1 Scientific activity

The research during the first nine months of the ERCIM fellowship was focused on the development of computationally efficient updating, downdating and up-downdating strategies for estimating and evaluating the linear regression model. Specifically, the following problems have been considered:

- The computation of the least-squares solution of a regression model after deleting a block of observations is investigated. The QR decomposition (QRD) is used as main computational tool for re-calculating the downdated solution. Two parallel strategies for downdating the QRD are considered. These are parallel versions of a recently proposed sequential algorithm which efficiently exploits the triangular structure of the matrices and is rich in BLAS-3 operations [2]. Theoretical and experimental results for both strategies are derived and analyzed. The performance of the first algorithm is found to degrade, because of the communication costs which increases significantly with the number of processors. The second approach is found to be efficient and scalable for large scale problems.

Furthermore, a modification of the second parallel algorithm which incorporates the explicit computation of the orthogonal matrix during the computations is proposed. The modified algorithm is analyzed and found to be also perfectly load-balanced, scalable for large scale problems and with efficiency close to one.

Finally, an alternative approach for downdating the QRD, which is based on Hyperbolic Householder transformations, is considered. The numerical results show that the serial hyperbolic algorithm slightly outperforms the best sequential strategy proposed in [2]. However, the hyperbolic downdating is found to be numerically unstable in general and thus, less attractive than the standard approach. Currently, an efficient parallelization of the hyperbolic strategy is under investigation. [4].

- A graph approach for optimizing the computation required in the $k$-fold cross-validation of the regression model is proposed. The main advantage of this approach is that it avoids the computation of all required (desired) models from scratch by efficiently utilizing previous computations. The training (and testing) subsets are presented as nodes of a complete weighted graph. The links between the nodes indicate the different possibilities for deriving the solution of the destination node, given the solution of the source node. The weights of the links represent the computational complexities required for updating, downdating or up-downdating the corresponding data matrices. The graph with all possible links is constructed and its properties investigated.

Several strategies for computing the $k$-fold cross-validation problem are discussed, where each strategy generates different number of models. During the computations, the algorithms traverse different paths of the complete graph and could estimate a number of new additional models (nodes), which are not initially required. In such a way, the different approaches provide different information about the evaluated problem. The advantages and disadvantages of each strategy are discussed. Numerical results are presented and analyzed. The problem of computing the optimal path (i.e the one with the minimum computational complexity) in the graph which computes all (desired) nodes is addressed as well.
In the context of detecting influential observations, the possibility of computing all 'close to a specific node' models is also discussed and numerical results are presented and analyzed. Currently, the proposed methods are evaluated using real datasets. [3].

- A generalization of a branch and bound strategy using a directed graph approach with similar properties is investigated for deriving all possible subset regression models. The graph is found to have a recursive structure with useful properties which can be efficiently exploited. A group of equivalent Minimum Spanning Trees (MSTs) is derived from the graph. The trees are equivalent in the sense of having the same (minimum) computational complexity for deriving all possible subset regression models. The sub-models are efficiently estimated by computing the MST which comprise all variables of the sub-model. The various MSTs yield different regression tree strategies for model selection. Each tree has specific structure and characteristics.

The new graph branch and bound algorithm (GBBA) computes the best sub-models, without generating all nodes of the graph, i.e. searching for the best sub-models it prunes the non-optimal sub-graphs. Numerical results show small improvement of the GBBA over the existing branch and bound algorithm based on a regression tree.

The generalization of the regression graph can be used in solving similar combinatorial problems as the exhaustive \(k\)-fold cross-validation and outliers identification. [1].

The submitted results are shown in the Attachment (copies of papers).

# 2 Publication(s) during your fellowship

The article [1] has been recently submitted for publication in the *Computational Statistics and Data Analysis* journal and is attached in the Appendix. The other two articles [3] and [4], which have been presented at the conferences mentioned below will be completed in the coming months and submitted for publication to international journals.

# 3 Attended Seminars, Workshops, and Conferences

Conference presentations:

1. "A Graph based strategy for the \(k\)-fold cross-validation problem".  

