Pekka J. Korhonen – Pyry-Antti Siitari

A DIMENSIONAL DECOMPOSITION APPROACH TO IDENTIFYING EFFICIENT UNITS IN LARGE-SCALE DEA MODELS

HELSINKI SCHOOL OF ECONOMICS
WORKING PAPERS
W-421

Pekka J. Korhonen – Pyry-Antti Siitari

A DIMENSIONAL DECOMPOSITION APPROACH TO IDENTIFYING EFFICIENT UNITS IN LARGE-SCALE DEA MODELS

Management Science

March
2007

HELSINGIN KAUPPAKORKEAKOULU
HELSINKI SCHOOL OF ECONOMICS
WORKING PAPERS
W-421

HELSINGIN KAUPPAKORKEAKOULU
HELSINKI SCHOOL OF ECONOMICS
PL 1210
FI-00101 HELSINKI
FINLAND

ISSN 1235-5674
(Electronic working paper)
ISBN 978-952-488-124-1

Helsinki School of Economics -
HSE Print 2007

April 19, 2007

A DIMENSIONAL DECOMPOSITION APPROACH TO IDENTIFYING EFFICIENT UNITS IN LARGE-SCALE DEA MODELS

Pekka J. Korhonen & Pyry-Antti Siitari

(March 2007)

Helsinki School of Economics
Department of Business Technology
P.O. Box 1210, 00101 Helsinki, FINLAND
Tel. +358-9-431 31
Fax. +358-9-431 38535
E-mail: Pekka.Korhonen@hkkk.fi / Siitari@hkkk.fi

The research was supported by the Academy of Finland.

Acknowledgements: We would like to thank Prof. Jose Dulá for useful discussions and providing us with the data files we used in our performance tests. We also thank Dr. Victor Podinovski and Ms. Luiza Badin for their useful comments

ABSTRACT

In this paper, we propose the use of a dimensional decomposition procedure together with lexicographic parametric programming to reduce computational burden when identifying the efficient decision making units in Data Envelopment Analysis (DEA). The use of lexicographic parametric programming makes it possible to develop an efficient algorithm for the problems with few inputs and outputs. That's why we propose the procedure which first partitions the original problem dimensionally into sub-problems and then identifies the efficient units of the sub-problems. Because those units are a subset of the weakly efficient solutions of the original problem, they are used as an initial approximation for the efficient units of the original problem. The efficiency of the approach is illustrated by numerical results.

Keywords: Efficiency Analysis, Data Envelopment Analysis, Decomposition, Computational Aspects

1 INTRODUCTION

Data Envelopment Analysis (DEA) was originally proposed by Charnes, Cooper and Rhodes [1978 and 1979] as a method for evaluating the relative (technical) efficiency of different Decision Making Units (DMUs) essentially performing the same task. Each of the units uses multiple inputs to produce multiple outputs. The units are assumed to operate under similar conditions. The DEA is value-free in the sense that efficiency evaluation is based on the data available without taking into account the decision maker's (DM) preferences.

Based on information about existing data on the input/output-values of the units and some preliminary assumptions, a so-called production possibility set (PPS) is specified. The production possibility set consists of all possible input/output combinations. A specific part of the production possibility set is called an efficient frontier (surface). The input/output – values on the efficient frontier have a property that none of the inputs or outputs can be improved without worsening at least one input or output. If a DMU lies on the efficient frontier, it is referred to as an efficient unit, otherwise it is considered inefficient. DEA also provides efficiency scores and reference set for inefficient DMUs. The efficiency scores represent a degree of inefficiency of the DMUs. The reference set for inefficient units consists of a set of efficient units and determines a virtual target unit on the efficient frontier.

The target unit is found in DEA by projecting an inefficient DMU radially[1] to the efficient surface. To check the efficiency of a unit and to find the reference set and the efficiency score for inefficient units requires the solving of an LP-model. The straightforward method is to solve iteratively an LP-model for each unit separately. The optimal basis of the previous iteration is not valid for the next iteration as such, because at each iteration, the right-hand side vector and one column in the coefficient matrix has to be updated.

[1] Term "radial" in the traditional DEA-literature means that an efficient frontier is tried to reach by increasing the values of the current outputs or decreasing the values of the current inputs.

When the number of the units is tens or even hundreds of thousands, computationally more efficient methods are needed. Such problems appear when, for example, the efficiency analysis is made at an individual level. In these kinds of problems, the straightforward approach to formulate an LP-model for each unit with an unknown status does not work. It is too time-consuming. Fortunately, the structure of the DEA-model makes it possible to develop special techniques for large-scale problems.

