proposed method with the MIQP and MISOCP formulations, and observe that the proposed method outperforms MIQP and MISOCP formulations.

The organization of this article is as follows: We give a brief introduction of linear regression based on AIC in Section 2. We introduce the MINLP formulation of the AIC minimization and procedures to find lower and upper bounds used in the branch-and-bound framework in Section 3. Section 4 introduces techniques for more efficient computation, e.g. branching rules and treatment on data with linear dependence. We present numerical results in Section 5. In particular, we show the numerical comparison with the MISOCP and MIQP formulations. In addition, we present numerical performances of the branching rules proposed in Section 4.4. We discuss future work of proposed method in Section 6. This article is an extended version of work published in [10].

2. Preliminary on AIC in linear regression

We explain how to select the best statistical model via AIC in linear regression analysis. This analysis is the fundamental statistical tool which determines coefficients $\beta_0, \ldots, \beta_p \in \mathbb{R}$ for the following equation from a given data set:

$$y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j.$$  (1)

Here $x_1, \ldots, x_p$ and $y$ are called the explanatory variables and the response variable, respectively. In fact, we adopt coefficients $\beta_0, \ldots, \beta_p$ which minimize $\sum_{i=1}^{n} \epsilon_i^2$ for a given data set $(x_{i1}, \ldots, x_{ip}, y_i) \in \mathbb{R}^p \times \mathbb{R} \ (i = 1, \ldots, n)$, where $\epsilon_i$ is the $i$th residual and defined by $\epsilon_i = y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij}$.

Variable selection in linear regression analysis is the problem to select the best subset of explanatory variables based on a given criterion. In statistical applications, a preferred model keeps the goodness of fit for a given data set, and contains as few unnecessary explanatory variables as possible. In fact, those explanatory variables may add noise to the prediction based on the statistical model. As a result, the prediction performance of the model may get worse. In addition, we need to observe and/or monitor more data for unnecessary explanatory variables, and thus will spend more cost.

AIC is one of the criteria for variable selection and proposed in [3]. It is used as the measure to select the preferred statistical model in all candidates. The statistical model with the smallest AIC
value is expected as the preferred statistical model. In linear regression analysis, this selection corresponds to the selection of a subset of the set of the explanatory variables in (1) via AIC. More precisely, for a set \( S \subseteq \{1, \ldots, p\} \) of candidates of the explanatory variables in the statistical model (1), the AIC value is defined in [3] as follows:

\[
AIC(S) := -2 \max_{\beta, \sigma^2} \{ \ell(\beta, \sigma^2) : \beta_j = 0 (j \in \{1, \ldots, p\} \setminus S) \} + 2(#(S) + 2),
\]

where \( \beta = (\beta_0, \ldots, \beta_p) \in \mathbb{R}^{p+1} \), #(S) stands for the number of elements in the set \( S \), and \( \ell(\beta, \sigma^2) \) is the log-likelihood function. Computing AIC values for all subsets \( S \) of the explanatory variables in (1), we can obtain the best AIC-based subset. However, the number of subsets is \( 2^p \) and the computation of all subsets is impractical.

We simplify (2). Under the assumption that all the residual \( \epsilon_i \) are independent and normally distributed with the zero mean and variance \( \sigma^2 \), the log-likelihood function can be formulated as

\[
\ell(\beta, \sigma^2) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \epsilon_i^2.
\]

We focus on the first term in (2). Let \( S \) be a set of candidates of the explanatory variables in (1). By substituting \( \beta_j = 0 (j \in \{1, \ldots, p\} \setminus S) \) to the objective function, the first term can be regarded as the unconstrained minimization. Thus minimum solutions satisfy the following equation:

\[
\frac{d\ell}{d(\sigma^2)} = -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2} \sum_{i=1}^{n} \epsilon_i^2 = 0.
\]

From this equation, we obtain \( \sigma^2 = (1/n) \sum_{i=1}^{n} \epsilon_i^2 \). Substituting this equation to (2), we simplify (2) as follows:

\[
AIC(S) = \min_{\beta_j} \left\{ n \log \left( \sum_{i=1}^{n} \epsilon_i^2 \right) : \beta_j = 0 (j \in \{1, \ldots, p\} \setminus S) \right\}
+ 2(#(S) + 2) + n(\log(2\pi/n) + 1).
\]

We use (3) to provide an MINLP formulation of the minimization of AIC in the next section. We remark that \( AIC(S) = -\infty \) if and only if all \( \epsilon_i \) for \( i = 1, \ldots, n \) are zero. Thus we assume throughout this manuscript that there does not exist \( \beta \in \mathbb{R}^p \) such that \( \epsilon_i = 0 \) for all \( i = 1, \ldots, n \). Moreover, the first term of (3) is equivalent to an unconstrained optimization problem with a convex quadratic objective function. It follows from [7, Section 9.1.1] that (3) has a finite optimal value and an optimal solution for all \( S \).

3. MINLP formulation for the minimization of AIC

We provide the minimization of \( AIC(S) \) over \( S \subseteq \{1, \ldots, p\} \) by the following MINLP formulation:

\[
\min_{\beta_j, z_j, \epsilon_i, k} \left\{ n \log \left( \sum_{i=1}^{n} \epsilon_i^2 \right) + 2k : \begin{array}{l}
\epsilon_i = y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \quad (i = 1, \ldots, n), \\
\sum_{j=1}^{p} z_j = k, \beta_0, \beta_j \in \mathbb{R} \quad (j = 1, \ldots, p), \\
z_j \in \{0, 1\}, z_j = 0 \Rightarrow \beta_j = 0 \quad (j = 1, \ldots, p)
\end{array} \right\}.
\]
Here the last constraints represent the logical relations, i.e. $\beta_j$ has to be zero if $z_j = 0$. This formulation is provided in [12, eq. (22)–(25)].