2. "Computational strategies for the \(k\)-fold cross-validation problem using a graph approach".  
   Presented at the 8th Workshop of the ERCIM working group on Matrix Computations and Statistics, Salerno, Italy, September 2-3, 2006.

3. "Parallel algorithms for downdating the least squares estimator of the regression model".  

Member of the scientific programme committee:

- 8th Workshop of the ERCIM working group on Matrix Computations and Statistics, Salerno, Italy, September 2-3, 2006.

# References


**Attachment**
A GRAPH APPROACH TO GENERATE ALL POSSIBLE REGRESSION SUBMODELS

CRISTIAN GATU†, PETKO YANEV‡, AND ERRICOS JOHN KONTOGHIORGHES§ ¶

Abstract. A regression graph to enumerate and evaluate all possible subset regression models is introduced. The graph is a
generalization of a regression tree. All the spanning trees of the graph are minimum spanning trees and provide an optimal computa-
tional procedure for generating all possible submodels. Each minimum spanning tree has a different structure and characteristics.
An adaptation of a branch-and-bound algorithm which computes the best-subset models using the regression graph framework is
proposed. Experimental results and comparison with an existing method based on a regression tree are presented and discussed.

Key words. Graphs; Regression trees; Model selection; Combinatorial algorithms.

1. Introduction. An important topic in statistical modeling is the subset-selection regression or,
equivalently, finding the best regression equation [10, 11, 12, 13, 19]. That is, given a list of possible
variables to be included in the regression, the problem is to select a subset of them which optimize some
statistical criterion. The latter derives from the estimation of the corresponding sub-model [21, 22, 25, 26].
In the case of the standard regression model having \( n \) parameters there are \( 2^n - 1 \) possible submodels which
have to be compared. Another important area where subset models need to be specified is within the context
of estimating the parameters of a Vector Autoregressive (VAR) process. A \( G \)-multivariate VAR process of
order \( p \) yields \( 2^{G^2} - 1 \) submodels [6]. When the number of parameters to choose from is not too large, it
is convenient to enumerate and evaluate all possible submodels [22, 27]. This approach has the advantage
of the exhaustive search based methods, that is to provide the optimum solution. Strategies for generating
all possible subset regression models have been previously considered. Within this context, a dropping
columns algorithm that generates a regression tree together with a parallel version of this algorithm have
been proposed [3, 5, 28]. The computation involved in these methods are based on the QR factorization
and re-triangularization of a matrix after deleting columns. One property of the regression tree is that
once a submodel comprising the variables \( \{v_1, v_2, \ldots, v_d\} \) is derived, the submodels corresponding to the
subsets \( \{v_1\}, \{v_1, v_2\}, \ldots, \{v_1, v_2, \ldots, v_d\} \) become available without extra computational cost. These subsets
 correspond to the already triangular sub-leading \( 1 \times 1, 2 \times 2, \ldots, d \times d \) matrices [5, 7]. Thus, the problem
of generating all \( 2^n - 1 \) possible subset regression models becomes to generate the regression tree having
as root node the full specified model \( \{1, 2, \ldots, n\} \).

Recently, an algorithm for computing the main matrix factorizations arising in the estimation of seem-
ingly unrelated regression equation models with common variables has been introduced [29]. This can be
seen as equivalent to estimating a set of sub-models having common variables. The algorithm is based on
weighted directed graphs. The nodes corresponds to models (sets of variables) and an edge exists between
two nodes if the destination node is subset of the source node. The weight of an edge is given by the
computational complexity of estimating the sub-model of the destination given that the source has been
already estimated. Thus, the sub-models are efficiently estimated by computing the shortest path from the

---

*This work is in part supported by the Swiss National Foundation Grants 101412-105978 and 200020-100116/1. Part of the
work of the second author was done while he was visiting the University of Cyprus, Cyprus under the support of the Swiss National
Science Foundation Grant PIOI-1-10144.

