Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Parallel Computing 29 (2003) 505–521

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Parallel algorithms for computing all possible subset regression models using the QR decomposition ☆

Cristian Gatu, Erricos J. Kontoghiorghes *

Institut d'informatique, Université de Neuchâtel, Émile-Argand 11, Case Postale 2, CH-2007 Neuchâtel, Switzerland

Received 20 February 2002; received in revised form 29 August 2002

Abstract

Efficient parallel algorithms for computing all possible subset regression models are proposed. The algorithms are based on the dropping columns method that generates a regression tree. The properties of the tree are exploited in order to provide an efficient load balancing which results in no inter-processor communication. Theoretical measures of complexity suggest linear speedup. The parallel algorithms are extended to deal with the general linear and seemingly unrelated regression models. The case where new variables are added to the regression model is also considered. Experimental results on a shared memory machine are presented and analyzed.

Keywords: Parallel algorithms; Subset regression; Least squares; QR decomposition; Givens rotations

☆ This work is in part supported by the Swiss National Foundation Grants 1214-056900.99/1, and 2000-061875.00/1. Part of the work of the second author was done while he was visiting INRIA-IRISA, Rennes, France under the support of the Swiss National Foundation Grant 83R-065887.

* Corresponding author.
E-mail addresses: cristian.gatu@unine.ch (C. Gatu), erricos.kontoghiorghes@unine.ch (E.J. Kontoghiorghes).

0167-8191/03/$ - see front matter
doi:10.1016/S0167-8191(03)00019-X

1. Introduction

The problem of computing all possible subset regression models arises in statistical model selection. Most of the criteria used to evaluate the subset models require the residual sum of squares (RSS) [21]. Consider the standard regression model

$$y = A\beta + \varepsilon, \tag{1}$$

where $y \in \mathbb{R}^m$ is the dependent variable vector, $A \in \mathbb{R}^{m\times n}$ is the exogenous data matrix of full column rank, $\beta \in \mathbb{R}^n$ is the coefficient vector and $\varepsilon \in \mathbb{R}^m$ is the noise vector. It is assumed that ε has zero mean and variance–covariance matrix $\sigma^2 I_m$. Let the QR decomposition (QRD) of A be given by

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \text{ and } Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \tag{2}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular and non-singular. The least squares (LS) solution and the RSS of (1) are given by $\hat{\beta} = R^{-1}y_1$ and $y_2^T y_2$, respectively, [2]. Let $A_{(S)} = AS$ and $\beta_{(S)} = S^T\beta$, where S is an $n \times k$ selection matrix such that AS selects k columns of A. Notice that the columns of S are the columns of the identity matrix I_n. For the LS solution of the modified model

$$y = A_{(S)}\beta_{(S)} + \varepsilon, \tag{3}$$

the QRD of $A_{(S)}$ is required. This is equivalent to re-triangularizing R in (2) after deleting columns [13,14,17]. That is, computing the factorization

$$Q_{(S)}^T RS = \begin{pmatrix} R_{(S)} \\ 0 \end{pmatrix} \begin{matrix} k \\ n-k \end{matrix} \text{ and } Q_{(S)}^T y_1 = \begin{pmatrix} \tilde{y}_1 \\ \hat{y}_1 \end{pmatrix} \begin{matrix} k \\ n-k \end{matrix}. \tag{4}$$

The LS estimator for the new model and its corresponding RSS are given by $\hat{\beta}_{(S)} = R_{(S)}^{-1}\tilde{y}_1$ and $\mathrm{RSS}_{(S)} = \mathrm{RSS} + \hat{y}_1^T\hat{y}_1$, respectively. Let e_i denote the ith column of the $n \times n$ identity matrix I_n. Notice that if $S = (e_1, e_2, \ldots, e_k)$, then $Q_{(S)}^T = I_n$ and $R_{(S)}$ is the leading $k \times k$ sub-matrix of RS, where $k = 1, \ldots, n$.

The number of all possible selection matrices S, and thus models, is $2^n - 1$. The leading sub-matrices of R provide n of these models. As n increases, the number of models to be computed increases exponentially. Therefore, efficient algorithms for fitting all possible subset regression models are required. Sequential strategies for computing the upper triangular $R_{(S)}$ and the corresponding $\mathrm{RSS}_{(S)}$ for all possible selection matrices S have been previously proposed [3,23]. Clarke developed an algorithm which derives all models based on a columns transposition strategy [3]. At each step of the algorithm two adjacent columns are transposed and a Givens rotation is applied to re-triangularize the resulted matrix. The order of transposition is important as it applies the minimum $2^n - n - 1$ Givens rotations. Smith and Bremner developed a dropping columns algorithm (DCA) which generates a regression tree [23]. The DCA applies the Givens rotations on matrices of smaller size and overall has less computational complexity. However, unlike Clarke's method, it requires intermediate storage [22]. The computations involved in these sequential methods are based on the re-triangularization of a matrix after interchanging or deleting columns.

Givens rotations can be efficiently applied to re-triangularize a matrix after it has been modified by one column or row [9,10]. Let the $m \times m$ Givens rotation $G_{i,j}^{(k)}$ have the structural form

$$G_{i,j}^{(k)} = \begin{array}{c} \begin{array}{cccccc} & i & & j & & \\ & \downarrow & & \downarrow & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array} \begin{array}{l} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \end{array} , \tag{5}$$

where $c^2 + s^2 = 1$. The Givens rotation $G_{i,j}^{(k)}$ is orthogonal and when applied from the left of a matrix, annihilates the kth element on the jth row and only the ith and jth rows are affected. Hereafter the Givens rotation $G_i \equiv G_{i,i-1}^{(i-i)}$. Constructing a Givens rotation requires six flops. The time to construct a Givens rotation will be denoted by t. The same time is required to apply the rotation to a 2-element vector [10]. Thus, $tn \equiv 6n$ flops are needed to annihilate an element of a $2 \times n$ non-zero matrix.

Parallel strategies for computing all possible subset regression models are investigated. An efficient parallelization of the DCA is proposed. Its extension for the general linear and seemingly unrelated regression models, and the case where new variables are added to the standard regression model are considered. Theoretical measures of complexity suggest linear speedup when $p = 2^\rho$ ($\rho \geqslant 0$) processors are used. All the parallel algorithms have been implemented using Fortran, BLAS and MPI on the shared memory SUN Enterprise 10,000 (16 CPU UltraSPARC of 400 MHz) [5]. The execution times in the experimental results are reported in seconds.

In Section 2 a formal description of the regression trees generated by the DCA is presented and their properties are investigated. The parallelization of the DCA together with an updating algorithm for generating all subset regression models are described in Section 3. Theoretical measures of complexity are derived. Section 4 considers the extension of the parallel DCA to the general linear and seemingly unrelated regression models. Conclusions and future work are presented and discussed in Section 5.