Next we propose a procedure to find a lower bound of the optimal value of a subproblem of (4) at each node in the branch-and-bound tree. Some variables $z_j$ in (4) are fixed to zero or one at each node of the tree. We define the sets $Z_0$, $Z_1$ and $Z$ for a given node as follows:

$$
Z_1 = \{ j \in \{ 1, \ldots, p \} : z_j \text{ is fixed to 1} \},
$$

$$
Z_0 = \{ j \in \{ 1, \ldots, p \} : z_j \text{ is fixed to 0} \},
$$

$$
Z = \{ j \in \{ 1, \ldots, p \} : z_j \text{ is not fixed} \}.
$$

We can specify a node in the branch-and-bound search tree by $Z_1, Z_0$ and $Z$. We denote the node by $V(Z_1, Z_0, Z)$. Then the subproblem at the node $V(Z_1, Z_0, Z)$ is formulated as follows:

$$
\begin{align*}
\min_{\beta_j, z_j} & \quad n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) \right)^2 + 2 \sum_{j=1}^{p} z_j \\
\text{subject to} & \quad z_j = 1 \ (j \in Z_1), \ z_j = 0 \ (j \in Z_0), \ z_j \in \{ 0, 1 \} \ (j \in Z), \\
& \quad \beta_0, \beta_j \in \mathbb{R} \ (j = 1, \ldots, p), \beta_j = 0 \ (j \in Z_0) \ z_j = 0 \Rightarrow \beta_j = 0 \ (j \in Z).
\end{align*}
$$

(5)

By relaxing the integrality of the variables $z_j$ in (5), we obtain the following relaxation problem:

$$
\begin{align*}
\min_{\beta_j, z_j} & \quad n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) \right)^2 + 2 \sum_{j=1}^{p} z_j \\
\text{subject to} & \quad z_j = 1 \ (j \in Z_1), \ z_j = 0 \ (j \in Z_0), \ 0 \leq z_j \leq 1 \ (j \in Z), \\
& \quad \beta_0, \beta_j \in \mathbb{R} \ (j = 1, \ldots, p), \beta_j = 0 \ (j \in Z_0) \ z_j = 0 \Rightarrow \beta_j = 0 \ (j \in Z).
\end{align*}
$$

(6)

Moreover we eliminate all the logical relations and all the variables $z_j$ from (6):

$$
\min_{\beta_j} \left\{ n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) \right)^2 + 2\#(Z_1) : \beta_0, \beta_j \in \mathbb{R} \ (j \in Z \cup Z_1), \beta_j = 0 \ (j \in Z_0) \right\}. 
$$

(7)

The optimal value of (7) is less than or equal to the optimal value of (6). In fact, any feasible solution $(\beta, z)$ of (6) is also feasible for (7), and satisfies $\sum_{j=1}^{p} z_j = \sum_{j \in Z} z_j + \#(Z_1) \geq \#(Z_1)$. The following lemma ensures that both values are the same. Thus we can deal with (7) instead of (6) as the relaxation problem of (5).

**Lemma 3.1** The optimal value of (7) is the same as the optimal value of (6).

**Proof** For an optimal solution $\beta^*$ of (7), we construct a sequence $((\beta^N, z^N))_{N=1}^\infty$ as follows:

$$
\beta^N_j = \begin{cases} 
1 & \text{if } j \in Z_1, \\
1/N & \text{if } j \in Z \text{ and } \beta^N_j \neq 0, \\
0 & \text{if } j \in Z \text{ and } \beta^N_j = 0, \\
0 & \text{if } j \in Z_0,
\end{cases}
$$

for all $N \geq 1$. Clearly, $(\beta^N, z^N)$ is feasible for (6) for all $N \geq 1$. It is sufficient to prove that the objective value $\theta^N$ of (6) at $(\beta^N, z^N)$ converges to the optimal value $\theta^*$ of (7) as $N$ goes to $\infty$. 

Since we have
\[ \theta^* \leq \theta^N = n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0^* - \sum_{j=1}^{p} \beta_j^* x_{ij} \right)^2 \right) + 2\#(Z_1) + \frac{2}{N} \#(Z) = \theta^* + \frac{2}{N} \#(Z), \]
the right-hand side converges to \( \theta^* \) as \( N \) goes to \( \infty \). This implies that the optimal value of (7) is the same as the optimal value of (6).

Although the objective function of (7) contains the logarithm function, we can freely remove the constant \( 2\#(Z_1) \) and the logarithm by the monotonicity of the logarithm function in (7). Thus the following problem is obtained from (7):
\[
\min_{\beta_i} \left\{ \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 : \beta_0, \beta_j \in \mathbb{R} \quad (j \in Z \cup Z_1) \quad \beta_j = 0 \quad (j \in Z_0) \right\}.
\]
(8)

Since (8) is the unconstrained minimization of a quadratic function, we can obtain an optimal solution of (8) by solving a system. In our implementation, we call \texttt{dpsov}, which is a built-in function of LAPACK [4] for solving a linear system. We denote the optimal value of (8) by \( \xi^* \). The optimal value of (7) is \( n \log(\xi^*) + 2\#(Z_1) \), which is used as a lower bound of the optimal value of (5).

We provide a procedure that constructs a feasible solution of (4) and computes an upper bound of the optimal value of (4). For this we use an optimal solution \( \tilde{\beta} \in \mathbb{R}^{p+1} \) obtained by solving (8). We define
\[
\tilde{z}_j = \begin{cases} 
1 & \text{(if } j \in \tilde{Z} \cup Z_1), \\
0 & \text{(otherwise)} 
\end{cases} \quad (j = 1, \ldots, p), \quad \tilde{\epsilon}_i = y_i - \tilde{\beta}_0 - \sum_{j=1}^{p} \tilde{\beta}_j x_{ij} (i = 1, \ldots, n) \quad \text{and} \quad \tilde{k} = \sum_{j=1}^{p} \tilde{z}_j,
\]
where \( \tilde{Z} = \{ j \in Z : \tilde{\beta}_j \neq 0 \} \). It is easy to see that \((\tilde{\beta}_j, \tilde{z}_j, \tilde{\epsilon}_i, \tilde{k})\) is feasible for (4) and that the objective value is \( n \log(\xi^*) + 2\#(\tilde{Z} \cup Z_1) \). If the objective value is smaller than the current best upper bound, we update it.

Finally, we give another understanding for our proposed formulation and propose an efficient computation based on this understanding.