†Institut d’informatique, Université de Neuchâtel, Emile-Argand 11, CH-2009 Neuchâtel, Switzerland (cristian.gatu@unine.ch).
‡Istituto di Informatica e Telematica del CNR, Area della Ricerca di Pisa, Via Moruzzi 1, 56124 Pisa, Italy
(petko.yanev@iit.cnr.it).
§Department of Public and Business Administration, University of Cyprus, Cyprus (erricos@ucy.ac.cy).
¶School of Computer Science and Information Systems, Birkbeck College, University of London, UK (erricos@dcs.bbk.ac.uk).
root node which comprise all variables of the model to the required nodes.

A regression graph which yields all possible subset regression models is introduced. The graph can be seen as a generalization of the regression tree in [5, 6, 7]. Specifically, the regression tree describes a non-unique shortest path for traversing the graph. Furthermore, all the subtrees of the graph containing all the nodes are equivalent in the sense that they provide all subset models with the same minimum computational complexity.

The next section introduces a class of regression graphs which can be employed in statistical model selection. It describes how the combinatorial problem of enumerating all possible subset regression models can be formalized with directed graphs. Theoretical measures of complexity of generating all models by traversing a regression graph are presented. Section 3 shows the relationship between the regression graphs and the regression trees. That is, how the various minimum spanning (regression) trees can be obtained from the regression graph. The merits of the derived regression trees are discussed. Section 4 presents the generalization of a branch-and-bound algorithm for computing the best-subset regression models using the graph structure. Finally, Section 5 concludes.

2. Regression graphs. Consider the standard regression model

\[ y = A\beta + \epsilon, \quad \epsilon \sim (0, \sigma^2 I), \]

where \( A \in \mathbb{R}^{m \times n} \) comprises the variables (columns) \( v_1, \ldots, v_n \). It is assumed that \( m > n \), \( A \) has full rank and the QR decomposition (QRD) of \( A \) is given by:

\[ Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}. \]  

Here \( Q \in \mathbb{R}^{m \times m} \) is orthogonal and \( R \in \mathbb{R}^{n \times n} \) is upper-triangular. Let \( A^{(i)} \) comprising the first \( i \) variables, i.e. \( v_1, \ldots, v_i \), and \( R^{(i)} \) denotes the leading \( i \times i \) triangular sub-matrix of \( R \) such that

\[ Q^T A^{(i)} = \begin{pmatrix} R^{(i)} \\ 0 \end{pmatrix}_{i \times m}, \quad i = 1, \ldots, n. \]

The least squares solution of the \( i \)-variable regression model \( y = A^{(i)}\beta^{(i)} + \epsilon \) is computed by \( \hat{y}^{(i)} = R^{(i)}\hat{\beta}^{(i)}, \)

\[ i = 1, \ldots, n, \]

where

\[ Q^T y = \begin{pmatrix} \hat{y}^{(i)} \\ \hat{\beta}^{(i)} \end{pmatrix}_{i \times m}. \]

Thus, the QRD (1) provides the least squares solution of \( n \) sub-models. Generally, \( 2^{(n-1)} - 1 \) submodels out of the \( 2^n - 1 \) possible ones are derived trivially in this way. Thus, the remaining \( 2^{(n-1)} \) non-trivial sub-models which can be generated from the set of \( n \) variables \( V = [v_1, \ldots, v_n] \) need to be computed. This includes the full model \( [v_1, \ldots, v_n] \) and excludes all the sub-models which are immediately available from the leading triangular sub-matrices [5, 7].

The \( 2^{(n-1)} \) sub-models are given by \( N(V, 0) \), and \( N(V, k) \) is defined recursively as

\[ N(V, k) = \begin{cases} \{V\} & \text{if } k \geq n - 1, \\
\{V\} \cup (\bigcup_{j=k}^{n-2} N(V - [v_{j+1}], j)) & \text{if } k < n - 1,
\end{cases} \]