2. Regression trees

The DCA has been briefly discussed in [22,23]. Here a formal and detailed description is given. Let $M_{k,\lambda}^{v}$ denote the upper triangular factor in the QRD of an

exogenous matrix comprising the columns (variables) $v_1, \ldots, v_{k+\lambda}$. Furthermore, the index pair (k, λ) indicates that the columns $k+1, \ldots, k+\lambda-1$ will be deleted one at a time from the triangular (exogenous) matrix in order to obtain new models. Within this context the regression tree $T^v_{k,\lambda}$ defines an $(\lambda-1)$—tree having as root node $M^v_{k,\lambda}$ with the children $T^{v^{(k+i)}}_{k+i-1,\lambda-i}$ for $i = 1, \ldots, \lambda-1$. Here, $v^{(k+i)}$ denotes the vector v, without its $(k+i)$th element. Notice that $M^v_{k,\lambda}$ together with the modified response variable $Q^T y$ can provide the RSS of the sub-models comprising the variables $(v_1), (v_1 v_2), \ldots$, and $(v_1, \ldots, v_{k+\lambda})$. The models $(v_1), (v_1 v_2), \ldots, (v_1, \ldots, v_k)$ can be extracted from a parent node of the regression tree while the new sub-models provided by $M^v_{k,\lambda}$ are $(v_1, \ldots, v_{k+1}), \ldots, (v_1, \ldots, v_{k+\lambda})$. The derivation of a child node from its parent requires a re-triangularization of an upper triangular matrix after deleting a column using Givens rotations. The rotations are also applied on the modified response vector $Q^T y$. Emphasis will be given to the re-triangularization of the matrices. For simplicity the application of the Givens rotations on the response vector will not be discussed, but it will be taken into consideration for the complexity analysis. Fig. 1 shows the sequence of Givens rotations for re-triangularizing a 5×5 upper triangular matrix after deleting its second column. Shadowed frames indicate the submatrices affected by the Givens rotations at each stage of the re-triangularization.

The application of the DCA on the regression model (1) is equivalent to a leftmost walk on the regression tree $T^v_{0,n}$, where $M^v_{0,n} \equiv R$ in (2), $v_i = i$ and $i = 1, \ldots, n$. Fig. 2 shows $T^v_{0,5}$ together with the sub-models which can be extracted from each node. A sub-model is denoted by a sequence of numbers which corresponds to the variable indices. The operations *Drop* and *Shift* are used to derive a child node from its parent. Given $M^v_{k,\lambda}$ the Drop operation deletes the $(k+1)$th column, applies the $\lambda-1$ Givens rotations $G_{k+2}, \ldots, G_{k+\lambda}$ to re-triangularize the modified matrix and returns $M^{v^{(k+1)}}_{k,\lambda-1}$. Fig. 1 corresponds to the application of Drop on $M^v_{1,4}$ which returns $M^{v^{(2)}}_{1,3}$. Given $M^v_{k,\lambda}$ the Shift operation returns $M^v_{k+1,\lambda-1}$. That is, it simply modifies the index of the first column to be deleted from $M^v_{k,\lambda}$ by incrementing k and decrementing λ. From the parent $M^v_{k,\lambda}$ the ith child is obtained by applying $(i-1)$ Shifts followed by a Drop. For example, in Fig. 2, $M^{(2,4,5)}_{1,2}$ derives from $M^{(2,3,4,5)}_{0,4}$ after a Shift followed by a Drop. This indicates that the sub-models derived from the subtree $T^{v^{(k+i)}}_{k+i-1,\lambda-i}$ will always comprise the variables $v_1, \ldots, v_{k+i-1}$.

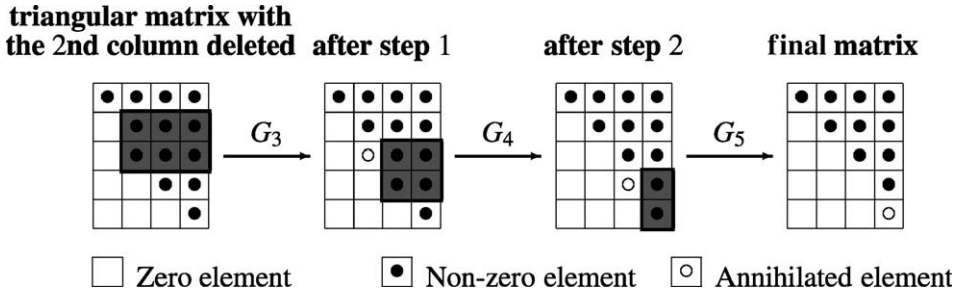

Fig. 1. Re-triangularization of an $n \times n$ upper triangular matrix after deleting the kth column using Givens rotations, where $n = 5$ and $k = 2$.

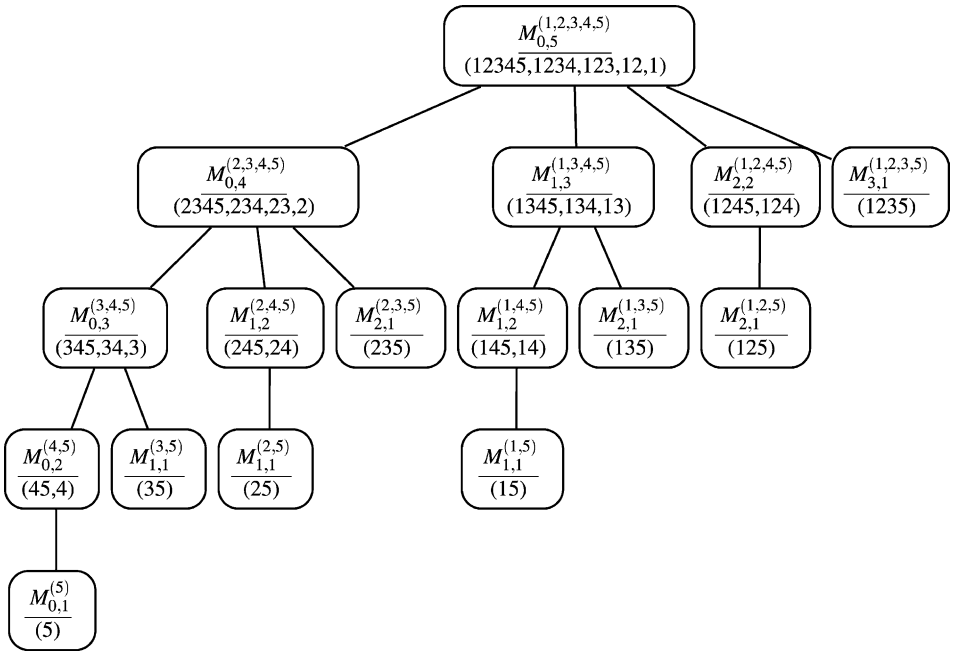

Fig. 2. The regression tree $T_{0,n}^{v}$, where $n = 5$ and $v = (1, \ldots, n)$.