- Since we can regard (8) as linear regression whose explanatory variables are in \( Z_1 \cup Z \), the computation of the lower bound from (8) corresponds to the computation of the value \( \text{AIC}(Z_1 \cup Z) - 2\#(Z) \). On the other hand, the upper bound corresponds to the AIC value of the statistical model whose explanatory variables are in \( Z_1 \cup Z \), i.e. \( \text{AIC}(Z_1 \cup Z) \). Therefore, our proposed method computes the AIC value of the statistical model with \( Z_1 \cup Z \) at each node \( V(Z_1, Z_0, Z) \), up to constant term \( 4 + n(\log(2n\pi) + 1) \) of (3).
- The statistical package \texttt{leaps} [11] in \( \mathbb{R} \) [14] adopts the branch-and-bound scheme in a similar manner. A QR decomposition is exploited at each node in the branch-and-bound search tree. In particular, \texttt{leaps} solves linear systems effectively by using the QR decomposition obtained at its parent node.\texttt{leaps} finds the best statistical model much faster than our proposed method for data sets with \( p \leq 32 \) and no linear dependence. If the data set has linear dependence, \texttt{leaps} does not work effectively. In contrast, our proposed method works more efficiently by using the linear dependence in data sets. This technique will be discussed in Section 4.2.
- We provide efficient computation of lower and upper bounds based on this understanding. We assume that the lower and upper bounds are obtained at a node \( V(Z_1, Z_0, Z) \). Then we do not need to solve (8) at its child node \( V(Z_1 \cup \{ j \}, Z_0, Z \setminus \{ j \}) \), where \( j \in Z \). This node is generated

\[
\begin{align*}
\log \left( \sum_{i=1}^{n} \left( y_i - \beta_0^* - \sum_{j=1}^{p} \beta_j^* x_{ij} \right)^2 \right) + 2\#(Z_1) + \frac{2}{N} \#(Z) = \theta^* + \frac{2}{N} \#(Z),
\end{align*}
\]

\[
\begin{align*}
\text{AIC}(Z_1 \cup Z) - 2\#(Z) &\leq \theta^* \leq \text{AIC}(Z_1 \cup Z) \leq \text{AIC}(Z_1 \cup Z_0) - 2\#(Z) \\
&= n(\log(2n\pi) + 1).
\end{align*}
\]
by branching \( z_j = 1 \) at the node \( V(Z_1, Z_0, Z) \). In fact, since we have \( (Z_1 \cup \{j\}) \cup (Z \setminus \{j\}) = Z_1 \cup Z \), the relaxation problem (8) at the child node \( V(Z_1 \cup \{j\}, Z_0, Z \setminus \{j\}) \) is equivalent to one at the node \( V(Z_1, Z_0, Z) \). Thus the upper bound at the child node is the same as the one at the node \( V(Z_1, Z_0, Z) \), and the lower bound is two plus the lower bound computed at the node \( V(Z_1, Z_0, Z) \) because of \( 2\#(Z_1 \cup \{j\}) = 2 + 2\#(Z_1) \).

4. Some techniques to improve the numerical performance

We describe some techniques to improve the numerical performance to solve (4).

4.1 SCIP

In order to implement our proposed method, we use SCIP [2,13,17], which is mathematical optimization software and a branch-and-bound framework. SCIP has high user plug-in flexibility which helps to solve (4) efficiently. We implement a procedure, which is called relaxator or relaxation handler, to obtain lower bounds as in Section 3. In addition, we also implement procedures to compute upper bounds via stepwise methods discussed in Section 4.3, and define branching rules described in Section 4.4.

4.2 Handling the linear dependence in data

We can efficiently compute the optimal value of (4) by the linear dependence in data sets. Although linear independence in data sets is often assumed in standard statistical textbooks, some practical data sets, e.g. servo and auto-mpg in the UCI Machine Learning Repository [16], often has linear dependence.

For a given data set \((x_{i1}, \ldots, x_{ip}, y_i) \in \mathbb{R}^p \times \mathbb{R} \quad (i = 1, \ldots, n)\), we denote

\[
    x^0 = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad x^j = \begin{pmatrix} x_{ij} \\ \vdots \\ x_{nj} \end{pmatrix} \quad (j = 1, \ldots, p).
\]

We say that the data have linear-dependent variables if the vectors \( x^0, x^1, \ldots, x^p \in \mathbb{R}^n \) are linearly dependent.

From the definition of linear dependence in data, we can reduce the computational cost for solving (8). More precisely, at the node \( V(Z_1, Z_0, Z) \), if there exists a subset \( S \subseteq Z_1 \cup Z \cup \{0\} \) such that the vectors \( \{x^k : k \in S \cup \{0\}\} \) are linearly dependent, we can fix one of the variables \( z_j \) in \( j \in S \cap Z \) to zero. In fact, since we have \( \sum_{j \in S \cup \{0\}} \alpha_j x^j = 0 \) for some \( (\alpha_j)_{j \in S \cup \{0\}} \neq 0 \), we can remove one variable \( z_j \) by substituting this equation to (8). This implies that the number of variables in (8) decreases. Thus the linear system to solve (8) has fewer variables.

Moreover we can prune some nodes efficiently by linear dependence. The following lemma ensures that we do not need to branch \( z_q = 1 \) for some \( q \in Z \) if the data have linear dependence. Thus we handle only \( z_q = 0 \) in this case.

**Lemma 4.1** Assume that in (5), there exists \( q \in Z \) such that the vector \( x^q \) and vectors \( \{x^j : j \in Z_1 \cup \{0\}\} \) are linearly dependent. Then an optimal solution of (5) satisfies \( z_q = 0 \).
Proof  Let \((\tilde{\beta}_j, \tilde{z}_j)\) be an optimal solution of (5), and \(\theta^*\) be the optimal value of (5). Suppose that \(\tilde{z}_q = 1\). It follows from the assumption that there exists a nonzero vector \((\alpha_j)_{j \in Z_1 \cup \{0\}}\) such that

\[ x^j = \sum_{j \in Z_1 \cup \{0\}} \alpha_j x^j. \]

Then the following solution \((\hat{\beta}_j, \hat{z}_j)\) is feasible for (5):

\[
\hat{\beta}_j = \begin{cases} 
\tilde{\beta}_j + \tilde{\beta}_q \alpha_j & \text{(if } j \in (Z \setminus \{q\}) \cup Z_1 \cup \{0\}), \\
0 & \text{(otherwise)}
\end{cases}
\]

\[
\hat{z}_j = \begin{cases} 
1 & \text{(if } j \neq q \text{ and } \tilde{z}_j = 1), \\
0 & \text{(otherwise)}
\end{cases}
\]

The objective value of (5) at \((\hat{\beta}_j, \hat{z}_j)\) is \(\theta^* - 2\), which contradicts the optimal value \(\theta^*\).

A given data set with linear dependence satisfies the assumption of Lemma 4.1. In fact, there exists a subset \(S \subseteq \{1, \ldots, p\}\) such that the vectors \\{\(x^k : k \in S \cup \{0\}\)\} are linearly dependent. Hence Lemma 4.1 ensures that we do not need to generate a child node by branching \(z_q = 1\) at the node \(V(Z_1, Z_0, Z)\) when there exists \(q \in S \cap Z\) such that \(S \setminus \{q\} \subseteq Z_1\).