where \( 0 \leq k < n \). The corresponding triangular sub-matrices can be represented as nodes of a weighted directed regression graph. A node in the graph is a set of indexes that corresponds to the variables (columns).
of the original set \([1,2,\ldots,n]\) included in \(V_i\) and is denoted by \(V_i = [v_i^1,v_i^2,\ldots,v_i^j]\), where \(i = 1,\ldots,2^{(n-1)}\) and \(1 \leq j \leq n\). An edge between two nodes \(V_i\) and \(V_j\) is denoted by \(E_{i,j}\). It exists and is directed from \(V_i\) towards \(V_j\) if and only if \(V_j \subseteq V_i\) \((i,j = 1,\ldots,2^{(n-1)}\) and \(i \neq j\)). Formally, this directed graph will be denoted by \(G_V\) and defined by the tuple
\[
G_V = (X, \mathcal{U}) \quad \text{with} \quad \begin{cases} 
X = N(V,0), \\
\mathcal{U} = \{E_{i,j} = (V_i,V_j) : V_i, V_j \in X \text{ and } V_j \subset V_i\}.
\end{cases}
\]

Figure 1 shows the graph \(G_V\) for \(V = [1,2,3,4]\).

![Figure 1. The graph \(G_V\) for \(V = [1,2,3,4]\) (Example)](image)

The weight of \(E_{i,j}\) is denoted by \(C_{i,j}\) and is proportional to the complexity of estimating the sub-model corresponding to \(V_j\) given that the one corresponding to \(V_i\) has already been estimated. Recall that this is equivalent to the re-triangularization of an upper-triangular matrix after deleting some columns \([5, 7, 9, 17, 29]\). Let \(k = |V_j|, V_j = [v_j^1,v_j^2,\ldots,v_j^k]\), the edge \(E_{i,j}\) from some node \(V_i\) towards \(V_j\) exist and \(p_{i,t}\) denote the position of the \(v_i^t\) in \(V_i\) \((t = 1,\ldots,k)\). Note that, since \(V_j \subset V_i\), \(p_{i,t} \geq t\) for every \(t\). Then, the weight of the edge \(C_{i,j}\) is given by
\[
C_{i,j} = \sum_{t=1}^{k} (p_{i,t} - t)(k - t + 1).
\]

This corresponds to the computational cost of the construction and application on 2-row sub-matrices of a sequence of Givens rotations in order to annihilate the non-zero elements below the main diagonal of the sub-matrix corresponding to \(V_j\) \([7, 29]\). Figure 2 shows this sequence of Givens rotations for \(V_i = [1,2,3,4,5,6]\) and \(V_j = [1,2,4,5,6]\). That is, when variable 3 has been deleted. Figure 3(a) shows the costs \(C_{i,j}\) corresponding to all the edges \(E_{i,j}\) of the complete graph \(G_V\), while Figure 3(b) shows all the nodes which are non-trivial subsets of \(V\) for \(V = [1,2,3,4]\). The computational costs of deriving these nodes from \(V\) are also shown as the weights of the edges.

Now, using this graph representation, the problem of generating all \(2^{(n-1)}\) non-trivial sub-sets of \(V = [1,2,\ldots,n]\), i.e. sub-matrices, becomes equivalent to visiting the \(2^{(n-1)}\) nodes of the graph \(G_{[1,\ldots,n]}\). The edges and weights of \(G_{[1,\ldots,n]}\) provide all the possibilities and the associate cost of moving from one node to another by deleting variables. Finally, in order to derive the triangular factors of all nodes with minimum cost, the optimal path for visiting all the nodes is required. This is equivalent to finding one or more minimum spanning trees (MST) of the proposed weighted directed graph \([18, 23]\).