The *SubTree* procedure shown in Algorithm 1 generates the regression tree $T_{k,\lambda}^{v}$ given as an argument the root node $M_{k,\lambda}^{v}$. Thus, the DCA for the n-variable model (1) is equivalent to SubTree $(M_{0,n}^{v})$, where $v = (1, \ldots, n)$ and in the QRD (2) $R \equiv M_{0,n}^{v}$. The application of Drop on $M_{k,\lambda+1}^{v}$ depends only on λ and has complexity

$$C_{\mathrm{Ret}}(\lambda) = t\sum_{j=1}^{\lambda}(j+1) = t(\lambda^2 + 3\lambda)/2. \tag{6}$$

Thus, the complexity of generating $T_{k,\lambda}^{v}$ with root node $M_{k,\lambda}^{v}$ is given by

$$\begin{aligned} C(\lambda) &= \sum_{i=1}^{\lambda-1}\left(C_{\mathrm{Ret}}(\lambda - i) + C(\lambda - i)\right) = C_{\mathrm{Ret}}(\lambda - 1) + 2C(\lambda - 1) \\ &= 2^{\lambda-1}C(1) + \sum_{i=1}^{\lambda-1} 2^{i-1}C_{\mathrm{Ret}}(\lambda - i). \end{aligned} \tag{7}$$

Now, since $C(1) = 0$ and using (6) in (7) it follows that

$$C(\lambda) = 3t2^{\lambda} - t(\lambda + 2)(\lambda + 3)/2. \tag{8}$$

Therefore, the complexity of the DCA is $O(2^n)$ and specifically given by

$$C_{\mathrm{DCA}}(n) = C(n) \approx 3t2^{n}.$$

Algorithm 1. Generating the regression tree $T^v_{k,\lambda}$ from the root node $M^v_{k,\lambda}$.

1: **procedure** SubTree ($M^v_{k,\lambda}$)
2: From $M^v_{k,\lambda}$ obtain the RSS of the sub-models $(v_1,\ldots,v_{k+1}),\ldots,(v_1,\ldots,v_{k+\lambda})$
3: **for** $i=1,\ldots,\lambda-1$ **do**
4: Store $M^v_{k,\lambda}$
5: $M^v_{k+i-1,\lambda-i+1} \leftarrow$ Apply $i-1$ Shifts on $M^v_{k,\lambda}$
6: $M^{v^{(k+i)}}_{k+i-1,\lambda-i} \leftarrow$ Apply Drop on $M^v_{k+i-1,\lambda-i+1}$
7: SubTree($M^{v^{(k+i)}}_{k+i-1,\lambda-i}$)
8: **end for**
9: **end procedure**

3. The parallel DCA

For the design of an efficient parallel DCA (hereafter PDCA), the properties of the regression trees need to be investigated and exploited. The number of nodes in the $(\lambda-1)-$tree $T^v_{k,\lambda}$ is given by

$$\delta_\lambda = 1+\sum_{i=1}^{\lambda-1}\delta_i = 2^{\lambda-1},$$

where δ_i $(i=1,\ldots,\lambda-1)$ is the number of nodes in the subtree $T^{v^{(k+\lambda-i)}}_{k+\lambda-i-1,i}$. Notice that,

$$\delta_{j+1} = 1+\sum_{i=1}^{j}\delta_i.$$

This indicates that the nodes of $T^v_{k,\lambda}$, excluding the root node, can be divided in two sets of nodes. The first set comprises the $\delta_{\lambda-1}$ nodes of the subtree $T^{v^{(k+1)}}_{k,\lambda-1}$ and the second set consists of the $\delta_{\lambda-1}-1$ nodes of the remaining subtrees. This property applies recursively for each resulted set. Thus, given $p=2^\rho$ processors ($\rho<n-1$), half of them are allocated to the $T^{v^{(k+1)}}_{k,\lambda-1}$ and the rest of the processors are allocated to the remaining of the tree, i.e. $T^v_{k+1,\lambda-1}$. This procedure is recursively applied to each $T^{v^{(k+1)}}_{k,\lambda-1}$ and $T^v_{k+1,\lambda-1}$ until each processor is allocated a unique subtree. These subtrees have the same complexity. Fig. 3 illustrates the case for $n=5$ and $p=4$. Dashed boxes indicate nodes derived from a Shift operation which requires no computation. The remaining nodes are obtained using a Drop operation. Notice that the number of Shifts and Drops performed by the processor P_r $(r=0,\ldots,2^\rho-1)$ is equal to the number of ones and zeros in the binary representation of r, respectively. Thus, P_0 and $P_{2^\rho-1}$ perform only Drops and Shifts, respectively. Shadowed boxes denote the subtrees which have the same complexity.

The PDCA uses a SPMD (single-program multiple-data) paradigm and is divided to *Mapping* and *Computation* phases [5]. Initially all the processors are allocated the parent node $M^v_{0,n}$. In the Mapping phase each processor performs a sequence of Drop or Shift operations until it has generated a unique node. In the Computation phase

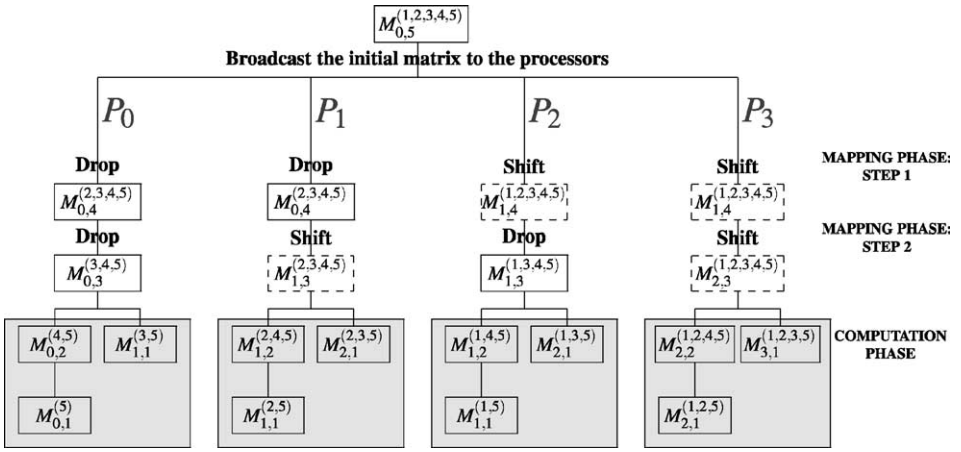

Fig. 3. The parallel computation of the $T_{0,5}^{v}$ using four processors (P_0, P_1, P_2 and P_3).

each processor uses this node to generate simultaneously $2^{n-\rho-1}-1$ $(0 \leqslant \rho < n-1)$ nodes. Algorithm 2 summarizes the steps of the PCDA, while Fig. 3 illustrates its execution on 4 processors, where $n = 5$. The initial computations in step 1 are not explicitly specified. All the processors can contribute to the computation of the QRD and obtain a copy of the triangular matrix, or one processor computes the QRD and broadcast the triangular factor to the remaining processors. The Mapping phase is shown in lines 4–11 of Algorithm 2 which is executed by each processor. Notice that a Drop generates a new node (re-triangularizing a matrix), while a Shift just changes the indices (k, λ) of the node. Furthermore, the time complexity of this phase is dominated by the first processor which performs ρ Drop operations and is given by

$$C_{\text{map}}(n,\rho) = \sum_{j=1}^{\rho} C_{\text{Ret}}(n-j) = t\rho(3n^2+6n-4-\rho(3n-\rho+3))/6.$$

Algorithm 2. The PDCA using $p = 2^{\rho}$ processors.