In addition, if there exists a nonempty subset \(S \subseteq \{1, \ldots, p\}\) such that for every \(j \in S\) the vectors \(\{x^k : k \in \{0\} \cup (S \setminus \{j\})\}\) are linearly dependent, then we can prune some nodes before applying our proposed method to (4). In fact, it follows from the assumption on \(S\) that for every \(j \in S\) we do not need to branch \(z_j = 1\) at the node \(V(Z_1, Z_0, Z)\). This implies that optimal solutions of (4) satisfy the following linear inequality:

\[ \sum_{j \in S} z_j \leq \#(S) - 1. \]

By adding this inequality to (4), we do not generate any nodes in which \(S \subseteq Z_1\) hold. We execute a greedy algorithm in Algorithm 1 to find a collection \(C\) of such sets \(S\).

---

**Algorithm 1:** A greedy algorithm to find a collection of sets of linearly dependent vectors

**Input:** Data \(x^0, x^1, x^2, \ldots, x^p \in \mathbb{R}^n\)

**Output:** A collection \(C\) of sets of linearly dependent vectors

\(C \leftarrow \emptyset; S \leftarrow \emptyset;\)

for \(j \rightarrow 1\) to \(p\) do

if the vectors \(\{x^j : j \in \{0\} \cup S \cup \{j\}\}\) is linearly independent then

\(S \leftarrow S \cup \{j\};\)

else

Solve the following linear equation:

\[ \sum_{k \in S \cup \{0\}} \alpha_k x^k = x^j. \]  \(\text{(9)}\)

\(S' \leftarrow \{k \in S : \alpha_k \neq 0\}, C \leftarrow C \cup \{S'\};\)

end

end

return \(C;\)
We remark that the linear equation (9) has a unique solution because the matrix \((x^k)_{k \in S \cup \{0\}}\) is of full column rank.

4.3 Computation of upper bounds based on stepwise methods

Using mainly the procedure described in Section 3 to compute upper bounds, we also use the stepwise methods with forward selection (SW+) and backward elimination (SW-). SW+ starts with no explanatory variables and adds one explanatory variable at a time until the AIC value does not decrease. More precisely, for the current set \(S\) of explanatory variables, we choose an explanatory variable whose AIC value \(\text{AIC}(S \cup \{j\})\) is minimized over \(j \in \{1, \ldots, p\} \setminus S\). SW- is just the reverse of SW+. It starts with all explanatory variables and removes one explanatory variable at a time until the AIC value does not decrease. Since these methods add or remove one explanatory variable at a time, they may miss the best statistical model. In this sense, we can say that they are local search algorithms for variable selection.

We describe our heuristics to compute an upper bound in more details in Algorithm 2. To this end, we define the following problem for any node \(V(Z_1, Z_0, Z)\) and \(S \subseteq Z_1 \cup Z\):

\[
\begin{align*}
&\min_{\beta, z_j} \quad n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right) + 2 \sum_{j=1}^{p} z_j \\
&\text{subject to} \quad \beta_0, \beta_j \in \mathbb{R}, z_j = 1 (j \in S), \beta_j = 0, z_j = 0 (j \in \{1, \ldots, p\} \setminus S).
\end{align*}
\] (10)

We denote the optimal value and an optimal solution of (10) by \(\tilde{\theta}_S\) and \((\tilde{\beta}_S, \tilde{z}_S)\), respectively.

**Algorithm 2:** Heuristics to compute an upper bound

**Input:** \(Z_1, Z_0\) and \(Z\)

**Output:** A feasible solution \((\beta, z)\) of (5)

/* Stepwise method with forward selection */
\(S \leftarrow Z_1, v_f \leftarrow \infty;\)

while \(\tilde{\theta}_S < v_f\) do

\(v_f \leftarrow \tilde{\theta}_S, (\tilde{\beta}_f, \tilde{z}_f) \leftarrow (\tilde{\beta}_S, \tilde{z}_S);\)

Find \(j \in Z \setminus S\) such that \(\tilde{\theta}_{S \cup \{j\}}\) is minimized over all \(j \in Z \setminus S;\)

\(S \leftarrow S \cup \{j\};\)

end

/* Stepwise method with backward elimination */
\(S \leftarrow Z_1 \cup Z, v_b \leftarrow \infty;\)

while \(\tilde{\theta}_S < v_b\) do

\(v_b \leftarrow \tilde{\theta}_S, (\tilde{\beta}_b, \tilde{z}_b) \leftarrow (\tilde{\beta}_S, \tilde{z}_S);\)

Find \(j \in Z \cap S\) such that \(\tilde{\theta}_{S \setminus \{j\}}\) is minimized over all \(j \in Z \cap S;\)

\(S \leftarrow S \setminus \{j\};\)

end

if \(v_f < v_b\) then

\(\text{return } (\tilde{\beta}_f, \tilde{z}_f);\)

else

\(\text{return } (\tilde{\beta}_b, \tilde{z}_b);\)

end
Algorithm 2 always returns a feasible solution of (5). In fact, any feasible solution of (10) is feasible for the subproblem (5) if $Z_1 \subseteq S$. Since $S$ generated in Algorithm 2 always contains $Z_1$, the returned solution $(\beta, z)$ is feasible for (5).

Algorithm 2 is different from the original stepwise methods. In fact, we set $Z_1$ as the initial set of $\text{SW}_+$ instead of the empty set because we execute Algorithm 2 at the node $V(Z_1, Z_0, Z)$. Similarly, we set $Z_1 \cup Z$ as the initial set of $\text{SW}_-$. This is the difference from the original stepwise methods.

In statistical applications, instead of finding the global minimum of (4), stepwise methods, which are local search algorithms, are commonly used in practice. Actually, they often find a better statistical model and work effectively in our implementation. However, since stepwise methods imply a higher computational cost than the procedure in Section 3, we apply Algorithm 2 only to the subproblems (5) at the nodes whose depths from the root node are less than or equal to 10 in our implementation.

### 4.4 Most frequent branching and Strong branching

We define two branching rules for variables $z_j$ to improve the performance of our implementation. The first one is called *most frequent branching*, and uses all feasible solutions stored in the procedure to compute upper bounds. The second one is called *strong branching* that is based on the strong branching rule in [1, Section 5.4]. We propose more efficient computation for the strong branching rule than [1]. We will show the numerical comparison with branching rules implemented in SCIP in Section 5.2. We will observe from the numerical results that the most frequent branching is effective for data sets with linear dependence, while strong branching is effective for data sets without linear dependence.