The nodes of \(G_V = (X, \mathcal{U})\) with \(|V| = n\) can be divided into \(n\) levels \(L_1,\ldots,L_n\), such that \(L_k\) contains all nodes having exactly \(k\) variables, i.e.
\[
L_k = \{W : W \in X \text{ and } |W| = k\}, \quad k = 1,\ldots,n.
\]
Note that, \( L_n = \{1, \ldots, n\} \) and \( L_1 = \{n\} \). Now, let \( V_s \in L_p \), \( V_h \in L_q \) and \( V_l \in L_r \), where \( E_{s,t} \) and \( E_{h,t} \) exist, and \( p > q > r \). If there is a path from \( V_s \) to \( V_h \), then \( C_{h,t} \leq C_{s,t} \) [29]. Therefore, the edge \( E_{s,t} \) can be deleted from the graph. Note that, a path from \( V_s \) to \( V_h \) exists if and only if the set of variable indexes of \( V_h \) is a subset of the set of variable indexes of \( V_s \). Figure 4(a) illustrates this “edges-deletion-rule” where \( V_s = [1, 2, 3, 4], p = 4, V_h = [1, 4], q = 2, V_l = [4] \) and \( r = 1 \). The concerned edges are in bold and the deleted one is in dashed. The cost \( C_{h,t} \) and \( C_{s,t} \) are also shown. Following this rule, the graph \( \hat{G}_V \) can be simplified by removing all the edges between non-adjacent levels. This graph denoted by \( \Gamma_V \) derives from \( \hat{G}_V = (X, E) \), and is given by

\[
\Gamma_V = (X, E) \quad \text{with} \quad \begin{cases} 
X & \text{- the same nodes as } \hat{G}_V, \\
E & = \{E_{i,j} : (V_i, V_j) \in \mathcal{U} \text{ and } |V_i| - |V_j| = 1\}. 
\end{cases}
\]

Figure 4(b) illustrates the graph \( \Gamma_V \) with all possible edges that exist between adjacent levels, for \( V = [1, 2, 3, 4, 5] \).

After computing the costs \( C_{i,j} \) for all remaining edges, which are only between adjacent levels, the number of edges in \( \Gamma_V \) can be further reduced. For each node, the incoming edge with minimum weight is chosen and remains in the graph, while all other incoming edges are deleted. If there is more than one incoming edge with minimum weight, then all of them are kept. This second “edges-deletion-rule” is illustrated in Figure 5(a) for the incoming edges of the node \( [2, 4] \), where the retained and deleted edges are denoted by bold and dashed arcs, respectively. Let this reduced graph be denoted by \( \overline{\Gamma}_V \). Figure 5(b) illustrates the graph \( \overline{\Gamma}_V \) after deleting the unnecessary edges, for \( V = [1, \ldots, n] \) and \( n = 5 \). The (minima) weights of the edges are also displayed. Each spanning tree of \( \overline{\Gamma}_V \) has the same total cost of visiting all the nodes which is equal to the sum of the weights of all its edges. This total sum is optimal (minimum) and thus, each spanning tree of \( \overline{\Gamma}_V \) is a MST.
A graph approach to generate all possible regression submodels

(a) The non-adjacent levels “edges-deletion-rule” for $V_s = [1, 2, 3, 4], p = 4, V_h = [1, 4], q = 2, V_t = [4]$ and $t = 1$.

(b) The graph $\Gamma_V$, for $V = [1, 2, 3, 4, 5]$ (32 edges).

Fig. 4. The reduction of edges between non-adjacent levels.

(a) The non-minimum weight “edges-deletion-rule” for the node $[2, 4]$.

(b) The graph $\hat{\Gamma}_V$ together with the minima weight of the edges, for $V = [1, 2, 3, 4, 5]$ (26 edges).

Fig. 5. The reduction of the non-minimum weight incoming edges.

Independent of these ”edges-deletion-rules”, the reduced graph $\hat{\Gamma}_V$ can be constructed recursively. Let $|V| = 1$, i.e. $V = \{v\}$. In this case, $\Gamma_V \equiv \hat{\Gamma}_V$. This is the graph having only one node corresponding to the set $\{v\}$, and has no edges. Assume, now, that $\Gamma_V = (\overline{X}, \overline{E})$ has been defined for some set $V = \{v_1, \ldots, v_n\}$ and a new variable $w$ is added to $V$. The definition of $\hat{\Gamma}_{w \cdot V}$ is required, where $w \cdot V \equiv \{w, v_1, \ldots, v_n\}$. Let

\[
\hat{\Gamma}_{w \cdot V} = (\hat{\chi}, \hat{\varepsilon}) \quad \text{with} \quad \left\{ \begin{array}{l}
\hat{\chi} = \{w \cdot V_i : V_i \in \overline{X}\}, \\
\hat{\varepsilon} = \{\hat{E}_{i,j} = (w \cdot V_i, w \cdot V_j) : \overline{E}_{i,j} = (V_i, V_j) \in \overline{E}\}. 
\end{array} \right.
\]