1: **initially do:**

- Compute the QRD of $A \in \mathbb{R}^{m\times n}$:

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}$$

- Obtain the RSS of the models $(v_1), \ldots, (v_1, \ldots, v_n)$
- Let $M_{k,\lambda}^{v} \equiv R$, where $v_i = i$ $(i = 1, \ldots, n)$, $k = 0$ and $\lambda = n$
- Broadcast $M_{k,\lambda}^{v}$ to the processors $P_0, \ldots, P_{p-1}$

2: $r \leftarrow$ rank of the processor
3: **each processor do:**
4: **for** $s = 1, \ldots, \rho$ **do**

5: **if** $((r \text{ div } 2^{\rho-s}) \bmod 2) = 0$ **then**
6: $M^v_{k,\lambda} \leftarrow$ Apply Drop on $M^v_{k,\lambda}$
7: Obtain the RSS of the models $(v_1, \ldots, v_{k+1}), \ldots, (v_1, \ldots, v_{k+\lambda})$
8: **else**
9: $M^v_{k,\lambda} \leftarrow$ Apply Shift on $M^v_{k,\lambda}$
10: **end if**
11: **end for**
12: **call** Subtree($M^v_{k,\lambda}$)
13: **end do**

The Computation phase executes the Subtree routine (see Algorithm 1) in line 12 of Algorithm 2. This has complexity $C(n-\rho)$—defined in (8)—and thus, the complexity of the PDCA is given by

$$C_{\text{PDCA}}(n,\rho) = C(n-\rho) + C_{\text{map}}(n,\rho) \approx 3t2^{n-\rho}.$$

Clearly the Computation phase dominates the PDCA which has an exponential complexity. Increasing the number of variables in the model by one it will require the doubling of the processors in order to achieve the same execution time. Even thought, when compared to the serial DCA the PDCA has almost a linear speedup for $\rho < n-1$ and large n, that is,

$$\text{Speedup}(n, 2^\rho) = C_{\text{DCA}}(n)/C_{\text{PDCA}}(n,\rho) \approx 2^\rho.$$

The theoretical measures of complexity do not take into account the overheads occurring during the implementation. These overheads are proportional to the dimension of the root matrix used in the Computation phase by each processor. Thus, if $i < j$, then the overheads of P_i are less than that of P_j, where $i, j = 0, \ldots, 2^\rho - 1$. This suggest that the load could be better balanced by expanding the Computation phase into a larger number of subtrees of smaller complexity and allocating these efficiently to the processors. Consider the case where each of the 2^ρ subtrees obtained after the Mapping phase is divided into 2^μ smaller subtrees. Thus, in the Computation phase 2^g subtrees, say $T_0, T_1, \ldots, T_{2^g-1}$, need to be computed, where $g = \mu + \rho$. The *shift cyclic* method can be used to allocate the computations of the subtrees to the processors. This adhoc distribution method allocates the subtree T_ζ to the processor P_ξ, where $\zeta = 0, \ldots, 2^g - 1$ and $\xi = (\zeta - (\zeta \div 2^\rho)) \bmod 2^\rho$. Fig. 4 shows the allocation of the subtrees to the processors, where $g = 4$ and $\rho = 2$.

This method, called PDCA-2, has been implemented with $\mu = \rho$. Table 1 shows the execution times of each processor when the PDCA and PDCA-2 are used, where $n = 25$ and $\rho = 3$. It can be observed that with PDCA-2 the load is better balanced.

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
P_0	P_1	P_2	P_3	P_3	P_0	P_1	P_2	P_2	P_3	P_0	P_1	P_1	P_2	P_3	P_0

Fig. 4. The cyclic allocation of 2^g subtrees on 2^ρ processors, where $g = 4$ and $\rho = 2$.

Table 1
The execution times of each processor using the PDCA and PDCA-2 , where $n = 25$ and $2^{\rho} = 8$

Processor	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
PDCA	89.70	94.56	94.48	99.00	94.46	99.01	98.96	102.89
PDCA-2	97.55	98.44	98.58	98.30	98.55	97.80	97.89	97.72

Table 2
Theoretical complexity and execution times of the DCA, PDCA and PDCA-2

n	2^{ρ}	DCA	Theoretical PDCA		PDCA		PDCA-2	
		Serial	Complx./t	Effi-ciency	Time	Effi-ciency	Time	Effi-ciency
15	1	0.600	98,151	1.00	0.603	0.99	0.610	0.98
15	2		49,135	0.99	0.313	0.97	0.310	0.98
15	4		24,679	0.99	0.160	0.94	0.157	0.96
15	8		12,496	0.98	0.080	0.94	0.080	0.94
19	1	10.49	1,572,633	1.00	10.60	0.99	10.54	0.99
19	2		786,411	0.99	5.41	0.97	5.31	0.99
19	4		393,385	0.99	2.80	0.94	2.66	0.98
19	8		196,948	0.99	1.40	0.94	1.37	0.96
20	1	21.52	3,145,475	1.00	21.52	1.00	21.53	0.99
20	2		1,572,842	0.99	11.03	0.98	10.81	0.99
20	4		786,620	0.99	5.63	0.96	5.46	0.98
20	8		393,594	0.99	2.88	0.93	2.78	0.97
21	1	43.49	6,291,180	1.00	43.59	0.99	43.55	0.99
21	2		3,145,705	0.99	22.54	0.96	22.03	0.98
21	4		1,573,072	0.99	11.41	0.95	11.16	0.96
21	8		786,850	0.99	5.83	0.93	5.47	0.98
25	1	757.28	100,662,900	1.00	759.89	0.99	773.71	0.98
25	2		50,331,621	0.99	389.56	0.97	381.54	0.99
25	4		25,166,122	0.99	201.12	0.94	190.97	0.99
25	8		12,583,510	0.99	102.89	0.92	98.55	0.97

Table 2 shows the execution time of the serial DCA, $C_{\mathrm{PDCA}}(n,\rho)/t$, the theoretical efficiency, i.e. $\mathrm{Speedup}(n,2^{\rho})/2^{\rho}$, the execution time and the actual efficiency of PDCA and PDCA-2 for various values of n and ρ. The speedup was calculated with respect to the serial time of the DCA. Clearly the PDCA-2 outperforms the PDCA and obtains an efficiency close to the theoretical derived value of 1. Furthermore, Table 2 shows clearly the doubling of the execution time when the number of variables is increased by one.