The most frequent branching is based on the tendency that some explanatory variables adopted for the best statistical model are also used in statistical models whose AIC value is close to the smallest AIC value. By branching variables $z_j$ in (5) which correspond to such explanatory variables, we can expect that (5) at the node generated by $z_j = 0$ is pruned as early as possible because the generated subproblem may have no optimal solutions of the original (4). To find such explanatory variables, we use feasible solutions stored in our procedure to compute upper bounds. We describe the most frequent branching rule executed at the current node in Algorithm 3.

**Algorithm 3:** Most frequent branching rule

**Input:** A positive integer $N$, a set $Z$ of unfixed variables in the node and all feasible solutions of (4) stored from the root node to the current node

**Output:** $J \in Z$

Choose $N$ feasible solutions $(\beta^1, z^1), \ldots, (\beta^N, z^N)$ out of all the stored feasible solutions;

/* Here $(\beta^i, z^i)$ is a feasible solution of (4) whose objective value is the $i$th smallest of all the stored solutions */

for $j \in Z$
do

Compute the score value $s_j$ defined by $s_j = \#(T_j)$, where $T_j = \{\ell \in \{1, \ldots, N\} : z_\ell^j = 1\}$

end

return $J \in Z$ with $s_J = \max_{j \in Z} \{s_j\}$

We observe in our preliminary numerical experiment that the lower bound at the child node generated by $z_j = 0$ tends to be relatively bigger. This implies that the pruning process tends to work earlier in comparison to branching rules of SCIP. As a result, our proposed method with
the most frequent branching rule often generates fewer nodes in the branch-and-bound tree than the other branching rules.

In the strong branching rule, we compute lower bounds for all possible branching \( z_k = 1 \) and \( z_k = 0 \), and choose index \( k \in Z \) so that the lower bound is maximized in all the computed lower bounds. More precisely, for the subproblem (5) at the node \( V(Z_1, Z_0, Z) \) and \( k \in Z \), the relaxation problems of the subproblems branched by \( z_k = 1 \) and \( z_k = 0 \) can be formulated as (11) and (12), respectively.

\[
\begin{align*}
\min_{\beta_j} & \quad n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right) + 2#(Z_1 \cup \{k\}) \\
\text{subject to} & \quad \beta_0, \beta_j \in \mathbb{R} \quad (j \in (Z \setminus \{k\}) \cup (Z_1 \cup \{k\})), \beta_j = 0 \quad (j \in Z_0)
\end{align*}
\]  

(11)

\[
\begin{align*}
\min_{\beta_j} & \quad n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right) + 2#(Z_1) \\
\text{subject to} & \quad \beta_0, \beta_j \in \mathbb{R} \quad (j \in (Z \setminus \{k\}) \cup Z_1), \beta_j = 0 \quad (j \in Z_0 \cup \{k\}).
\end{align*}
\]  

(12)

Since we have \((Z \setminus \{k\}) \cup (Z_1 \cup \{k\}) = Z \cup Z_1\), the optimal value of (11) for all \( k \in Z \) is \( \theta^* + 2 \), where \( \theta^* \) is the optimal value of (7) at the node \( V(Z_1, Z_0, Z) \). Hence we select an index \( k \in Z \) only from all the optimal values \( \theta^*_k \) of (12). We describe the strong branching rule at the current node in Algorithm 4.

**Algorithm 4:** Strong branching rule

**Input:** The node \( V(Z_1, Z_0, Z) \)

**Output:** \( J \in Z \)

for \( k \in Z \)
do
  Solve (12) with \( k \) and obtain optimal value \( \theta^*_k \);
end
return \( J \in Z \) with \( \theta^*_J = \max_{k \in Z} \{ \theta^*_k \} \);

5. Numerical experiments

We implement our method and procedures in Sections 3 and 4. We apply our implementation [18] to benchmark data sets in [16] which is standardized, i.e. the data are transformed to have the zero mean and unit variance. Note that the standardized data set has also linear dependence even if the original data have linear dependence. The specification of the computer is CPU : 3.5 GHz Intel Core i7, Memory : 16 GB and OS : OS X 10.9.5. In Sections 5.1 and 5.3, we adopt the most frequent branching rule for the data sets with linear dependence, while adopting the strong branching for the data sets with no linear dependence. In Section 5.2, we explain the reason why we use the different branching rules.

5.1 Comparison with stepwise methods and MISOCO approach

We compare our proposed method with stepwise methods (SW_+ and SW_-) and the MISOCO approach in [12] via CPLEX [9]. The MISOCO formulation is obtained from (4) as well.
Although the objective function of (4) is non-convex, the difficulty due to the non-convexity is overcome by the identity \( \exp(\log(x)) = x \) and the monotonicity of the exponential function \( \exp(x) \). See [12, Section 3.2] for details. The obtained problem is formulated as MISOCP and is tractable by CPLEX.

Table 1 shows the summary of numerical comparisons. The mark • in the first column indicates that the data set has linear dependence. The second, third, and sixth columns indicate the numbers of data, the explanatory variables in the statistical model (1), and the number of the explanatory variables in the models found by using each method. The fifth column indicates the obtained AIC values by each method. The values with the bold font are the best among four values. The seventh column indicates the CPU time in seconds to compute the optimal value. ‘\( > 5000 \)’ means that the corresponding method cannot find the optimal value within 5000 seconds. The last column