The weight of $\hat{E}_{i,j}$ is the same as that of $\overline{E}_{i,j}$. Furthermore, the graphs $\Gamma_V$ and $\hat{\Gamma}_{w \cdot V}$ have the same size and structure. The only difference is that the nodes represent different subsets (models) since the nodes of $\hat{\Gamma}_{w \cdot V}$ are obtained by adding $w$ to the nodes of $\Gamma_V$. Both graphs $\Gamma_V$ and $\hat{\Gamma}_{w \cdot V}$ are shown in Figure 6(a) and Figure 6(b), respectively, for $V = [2, 3, 4, 5]$ and $w = 1$. Notice that the variable $w$ exists only in $\hat{\Gamma}_{w \cdot V}$ but not in $\Gamma_V$. In Figure 6(a) the graph has been stretched by prolonging the edges and the recursive subgraphs $\Gamma_{[3,4,5]}$ and $\hat{\Gamma}_{2,[3,4,5]}$ have been framed in order to illustrate its recursive definition.
Now, given $\Gamma_V$ and $\hat{\Gamma}_{w,V}$, the graph $\Gamma_{w,V}$ is defined as

$$\Gamma_{w,V} = (X, E) \quad \text{with}$$

$$X = \overline{X} \cup \hat{X},$$

$$E = \overline{E} \cup \hat{E} \cup \{E_{i,i'} = ([w,v_i,v_{i+1},\ldots,v_n],[v_i,v_{i+1},\ldots,v_n]) : i = 1, \ldots, n\}. \quad (3)$$

The weight $C_{i,i'}$ of the added edges of form $E_{i,i'}$ that connect the two subgraphs $\Gamma_V$ and $\hat{\Gamma}_{w,V}$ is given by $C_{i,i'} = i(i + 1)/2$. This completes the recursive definition of $\Gamma_V$. From the recursive definition in (3) and the computation of the weight of the added edges $E_{i,i'} (i = 1, \ldots, n)$ in (2) it follows that each graph $\Gamma_{w,V}$ can be constructed once the smaller graph $\Gamma_V$ is derived. Figure 7 shows an example of the graph $\Gamma_{w,V}$, for $w = 1$ and $V = [2,3,4,5,6]$. The two subgraphs $\Gamma_V$ and $\hat{\Gamma}_{w,V}$ are well distinguished at the left and right of the illustration, respectively. The weights of the new edges that connects them are shown.

Hence, the recursive weighted directed graph $\Gamma_V$ is optimal in the sense that all its spanning trees are MST and provide an optimal computational procedure (i.e. minimum computational cost) for deriving all possible sub-models.

3. MSTs and Regression trees. The MSTs derived from $\Gamma_V$ differ in their structure. Thus, some of them exhibit properties and characteristics that could be more suitable for specific problems such as parallel strategies for deriving all subset models, branch-and-bound selection and subrange model derivation [5, 7, ...]
15]. Consider the MST of $\Gamma_5$, denoted by $T_5$ of Figure 8(b). This tree is of particular interest because it keeps the recursive structure of the graph. That is, it can be recursively constructed independently of $\Gamma_5$. This regression tree, $T_n$, has been investigated and its properties thoroughly exploited within the context of model selection in [7]. A parallel algorithm for computing all possible subset regression models using this regression tree has also been proposed [5].

**Fig. 8.** The MST $T_5$ derived from $\Gamma_V$, for $V = [1, \ldots, n]$, and $n = 5$.

Another MST of $\Gamma_n$, denoted by $T_n^*$, which can be of particular interest is presented in Figure 9(b), where $n = 5$. Note that, the cost of generating $T_n^*$ is the same as for that of generating $T_n$. However, $T_n^*$ has more balanced structure in the sense that the cost of generating the left subtrees with root nodes $[1245]$ and $[1345]$ is the same as the cost of generating the right subtrees with root nodes $[2345]$ and $[1235]$. This property allows $T_5^*$ to be computed by two processors using a complete load distribution [16]. I.e., two processors can compute the $T_5^*$ in half the time needed for computing the whole tree by a single processor.