3.1. Variable updating of the regression model

The DCA and PDCA can be extended to solve the variable-updated regression model. Consider adding a new column, say z, to the regression model (1) for which

the RSS of all subset regression models have already been obtained. In this case the RSS of all 2^n new subset models which comprise the new variable need to be computed. Let the new variable be added at the front of the exogenous matrix. The DCA when applied to the new model will generate the regression tree $T^v_{0,n+1}$ in which the leftmost child $T^{v^{(1)}}_{0,n}$ corresponds to the regression tree derived by the DCA when applied to the original model. This is illustrated in Fig. 2, where now, $n = 4$. The root node of the regression tree derives from the QRD

$$\hat{Q}^T \begin{pmatrix} z_1 & R \\ \zeta & 0 \end{pmatrix} \equiv M^v_{0,n+1}, \tag{9}$$

where

$$Q^T z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad \tilde{Q}^T z_2 = \begin{pmatrix} \zeta \\ 0 \end{pmatrix} \begin{matrix} 1 \\ m-n-1 \end{matrix}.$$

Here $\tilde{Q}$ and $\hat{Q}$ are orthogonal, $M^v_{0,n+1}$ is upper triangular of order $n+1$ and $\zeta^2 = \|z\|^2$. The QRD (9) is computed by a sequence of n Givens rotations between adjacent planes that annihilate from bottom to top the elements of $(z_1^T\ \zeta^T)^T$ except from the first one. Without taking into account these Givens rotations, the complexity of the DCA to solve the updated model is half of that needed to solve the model afresh. The PDCA solves the updated model $M^v_{0,n+1}$ by generating $T^v_{1,n}$. The parallel complexity in this case is given by $C_{\mathrm{PDCA}}(n,\rho)$. Notice that if the new transformed variable $Q^T z$ is added at the end of the matrix, then $\hat{Q} = I_{n+1}$, i.e. the QRD (9) does not need to be computed. However, this advantage is offset by the non-trivial derivation of the new RSS from the regression tree generated using the DCA.

4. The general linear and seemingly unrelated regression models

The DCA can be employed to compute all possible subset models when the dispersion of the the noise vector ε in (1) is non-spherical. The general linear model (GLM) is the regression model (1), where ε has zero mean, variance–covariance matrix $\sigma^2\Omega$, σ is a non-zero scalar and Ω is non-negative definite. Let Ω be non-singular with Cholesky factorization $\Omega = BB^T$, where B is upper triangular. The GLM can be formulated as the generalized linear least squares problem (GLLSP):

$$\underset{\beta,v}{\operatorname{argmin}} \|v\|^2 \text{ subject to } y = A\beta + Bv, \tag{10}$$

where $\|\cdot\|$ denotes the Eucledian norm and v is a random m-element vector with zero mean and variance–covariance matrix $\sigma^2 I_m$. Consider the Generalized QRD (GQRD) of A and B:

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad Q^T B P = W \equiv \begin{pmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \tag{11}$$

where W and R are upper triangular and $Q, P \in \mathbb{R}^{m\times m}$ are orthogonal. Let

$$Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix} \quad \text{and} \quad P^T \upsilon = \begin{pmatrix} \upsilon_1 \\ \upsilon_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}.$$

From this it follows that the GLLSP is reduced to

$$\underset{\beta,\upsilon_1}{\operatorname{argmin}} \|\upsilon_1\|^2 \text{ subject to } \tilde{y} = R\beta + W_{11}\upsilon_1, \tag{12}$$

where $\upsilon_2 = W_{22}^{-1} y_2$ and $\tilde{y} = y_1 - W_{12}\upsilon_2$. Thus, $\upsilon_1 = 0$, the LS estimator $\hat{\beta} = R^{-1}\tilde{y}$ and the RSS is computed by $\|\upsilon_2\|^2$.

Notice that the modified GLM (3) is equivalent to the GLLSP

$$\underset{\beta,\upsilon_1}{\operatorname{argmin}} \|\upsilon_1\|^2 \text{ subject to } \tilde{y} = RS\beta_{(S)} + W_{11}\upsilon_1. \tag{13}$$

For the solution of (13) the GQRD of RS and W_{11} is required. That is, the QRD (4) and the RQD $(Q_{(S)}^T W_{11})P_{(S)} = \tilde{W}_{11}$, where $\tilde{W}_{11}$ is upper triangular. Within the context of the DCA the matrix RS is the single column-downdated R and $Q_{(S)}^T$ is a product of Givens rotations. In this case, a Givens rotation, say G_i, when applied from the left of $(RS\ W_{11})$ annihilates and fills-in the elements $(i, i-1)$ of RS and W_{11}, respectively. A Givens rotation, say P_i, can be applied from the right of $G_i W_{11}$ to annihilate the fill-in, that is, $G_i W_{11} P_i$ is upper triangular [15,18,19]. Thus, $Q_{(S)}^T$ and $P_{(S)}$ are the products of the left and right Givens rotations, respectively. Now, writing

$$Q_{(S)}^T RS = \begin{pmatrix} R_{(S)} \\ 0 \end{pmatrix} \begin{matrix} n-1 \\ 1 \end{matrix}, \quad Q_{(S)}^T \tilde{y} = \begin{pmatrix} \tilde{y}_{(S)} \\ \eta_{(S)} \end{pmatrix} \begin{matrix} n-1 \\ 1 \end{matrix}, \quad P_{(S)}^T \upsilon_1 = \begin{pmatrix} \upsilon_{(S)} \\ f_{(S)} \end{pmatrix} \begin{matrix} n-1 \\ 1 \end{matrix}$$

and

$$Q_{(S)}^T W_{11} P_{(S)} = W_{(S)} \equiv \begin{matrix} \\ n-1 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} n-1 & 1 \end{matrix} \\ \begin{pmatrix} W_{11}^* & w_{(S)} \\ 0 & \omega_{(S)} \end{pmatrix} \end{matrix},$$

it follows that the RSS and solution of the modified GLM is given by $\mathrm{RSS}_{(S)} = \mathrm{RSS} + \hat{f}_{(S)}^2$ and $\hat{\beta}_{(S)} = R_{(S)}^{-1}\tilde{y}_{(S)}^*$, where $\hat{f}_{(S)} = \eta_{(S)}/\omega_{(S)}$ and $\tilde{y}_{(S)}^* = \tilde{y}_{(S)} - w_{(S)}\hat{f}_{(S)}$. Notice that if $S = (I_{\bar{n}}\ 0^T)^T$, then $Q_{(S)} = P_{(S)} = I_n$ and $\mathrm{RSS}_{(S)} = \mathrm{RSS} + \|\hat{f}_{(S)}\|^2$, where now $\hat{f}_{(S)} = \omega_{(S)}^{-1}\eta_{(S)}$, $\eta_{(S)} \in \mathbb{R}^{n-\bar{n}}$ and $\omega_{(S)} \in \mathbb{R}^{(n-\bar{n})\times(n-\bar{n})}$.