| Name           | \( n \) | \( p \) | Methods | AIC   | \( k \) | time(sec) | gap(%) |
|----------------|--------|--------|---------|-------|--------|-----------|--------|
| housing        | 506    | 13     | MINLP   | 776.21| 11     | 0.04      | 0.00   |
|                |        |        | MISOCP  | 776.21| 11     | 7.96      | 0.00   |
|                |        |        | SW+     | 776.21| 11     | 0.35      | –      |
|                |        |        | SW−     | 776.21| 11     | 0.10      | –      |
| •servo         | 167    | 19     | MINLP   | 258.35| 9      | 0.79      | 0.00   |
|                |        |        | MISOCP  | 258.35| 9      | 7.99      | 0.00   |
|                |        |        | SW+     | 258.35| 9      | 0.19      | –      |
|                |        |        | SW−     | 260.16| 10     | 0.18      | –      |
| •auto-mpg      | 392    | 25     | MINLP   | 332.88| 15     | 1.76      | 0.00   |
|                |        |        | MISOCP  | 332.88| 15     | 303.83    | 0.00   |
|                |        |        | SW+     | 334.73| 16     | 0.49      | –      |
|                |        |        | SW−     | 337.96| 18     | 0.32      | –      |
| •solarflareC   | 1066   | 26     | MINLP   | 2816.29| 9 | 10.49      | 0.00   |
|                |        |        | MISOCP  | 2816.29| 9 | 304.51     | 0.00   |
|                |        |        | SW+     | 2816.29| 9 | 0.45       | –      |
|                |        |        | SW−     | 2821.61| 12 | 1.08       | –      |
| •solarflareM   | 1066   | 26     | MINLP   | 2926.90| 7 | 3.99       | 0.00   |
|                |        |        | MISOCP  | 2926.90| 7 | 255.02     | 0.00   |
|                |        |        | SW+     | 2926.90| 7 | 0.36       | –      |
|                |        |        | SW−     | 2930.91| 9 | 1.16       | –      |
| •solarflareX   | 1066   | 26     | MINLP   | 2882.80| 3 | 0.92       | 0.00   |
|                |        |        | MISOCP  | 2882.80| 3 | 19.39      | 0.00   |
|                |        |        | SW+     | 2882.80| 3 | 0.18       | –      |
|                |        |        | SW−     | 2891.56| 9 | 1.20       | –      |
| breastcancer   | 194    | 32     | MINLP   | 508.40| 10 | 90.21      | 0.00   |
|                |        |        | MISOCP  | 508.62| 10 | > 5000     | 3.72   |
|                |        |        | SW+     | 509.30| 8  | 0.24       | –      |
|                |        |        | SW−     | 509.96| 14 | 0.60       | –      |
| •forestsires   | 517    | 63     | MINLP   | 1429.64| 12 | > 5000     | 0.77   |
|                |        |        | MISOCP  | 1431.32| 12 | > 5000     | 6.44   |
|                |        |        | SW+     | 1429.64| 12 | 0.94       | –      |
|                |        |        | SW−     | 1447.36| 21 | 7.43       | –      |
| •automobile    | 159    | 65     | MINLP   | −61.28| 32 | > 5000     | 13.95  |
|                |        |        | MISOCP  | −55.83| 34 | > 5000     | 27.22  |
|                |        |        | SW+     | −28.55| 21 | 1.12       | –      |
|                |        |        | SW−     | −47.61| 40 | 2.64       | –      |
| crime          | 1993   | 100    | MINLP   | 3410.25| 50 | > 5000     | 0.50   |
|                |        |        | MISOCP  | 3469.34| 74 | > 5000     | 8.51   |
|                |        |        | SW+     | 3430.19| 37 | 17.03      | –      |
|                |        |        | SW−     | 3410.25| 50 | 105.40     | –      |
indicates the gap in the percent defined by

$$\text{gap} := \frac{\text{upper bound} - \text{lower bound}}{\max\{1, \text{upper bound}\}} \times 100.$$  

MINLP, MISOCP, SW$_+$ and SW$_-$ indicate our proposed method, the MISOCP approach and the stepwise method with forward selection and backward elimination, respectively. We observe the following from Table 1.

- MINLP computes the optimal value much faster than MISOCP. MINLP finds smaller AIC values than MISOCP, even when MINLP cannot find any optimal solutions within 5000 seconds.
- MINLP is comparable to the stepwise methods SW$_+$ or SW$_-$ for the benchmark sets where MINLP cannot find any optimal solutions within 5000 seconds, e.g., forestfires, automobile and crime. In fact, as in Section 4.3, we use the stepwise methods described in Algorithm 2 in some nodes in our implementation.

5.2 Comparison of branching rules

We compare the numerical performance of the most frequent branching and the strong branching with branching rules implemented in SCIP. In Table 2, Std, MFB and SB stand for numerical results by the branching rules in SCIP, the most frequent branching rule and the strong branching rule. The sixth column indicates the number of the nodes generated by our proposed method with the applied branching rule. The values with the bold font are the best among the three values. We observe from Table 2:

- The most frequent branching rule works more effectively than the others for the benchmark sets with linear dependence. In fact, except for servo and automobile, MFB computes the optimal value the fastest of the three, and the number of the nodes generated by MFB is the smallest. In contrast, the strong branching rule is more efficient than the other branching rules for data sets without linear dependence. This is why we use the strong branching rule for data sets with no linear dependence and the most frequent branching rule for data sets with linear dependence in Tables 1 and 3.

5.3 Comparison with MIQP formulation

Bertsimas and Shioda [6] and Bertsimas et al. [5] provide an MIQP formulation with a cardinality constraint for linear regression. The minimization of AIC can be divided into $(p + 1)$ MIQP problems by fixing the number of selected explanatory variables from 0 to $p$. In fact, the minimization can be equivalently reformulated as follows:

$$\min_{k=0,\ldots,p} \min \{ \text{AIC}(S) : |S| = k \}. \quad (13)$$

Since each inner optimization problem in (13) can be formulated as an optimization problem, we obtain the best statistical model by solving all $(p + 1)$ MIQP problems. In this subsection, we introduce an MIQP formulation by Bertsimas and Shioda [6] and Bertsimas et al [5] for the inner optimization problems in (13). In addition, we provide an efficient algorithm for the optimization problem (13).

Each inner problem in (13) is defined by fixing $k$ in (4). For any fixed $k$, since the logarithm function in the inner problem in (13) has the monotonicity, we find an optimal solution of the
Table 2. Summary of numerical results by branching rules in SCIP (Std), the most frequent branching (MFB) and the strong branching (SB).