**Fig. 9.** The MST $T_5^*$ derived from $\Gamma_V$, for $V = [1, \ldots, n]$, and $n = 5$.

4. **Branch-and-bound strategy for model selection.** A branch-and-bound algorithm (BBA) for computing the best-subset regression models together with its heuristic counterpart have recently been developed [7, 15]. Here, a generalization of these strategies using the regression graph $\Gamma_V$ is proposed (GBBA) [2, 4, 20, 24]. Specifically, the pruning procedure described in detail in [7, 15] can be enhanced by using the edges of the graph. Thus, when the cutting test holds, the right-hand side subtrees (subgraphs)
are cut, but also some subgraphs from the left-hand side of the graph.

This is illustrated in Figure 10 for \( n = 6 \). Specifically, using the regression tree approach (BBA), the node \([16]\) is tested against the bound node \([13456]\). If the test holds, then the whole subtree having as root \([13456]\) is cut. Now, using the graph structure (GBBA), in particular the edge \(([13456],[3456])\), the subgraph having as root the node \([3456]\) can be also pruned, except of the nodes on the leftmost path — \([3456], [456], [56], [6]\) — that contain sub-models of size one — \([3], [4], [5], [6]\). The latter nodes need to be tested before pruning. The reduced search space is depicted in Figure 10(b). The special nodes that require additional testing are shown in rectangular frames.

The details of the GBBA procedure are shown in Algorithm 1. A set \( V \) with \( k \) passive variables which are kept in the subsequent generated subgraphs is denoted by the pair \( \{V,k\} \). These pairs represent the nodes of the graph and are stored in the list \( L \) based on the "first in - first out" principle. In the beginning the root node of the graph is placed in the list \( L \). That is, the initial full set of variables. The number of passive variables is set to 0, i.e. all variables are candidate to be dropped. During the execution of the algorithm a pair \( \{V,k\} \) is extracted from the list and the leading \( |V| - k \) new submodels are obtained. The remaining \( k \) leading submodels have been already generated at the earlier stages of the algorithm. In subsequent steps the children of the node are generated one by one subject to that the branch-and-bound cutting test fails. A variable from the first set in the list \( L \) is flagged for the deletion when a subgraph is pruned. In the above example the variable 2 of the node \([23456]\) is flagged. This allows to cut directly the subgraph with root \([3456]\) when processing the node \([23456]\).

The performance of BBA was significantly improved by sorting the variables in the initial list, prior the execution of the algorithm [7]. The same strategy has been also applied on the GBBA. These versions are denoted by BBA–1 and GBBA–1, respectively. The algorithms have been implemented in C++ with BLAS and LAPACK using GNU compiler collection on Pentium class machines with 512 MB of RAM, running under Linux. The datasets employed in the experiments have been constructed using randomly generated data from a uniform distribution. Table 1 shows the execution times in seconds and the number of generated nodes of BBA, GBBA, BBA–1 and GBBA–1. All methods are exhaustive and generate the (same) best sub-models.