Let $M_{k,\lambda+1}^v$ denote the triangular factor $R_{(S)}$ and its corresponding $W_{(S)}$ matrix. The Drop applied to this model derives $M_{k,\lambda}^v$ using λ left and right Givens rotations. The complexity of the jth $(j = 1, \ldots, \lambda)$ left and right rotation are given by $(2j+2)t$ and $(k+\lambda+3-j)t$, respectively. Thus, the complexities of deriving $M_{k,\lambda}^v$ from $M_{k,\lambda+1}^v$, and the regression tree $T_{k,\lambda}^v$ are given, respectively, by

$$C_{\mathrm{GRet}}(k,\lambda) = t\sum_{j=1}^{\lambda}(j+k+\lambda+5) = t\lambda(2k+3\lambda+11)/2 \tag{14}$$

and

$$\begin{aligned}C(k,\lambda) &= \sum_{i=1}^{\lambda-1}(C_{\mathrm{GRet}}(k+i-1,\lambda-i-1)+C(k+i-1,\lambda-i-1))\\ &= C_{\mathrm{GRet}}(k,\lambda-1)+C(k,\lambda-1)+C(k+1,\lambda-1)\\ &= \sum_{i=1}^{\lambda-1}\left(\sum_{j=1}^{i}\binom{i-1}{j-1}C_{\mathrm{GRet}}(k+j-1,\lambda-i)\right)\\ &\quad+\sum_{i=1}^{\lambda}\binom{\lambda-1}{i-1}C(k+i-1,1).\end{aligned}$$

Here $\lambda > 1$, $C(k,1) = 0$ and $\binom{m}{n} = m!/n!(m-n)!$. Using $C(k,1) = 0$ the latter becomes

$$\begin{aligned}C(k,\lambda) &= \sum_{i=1}^{\lambda-1}\left(\sum_{j=1}^{i}\binom{i-1}{j-1}C_{\mathrm{GRet}}(k+j-1,\lambda-i)\right)\\ &= t(2^{\lambda}(\lambda+2k+16)-(\lambda+1)(3\lambda+2k+12)-4)/2.\end{aligned} \tag{15}$$

Thus, the complexity of the DCA when employed to the GLM (denoted by GDCA) is given by $C(0,n)$ which has $O(n2^n)$, that is,

$$C_{\mathrm{GDCA}}(n) = C(0,n) \equiv t(2^n(n+16)-(n+1)(3n+12)-4)/2. \tag{16}$$

Now, consider the adaptation of the PDCA in the case of the GLM and call this GPDCA. In the Mapping phase of the GPDCA the last processor performs only Shifts (no computations), while the first processor performs only Drops and has the highest complexity which is given by

$$C_{\mathrm{GPm}}(\rho,n) = \sum_{j=1}^{\rho} C_{\mathrm{GRet}}(0,n-j) = t\rho(3n^2+8n-5-\rho(3n-\rho+4))/2.$$

The complexity of each processor during the Computation phase is given by $C(k,n-\rho)$, where k denotes the number of Shifts performed in the Mapping phase. Thus, the last processor $P_{2^{\rho}-1}$ which performs the maximum of ρ Shifts has in this phase the highest complexity $C(\rho,n-\rho)$. For $n \gg \rho$ the computations during the Mapping phase are negligible when compared to the exponential complexity of the Computation phase. In this case, the complexity of the GPDCA will be given by $C(\rho,n-\rho)$ which is of $O((n+\rho)2^{n-\rho})$. Specifically

$$\begin{aligned}C_{\mathrm{GPDCA}}(n,\rho) &= C(\rho,n-\rho)\\ &\equiv t(2^{n-\rho}(n+\rho+16)-3(n+1)(n+4)+\rho(4n-\rho+13)-4)/2.\end{aligned} \tag{17}$$

The speedup of the GPDCA is given by

$$\mathrm{Speedup}_G(n,2^{\rho}) = C_{\mathrm{GDCA}}(n)/C_{\mathrm{GPDCA}}(n,\rho) \approx n2^{\rho}/(n+\rho). \tag{18}$$

The GPDCA does not achieve a linear speedup as in the case of the theoretical PDCA. This is due to the different complexities of the subtrees allocated to each

Table 3
The execution times of each processor using the GPDCA and GPDCA-2 , where $n = 25$ and $2^{\rho} = 8$

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
GPDCA	219.00	223.23	223.50	240.23	223.64	240.24	239.85	246.68
GPDCA-2	227.97	229.37	228.45	229.33	229.09	229.76	229.08	230.96

Table 4
Theoretical complexity and execution times of the GDCA, GPDCA and GPDCA-2

n	2^{ρ}	GDCA	Theoretical GPDCA		GPDCA		GPDCA-2	
		Serial	Complx./t	Effi-ciency	Time	Effi-ciency	Time	Effi-ciency
15	1	1.430	50,7446	1.00	1.448	0.99	1.450	0.99
15	2		261,722	0.96	0.748	0.96	0.730	0.98
15	4		134,781	0.94	0.392	0.91	0.370	0.97
15	8		69,279	0.91	0.224	0.80	0.184	0.97
19	1	25.02	9,174,348	1.00	25.14	0.99	25.20	0.99
19	2		4,717,944	0.97	13.04	0.96	12.75	0.98
19	4		2,424,227	0.95	6.98	0.90	6.44	0.97
19	8		1,244,621	0.92	3.57	0.88	3.16	0.98
20	1	50.51	1,887,3610	1.00	50.62	0.99	50.66	0.99
20	2		9,698,616	0.97	26.11	0.97	25.68	0.98
20	4		4,980,069	0.94	13.81	0.91	12.94	0.98
20	8		2,555,281	0.92	6.90	0.91	6.44	0.98
21	1	103.19	38,796,485	1.00	104.22	0.99	103.50	0.99
21	2		19,922,165	0.97	54.46	0.95	52.02	0.98
21	4		10,222,884	0.95	27.75	0.93	26.74	0.96
21	8		5,242,194	0.93	14.57	0.88	13.18	0.98
25	1	1802.23	687,864,700	1.00	1815.98	0.99	1809.64	0.99
25	2		352,320,400	0.98	932.33	0.97	909.20	0.99
25	4		180,354,000	0.95	481.17	0.94	459.32	0.98
25	8		92,273,720	0.93	246.68	0.92	230.96	0.98

processor in the Computation phase. A better load balancing can be achieved using the same approach that has been employed by the PDCA-2. The efficiency of this strategy (GPDCA-2) compared to that of GPDCA is illustrated by the Tables 3 and 4. Notice that the efficiency obtained by the GPDCA-2 outperforms the theoretical one of the GPDCA.