There are not very many authors who have studied computational problems in DEA. One of those exceptions is Ali [1993, 1994], who proposed an idea to restrict the basis entry. The basis always consists of a set of existing efficient units. When a unit is diagnosed inefficient, the corresponding column can be dropped from the set of potential basic vectors. Moreover, Dulá and Helgason [1996] proposed solving the problem in two phases. In phase I, the extreme point solutions of the polytope consisting of all units in the data set are defined. The efficiency scores of the other vectors are computed in phase II by using the minimal set of potential basic vectors, i.e. efficient units. The idea was further developed in the paper by Dulá *et al.* [1997]. The more recent developments by Dulá and his associates are presented in Dulá and López [2002].

Because the computing time as the function of the units increases more than linearly, Barr and Durchholz [1997] proposed the partition of the set of the units. The efficient units are first identified in a small data set, and then the union of those units is used to build a set of potential basic vectors for the original problem. The union consists of all efficient units, but usually also inefficient units. The idea leads to computationally efficient algorithms, because the savings in computing time are remarkable in comparison to solving the original problem in one phase.

In our previous paper (Korhonen and Siitari [2007]), we used lexicographic parametric programming (Korhonen and Halme [1996]) to traverse from unit to unit along the efficient frontier. On the way, it was possible to recognize a set of units efficient or inefficient. The procedure is computationally efficient, when the number of inputs and outputs is small.

In this paper, we propose a dimensional decomposition to improve the performance of the procedure by Korhonen and Siitari [2007]. An original set is divided dimensionally into subsets, and the efficient units are identified in each subset. Those units are the weakly efficient units of the original problem. However, they provide a good initial approximation for the set of all efficient units of the original problem. The size of that set is usually much smaller than the number of the original units. The set is supplemented during the solution procedure with the units which are diagnosed super-efficient (Andersen and Petersen [1993]). Finally, the efficient units of the original problem are recognized from this supplemented set.

The paper is given in five sections. In the next section, the necessary theoretical questions are discussed. Section 3 represents the algorithm developed in this paper. Computational results are given and discussed in section 4. Section 5 concludes the paper with some remarks.

2 THEORETICAL CONSIDERATIONS

2.1 Preliminary Considerations

Consider a production technology, where m inputs are needed to produce p outputs. We denote inputs by $\boldsymbol{x} \in \Re^m_+$ and outputs by $\boldsymbol{y} \in \Re^p_+$. We define the production possibility set (PPS):

$$\mathrm{P}^0 = \{(\boldsymbol{y}, \boldsymbol{x}) \mid \boldsymbol{y} \text{ can be produced from } \boldsymbol{x}\} \in \Re^{p+m}_+ \qquad (2.1)$$

which consists of all feasible inputs and outputs in the general sense that the inputs are capable of producing the outputs. We assume that both inputs and outputs are freely disposable. As usually, we assume that in outputs more is better and in inputs less is better.

In practice, set P^0 is unknown. To approximate P^0, we usually gather sample information about the existing units, set up some assumptions, and define a set P, which is assumed to be a subset of P^0.

Now we are ready to define some efficiency concepts for the elements of the production possibility set P.

Definition 1. A point $(\boldsymbol{y}^*, \boldsymbol{x}^*) \in \mathrm{P}$ is *efficient* in set P iff (if and only if) there does not exist another $(\boldsymbol{y}, \boldsymbol{x}) \in \mathrm{P}$ such that $\boldsymbol{y} \geq \boldsymbol{y}^*, \boldsymbol{x} \leq \boldsymbol{x}^*$ and $(\boldsymbol{y}, \boldsymbol{x}) \neq (\boldsymbol{y}^*, \boldsymbol{x}^*)$.

Definition 2. A point $(\boldsymbol{y}^*, \boldsymbol{x}^*) \in \mathrm{P}$ is *weakly efficient* in set P iff there does not exist another $(\boldsymbol{y}, \boldsymbol{x}) \in \mathrm{P}$ such that $\boldsymbol{y} > \boldsymbol{y}^*$ and $\boldsymbol{x} < \boldsymbol{x}^*$.

Definition 3. A point $(\boldsymbol{y}^*, \boldsymbol{x}^*) \in \mathrm{P}^0 - \mathrm{P}$ is *super-efficient* with respect to set P iff there does not exist $(\boldsymbol{y}, \boldsymbol{x}) \in \mathrm{P}$ such that $\boldsymbol{y} \geq \boldsymbol{y}^*, \boldsymbol{x} \leq \boldsymbol{x}^*$.