| Name          | Methods | AIC  | k   | time (sec) | # of the generated nodes | gap(%) |
|---------------|---------|------|-----|------------|--------------------------|--------|
| housing       | Std     | 776.21 | 11  | 0.05       | 55                       | 0.00   |
|               | MFB     | 776.21 | 11  | 0.05       | 49                       | 0.00   |
|               | SB      | 776.21 | 11  | **0.04**   | 27                       | 0.00   |
| servo         | Std     | 258.35 | 9   | 1.17       | 7577                     | 0.00   |
|               | MFB     | 258.35 | 9   | 0.79       | 4705                     | 0.00   |
|               | SB      | 258.35 | 9   | **0.41**   | 2261                     | 0.00   |
| auto-mpg      | Std     | 332.88 | 15  | 4.06       | 18,959                   | 0.00   |
|               | MFB     | 332.88 | 15  | **1.76**   | 5723                     | 0.00   |
|               | SB      | 332.88 | 15  | 2.68       | 11,586                   | 0.00   |
| solarflareC   | Std     | 2816.29 | 9   | 53.33      | 166,639                  | 0.00   |
|               | MFB     | 2816.29 | 9   | **10.49** | 32,261                   | 0.00   |
|               | SB      | 2816.29 | 9   | 23.13      | 79,015                   | 0.00   |
| solarflareM   | Std     | 2926.90 | 7   | 40.03      | 117,899                  | 0.00   |
|               | MFB     | 2926.90 | 7   | **3.99**   | **11,903**               | 0.00   |
|               | SB      | 2926.90 | 7   | 23.72      | 81,899                   | 0.00   |
| solarflareX   | Std     | 2882.80 | 3   | 4.37       | 9737                     | 0.00   |
|               | MFB     | 2882.80 | 3   | **0.92**   | **1519**                 | 0.00   |
|               | SB      | 2882.80 | 3   | 3.40       | 7453                     | 0.00   |
| breastcancer  | Std     | 508.40 | 10  | 505.70     | 3851 × 10^3              | 0.00   |
|               | MFB     | 508.40 | 10  | 478.66     | 3422 × 10^3              | 0.00   |
|               | SB      | 508.40 | 10  | **90.21**  | **550 × 10^3**           | 0.00   |
| forestfires   | Std     | 1429.64 | 12  | > 5000     | 7480 × 10^3              | 1.11   |
|               | MFB     | 1429.64 | 12  | > 5000     | **13,179 × 10^3**        | **0.77** |
|               | SB      | 1429.64 | 12  | > 5000     | 9938 × 10^3              | 0.95   |
| automobile    | Std     | −60.29 | 32  | > 5000     | 32,192 × 10^3            | **12.30** |
|               | MFB     | −61.28 | 32  | > 5000     | 29,785 × 10^3            | 13.95   |
|               | SB      | −61.59 | 33  | > 5000     | 15,300 × 10^3            | 16.43   |
| crime         | Std     | 3410.25 | 50  | > 5000     | 10272 × 10^3             | 0.78   |
|               | MFB     | 3410.25 | 50  | > 5000     | 9753 × 10^3              | 0.52   |
|               | SB      | 3410.25 | 50  | > 5000     | 1904 × 10^3              | **0.50** |

inner problem by solving the following optimization problem:

\[
\min_{\beta_j} \left\{ \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 : \begin{array}{l}
\sum_{j=1}^{p} z_j = k, \beta_0 \in \mathbb{R}, \\
z_j \in \{0, 1\} (j = 1, \ldots, p), \\
z_j = 0 \Rightarrow \beta_j = 0 (j = 1, \ldots, p)
\end{array} \right\}
\]

(14) is an MIQP formulation. We denote the optimal value of (14) by \( \eta_k^* \). If (14) is infeasible, we set \( \eta_k^* = +\infty \). Then the optimal value of the inner problem in (13) with \( k \) is \( n \log(\eta_k^*) + 2k \). Therefore we obtain the optimal value and an optimal solution of (13) for \( k = 0, \ldots, p \). We describe the naive algorithm in Algorithm 5.

Algorithm 5: Naive algorithm for (4) via MIQP

Input: Minimization of AIC (4)
Output: An optimal solution of (4)

\[\text{for } k \rightarrow 0 \text{ to } p \text{ do}\]

\[\text{Find the optimal value } \eta_k^* \text{ and an optimal solution } (\beta_k^*, z_k^*) \text{ of (14) with } k\]

\[\text{end}\]

\[\text{Find an index } K \text{ with } \theta_k^* = \min_{k=0,\ldots,p} \{n \log(\eta_k^*) + 2k\}\]

\[\text{return } (\beta_K^*, z_K^*);\]
Table 3. Summary of numerical results by MINLP, Naive (Algorithm 5), Fast = and Fast ≤ (Algorithm 6).

| Name         | Methods | AIC  | k   | time (sec) |
|--------------|---------|------|-----|------------|
| housing      | MINLP   | 776.21 | 11  | 0.04       |
|              | Naive   | 776.21 | 11  | 2.54       |
|              | Fast =  | 776.21 | 11  | 2.15       |
|              | Fast ≤  | 776.21 | 11  | 2.43       |
| *servo       | MINLP   | 258.35 | 9   | 0.79       |
|              | Naive   | 258.35 | 9   | 2.27       |
|              | Fast =  | 258.35 | 9   | 1.27       |
|              | Fast ≤  | 258.35 | 9   | 1.29       |
| *auto-mpg    | MINLP   | 332.88 | 15  | 1.76       |
|              | Naive   | 332.88 | 15  | 22.22      |
|              | Fast =  | 332.88 | 15  | 19.04      |
|              | Fast ≤  | 332.88 | 15  | 14.45      |
| *solarflareC | MINLP   | 2816.29 | 9  | 10.49      |
|              | Naive   | 2816.29 | 9  | 26.49      |
|              | Fast =  | 2816.29 | 9  | 18.17      |
|              | Fast ≤  | 2816.29 | 9  | 15.03      |
| *solarflareM | MINLP   | 2926.90 | 7  | 3.99       |
|              | Naive   | 2926.90 | 7  | 25.27      |
|              | Fast =  | 2926.90 | 7  | 8.15       |
|              | Fast ≤  | 2926.90 | 7  | 7.24       |
| *solarflareX | MINLP   | 2882.80 | 3  | 0.92       |
|              | Naive   | 2882.80 | 3  | 10.65      |
|              | Fast =  | 2882.80 | 3  | 2.25       |
|              | Fast ≤  | 2882.80 | 3  | 2.40       |
| breastcancer | MINLP   | 508.40  | 10  | 90.21      |
|              | Naive   | 508.40  | 10  | 420.44     |
|              | Fast =  | 508.40  | 10  | 402.64     |
|              | Fast ≤  | 508.40  | 10  | 421.96     |
| *forestfires | MINLP   | 1429.64 | 12  | > 5000     |
|              | Naive   | 1435.07 | 7   | > 5000     |
|              | Fast =  | 1435.07 | 7   | > 5000     |
|              | Fast ≤  | 1435.07 | 7   | > 5000     |
| *automobile  | MINLP   | −61.28  | 32  | > 5000     |
|              | Naive   | 52.84   | 8   | > 5000     |
|              | Fast =  | 52.84   | 8   | > 5000     |
|              | Fast ≤  | 52.84   | 8   | > 5000     |
| crime        | MINLP   | 3410.25 | 50  | > 5000     |
|              | Naive   | 3646.35 | 4   | > 5000     |
|              | Fast =  | 3646.35 | 4   | > 5000     |
|              | Fast ≤  | 3646.35 | 4   | > 5000     |

The following lemma ensures that we can find an upper bound of $k$ if we have a feasible solution of (4).