4.1. Seemingly unrelated regression models

A special case of the GLM is the seemingly unrelated regression model. In this model the exogenous matrix A has the block-diagonal structure

$$\oplus_i^G A^{(i)} = \operatorname{diag}(A^{(1)}, \ldots, A^{(G)}) = \begin{pmatrix} A^{(1)} & & \\ & \ddots & \\ & & A^{(G)} \end{pmatrix},$$

$A^{(i)} \in \mathbb{R}^{m \times n_i}$ $(i = 1, \ldots, G)$, the variance–covariance matrix of the disturbances is given by $\Sigma \otimes I_m$ and $\Sigma \in \mathbb{R}^{G \times G}$ is positive-definite [4,14,16,24,25,27]. Here $\otimes$ and $\oplus$ are the Kronecker and direct sum of matrices operators [20]. Thus, in (12), and consequently (13), the upper triangular matrix $R = \operatorname{diag}(R^{(1)}, \ldots, R^{(G)})$, $R^{(i)} \in \mathbb{R}^{n_i \times n_i}$, $W_{11} \in \mathbb{R}^{n^{(G)} \times n^{(G)}}$ and $n^{(i)} = \sum_{j=1}^{i} n_j$, $(i = 1, \ldots, G)$. Let $RS_{i,j}$ denote the matrix R after deleting the jth column from $R^{(i)}$, where $i = 1, \ldots, G$ and $j = 1, \ldots, n_i$. Furthermore, let $\tilde{R}_j^{(i)}$ denote $R^{(i)}$ without its jth column and W_{11} be partitioned as

$$W_{11} = \begin{array}{c} \begin{matrix} n_1 & n_2 & \cdots & n_G \end{matrix} \\ \begin{pmatrix} \widetilde{W}_{1,1} & \widetilde{W}_{1,2} & \cdots & \widetilde{W}_{1,G} \\ \widetilde{W}_{2,1} & \widetilde{W}_{2,2} & \cdots & \widetilde{W}_{2,G} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{W}_{G,1} & \widetilde{W}_{G,2} & \cdots & \widetilde{W}_{G,G} \end{pmatrix} \end{array} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_G \end{matrix}, \tag{19}$$

where $i, j = 1, \ldots, G$. The re-triangularization of $RS_{i,j}$ is obtained in two stages. In the first stage, as in the case of the GLM, $n_i - j$ rotations between adjacent planes are applied from the left and right of $(\tilde{R}_j^{(i)} \widetilde{W}_{i,i}, \ldots, \widetilde{W}_{i,G})$ and $(\widetilde{W}_{i,1}^T, \ldots, \widetilde{W}_{i,i}^T)^T$, respectively, in order to re-triangularize $\tilde{R}_j^{(i)}$ and $\widetilde{W}_{i,1}$. Let $\widehat{Q}_{(S)}^T$ and $\widehat{P}_{(S)}$ denote the products of the left and right Givens rotations, respectively, and Π^T be the permutation matrix

$$\Pi^T = \begin{pmatrix} I_{n^{(i)}-1} & 0 & 0 \\ 0 & 0 & I_{n^{(G,i)}-1} \\ 0 & 1 & 0 \end{pmatrix},$$

where $n^{(G,i)} = n^{(G)} - n^{(i)}$. Thus,

$$\Pi^T \widehat{Q}_{(S)}^T RS_{i,j} = \begin{pmatrix} R_{(S)} \\ 0 \end{pmatrix} \begin{matrix} n^{(G)} - 1 \\ 1 \end{matrix},$$

where $R_{(S)} = \operatorname{diag}(R_{(S)}^{(1)}, \ldots, R_{(S)}^{(G)})$, $R_{(S)}^{(i)}$ is upper triangular and $R_{(S)}^{(q)} \equiv R^{(q)}$ for $q = 1, \ldots, G$ and $q \neq i$.

The second stage computes the RQD $W_{(S)} = \widehat{W} \tilde{P}_{(S)}$, where $\widehat{W} = \Pi^T \widehat{Q}_{(S)}^T W_{11} \widehat{P}_{(S)}$, $W_{(S)}$ and $\widehat{Q}_{(S)}^T W_{11} \widehat{P}_{(S)}$ are upper triangular and $\tilde{P}_{(S)}$ is the product of $n^{(G,i)}$ Givens rotations. The μth $(\mu = 1, \ldots, n^{(G,i)})$ rotation, say $\tilde{P}_\mu$, annihilates the $(n^{(i)} + \mu - 1)$th element of the last row of $\widehat{W}$ by rotating adjacent planes. This is illustrated in Fig. 5, where $G = 3$, $n_1 = 4$, $n_2 = 6$, $n_3 = 3$, $i = 2$ and $j = 3$. An arc denotes the affected columns during the rotation.

The complexity of re-triangularizing $RS_{i,j}$ and that of generating all 2^{n_i} possible models by deleting one or more column from the ith block of the SUR model are given, respectively, by

$$C_{\mathrm{SRet}}(i, j) = t(j^2 + j(2n^{(G)} + 7) + n^{(G,i)}(n^{(G)} + n^{(i)} - 5) - 2n^{(i)} - 8)/2$$

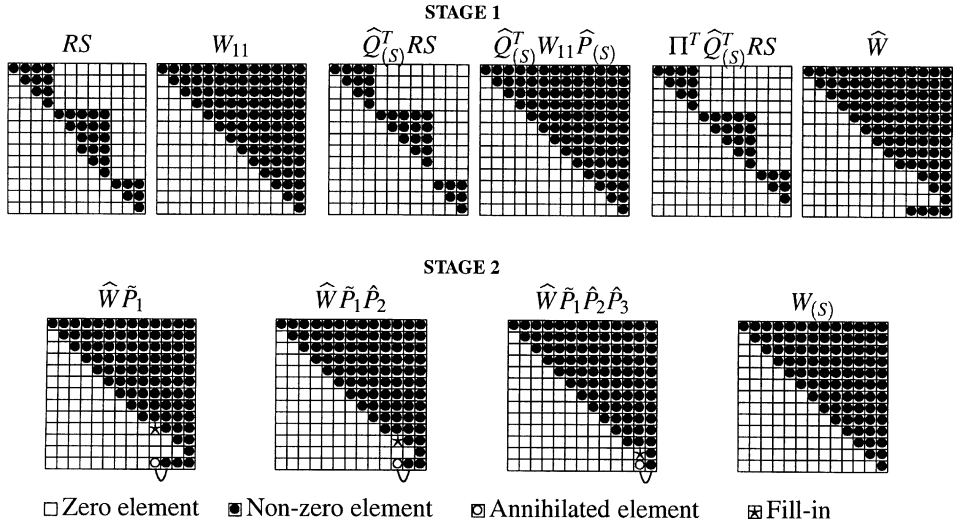

Fig. 5. The two stages of solving a seemingly unrelated regression model after deleting a variable.

and

$$C_{\mathrm{SGenB}}(i) \approx t2^{n_i}((n^{(G,i)}+2)(n^{(G)}+n^{(i-1)}+2)+28).$$