As in the case of the BBA, it can be observed that preordering the variables considerably improves the performance of GBBA. The graph approach performs slightly better when compared with the classical approach based on regression trees. In fact, it can be proved that the two algorithms are equivalent. That
### Algorithm 1 The Graph Branch-and-Bound algorithm for finding the best-subset models

```
1: procedure gbba(V_root)  
2: insert {V_root, 0} in the list L  
3: while L ≠ ∅ do  
4:   extract {V, k} from the list L, where n = |V| and V = [v1, v2, ..., vn].  
5:   obtain the leading submodels [v1, ..., vk+1], ..., [v1, ..., vk+1, ..., vn]  
6: if n − k < 1 then  
7:   if RSS(V) > ρ_{k+1} then  
8:     if L ≠ ∅ then  
9:       extract {W, μ} from the list L  
10:      flag the variable w_{μ+1} for deletion; insert {W, μ} in the list L  
11:     end if  
12:   else  
13:     if variable v_{k+1} is flagged for deletion then  
14:       W ← V  
15:       repeat  
16:         W ← drop(W, k + 1)  
17:         obtain the leading submodels [w1, ..., w_{k+1}], ..., [w1, ..., w_{k+1}, ..., w_{|W|}]  
18:         i ← |W| − k  
19:         if RSS(W) > ρ_{k+1} then  
20:           i ← 1  
21:           [Terminate the repeat-until iteration]  
22:         if L ≠ ∅ then  
23:           extract {U, p} from the list L  
24:           flag the variable u_{p+1+|U|−|W|} for deletion; insert {U, p} in the list L  
25:         end if  
26:       end if  
27:     else insert {drop(V, k + 1), k} in the list L  
28:   end if  
29: remove flags from the variables v_{k+1}, ..., vn  
30: for j = k + 2, ..., n − 1 do  
31:   if RSS(V) > ρ_{j} then j ← n else insert {drop(V, j), j − 1} in the list L  
32: end for  
33: end if  
34: end if  
35: end while  
36: end procedure
```

is, any left-hand side subgraph pruned by the GBBA, is also cut by the BBA. The improvement observed by GBBA against the BBA is due to the fact that the former derives additional nodes in earlier stages (the rectangular nodes in Figure10(b)) and, thus, provide better bounds when investigating the remaining subgraphs.
C. GATU, P. YANEV and E. J. KONTOGHIORGES

Table 1

Execution times in seconds and number of generated nodes for the BBA and GBBA without and with preordering.

<table>
<thead>
<tr>
<th>n</th>
<th>BBA</th>
<th>GBBA</th>
<th>BBA-1</th>
<th>GBBA-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>nodes</td>
<td>time</td>
<td>nodes</td>
</tr>
<tr>
<td>20</td>
<td>0.09</td>
<td>16’126</td>
<td>0.08</td>
<td>15’911</td>
</tr>
<tr>
<td>25</td>
<td>0.88</td>
<td>136’811</td>
<td>0.86</td>
<td>136’798</td>
</tr>
<tr>
<td>30</td>
<td>8.13</td>
<td>1’036’134</td>
<td>7.73</td>
<td>1’024’216</td>
</tr>
<tr>
<td>35</td>
<td>62.88</td>
<td>7’171’837</td>
<td>59.58</td>
<td>6’913’952</td>
</tr>
<tr>
<td>40</td>
<td>345.74</td>
<td>31’209’257</td>
<td>327.73</td>
<td>30’767’331</td>
</tr>
<tr>
<td>45</td>
<td>8229.82</td>
<td>739’626’349</td>
<td>7981.01</td>
<td>727’498’653</td>
</tr>
</tbody>
</table>

5. Conclusions. A directed graph approach which can be employed in statistical model selection has been proposed. Specifically, the combinatorial problem of generating all possible subset regression models has been formalized with the regression graph. Thus, enumerating all subsets becomes equivalent to traversing all nodes of the graph. Theoretical measures of complexity of generating all submodels by traversing the regression graph have been presented. The properties of the graph have been investigated and exploited in order to significantly reduce the number of edges. The resulting graph has a recursive structure and it is optimal in the sense that all of its spanning trees are Minimum Spanning Trees (MST) which provide an optimal computational procedure for generating all possible subset models. The MSTs yield different regression tree strategies for model selection. Each tree has different structure and characteristics.

A generalization of a branch-and-bound algorithm for computing the best-subset models using the regression graph structure (GBBA) has been proposed. The algorithm avoids to generate all nodes of the graph when searching for the best submodels by pruning non-optimal subgraphs. Experiments have shown a small improvement of the GBBA over the existing Branch-and-Bound Algorithm based on a regression tree [7].

The above model selection algorithms can be modified to deal with Vector Autoregressive subset model selection [5, 6, 8]. The regression graph provides a framework that can be extended to solve similar combinatorial problems such as k-fold cross validation and identification of influential data. In this case, the node indices represent observations, rather than variables [1, 14].

REFERENCES

A graph approach to generate all possible regression submodels