**Lemma 5.1** Let $\hat{\theta} \in \mathbb{R}^{p+1}$ be the optimal value of the following optimization problem:

$$
\min_{\beta_j} \left\{ n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right) : \beta_0, \beta_j \in \mathbb{R} \ (j = 1, \ldots, p) \right\}. \tag{15}
$$

In addition, $\bar{\theta}$ is the objective value of (4) at a feasible solution of (4). Then any optimal solution $(\beta^*, z^*)$ of (4) satisfies

$$
\sum_{j=1}^{p} z_j^* \leq \left\lfloor \frac{\bar{\theta} - \hat{\theta}}{2} \right\rfloor.
$$
Proof Let $\theta^*$ be the optimal value of (4) and $(\beta^*, z^*)$ be an optimal solution of (4). Then we have

$$\bar{\theta} \geq \theta^* = n \log \left( \sum_{i=1}^{n} \left( y_i - \beta_0^* - \sum_{j=1}^{p} \beta_j^* x_{ij} \right)^2 \right) + 2 \sum_{j=1}^{p} z_j^* \geq \hat{\theta} + 2 \sum_{j=1}^{p} z_j^*,$$

and thus we have $\sum_{j=1}^{p} z_j^* \leq (\bar{\theta} - \hat{\theta})/2$. Since $z_j^*$ is integer, we obtain the desired result. 

We describe an algorithm based on Lemma 5.1 in Algorithm 6.

Algorithm 6: Faster algorithm for (4) via MIQP

| Input: Minimization of AIC (4) |
|--------------------------------|
| Output: An optimal solution of (4) |
| Solve (15) and obtain $\hat{\theta}$; |
| $\bar{\theta} \leftarrow +\infty$; |
| for $k \rightarrow 0$ to $p$ do |
| | if $k > \left\lfloor \frac{\hat{\theta} - \bar{\theta}}{2} \right\rfloor$ then |
| | break; |
| | end |
| Find the optimal value $\eta_k^*$ and an optimal solution $(\beta_k^*, z_k^*)$ of (14) with $k$; |
| if $\bar{\theta} \geq n \log(\eta_k^*) + 2k$ then |
| | $\bar{\theta} \leftarrow n \log(\eta_k^*) + 2k, (\beta_k^*, z_k^*) \leftarrow (\beta_k^*, z_k^*)$; |
| end |
| return $(\beta^*, z^*)$; |

We provide numerical results on our proposed method, Algorithms 5 and 6 in Table 3. We observe the following from Table 3:

- MINLP outperforms MIQP approaches. In particular, for larger $p$, MINLP obtains much better AIC values than MIQP approaches, although all the approaches cannot solve within 5000 seconds.
- The performance of Fast $\leq$ is similar to Fast $=$, though Fast $\leq$ uses an initial upper bound.
6. Conclusion

We propose the branch-and-bound algorithm for the MINLP formulation (4) of AIC minimization for linear regression, and implement it with SCIP. We formulate an unconstrained optimization problem (7) as the relaxation problem of the subproblem (5). As a result, a lower bound can be computed by solving a linear equation at each node. In addition, an upper bound can be easily computed, i.e. the lower bound plus a constant. A feasible solution of the MINLP formulation (4) is generated from an optimal solution of the relaxation problem (7).

We implement these procedures with SCIP because it has high flexibility in the user plugin. In fact, we implement a relaxator to compute lower and upper bounds, and two branching rules to prune subproblems efficiently. In addition, our implementation efficiently prunes and generates subproblems by using linear dependence in data sets. As a result, we can obtain the best statistical model for \( p \leq 32 \). In addition, our implementation outperforms MISOCOP approach [12] and MIQP approaches [5, 6] in our numerical experiments.

Future work is to apply our implementation to data sets with larger \( p \) and/or \( n \). One way involves the use of parallel computation via ParaSCIP and FiberSCIP [15]. Other information criteria, e.g. BIC and Hannan–Quinn information criteria are already proposed. By changing the objective function in (4), our proposed method can be applied to these information criteria as well.

Acknowledgments

We would like to thank the anonymous referee for providing valuable comments on the presentation of the manuscript.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The second author was supported by JSPS KAKENHI Grant Numbers JP26400203.

References

[1] T. Achterberg, *Constraint integer programming*, Ph.D. thesis, Technische Universität Berlin, 2007.
[2] T. Achterberg, *SCIP: solving constraint integer programs*, Math. Prog. Comp. 1(1) (2009), pp. 1–41.
[3] H. Akaike, *A new look at the statistical model identification*, IEEE Trans. Autom. Control 19(6) (1974), pp. 716–723.
[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users’ Guide*, SIAM, Philadelphia, PA, 1999.
[5] D. Bertsimas, A. King, and R. Mazumder, *Best Subset Selection via a Modern Optimization Lens*, Ann. Stat. 44(2) (2016), pp. 813–852.
[6] D. Bertsimas and R. Shioda, *Algorithm for cardinality-constrained quadratic optimization*, Comput. Optim. Appl. 43 (2009), pp. 1–22.
[7] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
[8] I. Guyon and A. Elisseeff, *An Introduction to Variable and Feature Selection*, J. Mach. Learn. Res. 3 (2003), pp. 1157–1182.
[9] IBM ILOG CPLEX Optimizer 12.6.2, IBM ILOG 2015.
[10] K. Kimura and H. Waki, *Minimization of Akaike’s information criterion via mixed integer nonlinear program*, Proceedings of the 5th International Congress on Mathematical Software, Berlin, 2016, 292–300.
[11] T. Lumley, *Package ‘leaps’*, 2015.
[12] MAIC. Available at https://github.com/k-kimura1224/MAIC.
[13] R. Miyashiro and Y. Takano, *Mixed integer second-order cone programming formulations for variable selection*, Eur. J. Oper. Res. 247 (2015), pp. 721–731.
[14] SCIP: Solving Constraint Integer Programs. Available at http://scip.zib.de/.
[15] R Development Core Team: R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2008. Available at http://www.R-project.org.
[16] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch, ParaSCIP – A parallel extension of SCIP, Competence in High Performance Computing 2010, In: C. Bischof, H.-G. Hegering, W. E. Nagel and G. Wittum (eds.), Springer, 135–148, 2012.
[17] UCI Machine Learning Repository. Available at http://archive.ics.uci.edu/ml/.
[18] S. Vigerske and A. Gleixner, SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework, ZIB-Report 16-24, Zuse Institute Berlin, May 2016.