Thus, the complexity of the DCA applied to seemingly unrelated regression models (SDCA) is given by

$$\begin{aligned} C_{\mathrm{SDCA}}(G,N) &= \sum_{i=1}^{G-1}\left(\sum_{j=0}^{n^{(i-1)}}\binom{n^{(i-1)}}{j}C_{\mathrm{SGenB}}(i+1)\right)+\sum_{j=0}^{n^{(G-1)}}\binom{n^{(G-1)}}{j}C_{\mathrm{SGenB}}(G) \\ &\approx t\sum_{i=1}^{G-1}\left(2^{n^{(i)}+1}((n^{(G,i)}+1)(n^{(G)}+2)+28)\right)+t2^{n^{(G)}+1}(n^{(G)}+11). \end{aligned}$$

5. Conclusions

A parallel algorithm has been developed to compute the RSS of all possible subset models of the standard regression model. The algorithm (PDCA) is a parallelization of the DCA proposed in [22,23]. The properties of the regression tree generated by the DCA have been studied in order to derive an efficient load-balancing strategy. The PDCA uses a single-program multiple-data paradigm and requires no inter-processor communication. The theoretical measures of complexity had showed that the PDCA has a linear speedup. Experimental results on a shared memory machine indicated that overheads cause a non-perfect load balancing among the processors. This resulted in the efficiency of the PDCA to divert from the theoretical one. A second algorithm (PDCA-2) which obtains a better load balancing and an efficiency close to the maximum theoretical value of one has been designed.

The DCA and PDCA have been extended to GDCA and GPDCA, respectively, in order to compute the RSS of all possible subset models of the GLM. In this case the theoretical complexity has shown that the speedup obtained by the GPDCA is lower than that obtained by the PDCA. The main reason for this is that unlike the PDCA, the GPDCA allocates to the processors subtrees of different complexity. However, GPDCA-2, which is an extension of PDCA-2, has achieved an efficiency closed to one.

The adaptation of the serial GDCA to solve the seemingly unrelated regression model has also been developed. The employment of this algorithm and its parallelization for estimating all subsets of a seemingly unrelated regression model arising in some vector autoregressive processes are currently considered. In this case $R^{(1)} = \cdots = R^{(G)}$ have dimension $n \times n$ and in (19) $W_{11} = B \otimes I_n$, where $B \in \mathbb{R}^{G \times G}$.

The proposed algorithms will be inefficient for heterogeneous parallel systems. In such platforms a dynamic distribution, such as that obtained by task-farming, can yield a better performance. Furthermore, it will be computationally not feasible to consider all models when the number of variables, i.e. n, is very big. In such cases a parallel procedure which computes the best subset without examining all the possible subsets needs to be developed. A possibility is to use a branch and bound algorithm based on some criteria (statistics), or some other heuristic approaches [6,7,11,12,26]. Currently these non-trivial parallel strategies, the extension of the existing algorithms to other linear models (e.g. mixed and simultaneous equation models) and the adaptation of the PDCA to multiple-row diagnostics are investigated [1,8].

References

[1] D.A. Belsley, E. Kuh, R.E. Welsch, Regression Diagnostics: Identifying Influential Observations and Sources of Collinearity, John Wiley and Sons, 1980.
[2] Å. Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, 1996.
[3] M.R.B. Clarke, Algorithm AS163. A Givens algorithm for moving from one linear model to another without going back to the data, Applied Statistics 30 (2) (1981) 198–203.
[4] P. Foschi, E.J. Kontoghiorghes, Seemingly unrelated regression model with unequal size observations: computational aspects, Computational Statistics and Data Analysis 41 (2002) 211–229.
[5] I. Foster, Designing and Building Parallel Programs, Addison-Wesley, 1995.
[6] G.M. Fournival, R.W. Wilson Jr., Regression by leaps and bounds, Technometrics 16 (4) (1974) 499–511.
[7] M.J. Garside, Some computational procedures for the best subset problem, Applied Statistics 20 (1971) 8–15.
[8] C. Gatu, E.J. Kontoghiorghes, A branch and bound algorithm for computing the best subset regression models using the QR decomposition, Technical Report RT-2002/08-1, Institut d'informatique, Université de Neuchâtel, Switzerland, 2002.
[9] P.E. Gill, G.H. Golub, W. Murray, M.A. Saunders, Methods for modifying matrix factorizations, Mathematics of Computation 28 (126) (1974) 505–535.
[10] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, Maryland, 1996.
[11] R.R. Hocking, Criteria for selection of a subset regression: which one should be used? Technometrics 14 (4) (1972) 967–970.

[12] R.R. Hocking, The analysis and selection of variables in linear regression, Biometrics 32 (1976) 1–49.
[13] E.J. Kontoghiorghes, Parallel strategies for computing the orthogonal factorizations used in the estimation of econometric models, Algorithmica 25 (1999) 58–74.
[14] E.J. Kontoghiorghes, Parallel Algorithms for Linear Models: Numerical Methods and Estimation Problems, vol. 15, Advances in Computational Economics, Kluwer Academic Publishers, Boston, MA, 2000.
[15] E.J. Kontoghiorghes, Parallel Givens sequences for solving the general linear model on a EREW PRAM, Parallel Algorithms and Applications 15 (1–2) (2000) 57–75.
[16] E.J. Kontoghiorghes, Computational methods for modifying seemingly unrelated regressions models, Journal of Computational and Applied Mathematics, in press.
[17] E.J. Kontoghiorghes, M.R.B. Clarke, Parallel reorthogonalization of the QR decomposition after deleting columns, Parallel Computing 19 (6) (1993) 703–707.
[18] C.C. Paige, Numerically stable computations for general univariate linear models, Communications in Statistics B 7 (1978) 437–453.
[19] C.C. Paige, Fast numerically stable computations for generalized least squares problems, SIAM Journal on Numerical Analysis 16 (1979) 165–171.
[20] P.A. Regalia, S.K. Mitra, Kronecker products, unitary matrices and signal processing applications, SIAM Review 31 (4) (1989) 586–613.
[21] S.R. Searle, Linear Models, John Wiley and Sons Inc., 1971.
[22] D.M. Smith, Regression using QR decomposition methods, PhD Thesis, University of Kent, UK, 1991.
[23] D.M. Smith, J.M. Bremner, All possible subset regressions using the QR decomposition, Computational Statistics and Data Analysis 7 (1989) 217–235.
[24] V.K. Srivastava, T.D. Dwivedi, Estimation of seemingly unrelated regression equations models: a brief survey, Journal of Econometrics 10 (1979) 15–32.
[25] V.K. Srivastava, D.E.A. Giles, Seemingly Unrelated Regression Equations Models: Estimation and Inference (Statistics: Textbooks and Monographs), vol. 80, Marcel Dekker Inc., 1987.
[26] A. Sudjianto, G.S. Wasserman, H. Sudarbo, Genetic subset regression, Computers & Industrial Engineering 30 (4) (1996) 839–849.
[27] A. Zellner, An efficient method of estimating seemingly unrelated regression equations and tests for aggregation bias, Journal of the American Statistical Association 57 (1962) 348–368.