To simplify notations in the following considerations, we define set Q:

$$(\boldsymbol{y}, \boldsymbol{x}) \in \mathrm{P} \Leftrightarrow (\boldsymbol{y}, -\boldsymbol{x}) \in \mathrm{Q}, \qquad (2.2)$$

and a dimensional chip of Q: $\mathrm{Q_B} \in \Re^h_+$, $h < p + m$

$$\mathrm{Q_B} = \{q_B \mid q = \begin{pmatrix} q_A \\ q_B \end{pmatrix} \in \mathrm{Q}\}\,. \qquad (2.3)$$

Theorem 1. If $q_B^* \in \mathrm{Q_B}$ is weakly efficient in $\mathrm{Q_B}$, then all $q^* = \begin{pmatrix} q_A^* \\ q_B^* \end{pmatrix} \in \mathrm{Q}$ are weakly efficient in Q.

Proof: Assume $q^* = \begin{pmatrix} q_A^* \\ q_B^* \end{pmatrix} \in \mathrm{Q}$ is not weakly efficient. Then there is another $q = \begin{pmatrix} q_A \\ q_B \end{pmatrix} \in \mathrm{Q}$ such that $q > q^*$. However, it leads to the contradiction with the assumption that $q_B^* \in \mathrm{Q_B}$ is weakly efficient in $\mathrm{Q_B}$, because there cannot exist another $q_B \in \mathrm{Q_B}$ such that $q_B > q_B^*$ □

Remark 1. If point q_B^* is efficient in Q$_B$, q^* is weakly efficient in Q, but not necessarily efficient.
Remark 2. If point q^* is (weakly) efficient in Q, then point q_B^* may be (weakly) efficient or inefficient in Q$_B$.

2.2 Basic Data Envelopment Models

Assume we have n DMUs each consuming m inputs ($m \neq 0$), and producing p outputs ($p \neq 0$). Information can be presented by two matrices. Let $\mathbf{X}$ be an ($m \times n$) - matrix and $\mathbf{Y}$ be a ($p \times n$) - matrix consisting of non-negative elements, containing observed input and output measures for the DMUs, respectively. We denote by $\boldsymbol{x}_j$ (the jth column of $\mathbf{X}$) the vector of inputs consumed by DMU$_j$, and by x_{ij} the quantity of input i consumed by DMU$_j$. A similar notation is used for outputs. We further assume that $\boldsymbol{x}_j \neq \boldsymbol{0}$ and $\boldsymbol{y}_j \neq \boldsymbol{0}$, $j = 1, 2, \ldots, n$, and for simplicity, we assume that there are no duplicates in the data set. Furthermore, we denote $\boldsymbol{1} = [1, ..., 1]^{\mathrm{T}}$.

Using the existing data set, we define the (practical) production possibility set as a set P $=\{(\boldsymbol{y}, \boldsymbol{x}) \mid \boldsymbol{x} \geq \mathbf{X}\boldsymbol{\lambda}, \boldsymbol{y} \leq \mathbf{Y}\boldsymbol{\lambda}, \boldsymbol{\lambda} \in \Lambda\}$, where the definition of Λ depends on the assumptions made about the form of set P.

Data Envelopment Analysis models can be interpreted as tools to check which existing or virtual units are on the efficient frontier of P. The traditional CCR-models, as introduced by Charnes *et al.* [1978, 1979] are originally presented as fractional linear programs which can easily be formulated and solved as linear programs. Those models are so-called constant returns to scale models. Later Banker, Charnes and Cooper [1984] developed the so-called BCC models with variable returns to scale[2]. The CCR and BCC models are the basic model types in DEA. Those basic models can be presented in a primal or dual form. Which one is primal or dual varies. It is more recommendable to call them multiplier and envelopment models accordingly. The multiplier model provides information on the weights of inputs and outputs which are interpreted as prices in many applications. Instead, the envelopment models provide the user with information on the lacks of outputs and the surplus of inputs of a unit. Moreover, the envelopment model characterizes the reference set for inefficient units.

The models can be further classified as input-oriented or output-oriented models. The model is input-oriented, when the output levels are given, and the inputs are tried to reduce radially in the production feasibility set. In the output-oriented models, the role of inputs and outputs are changed with the exception that the outputs are tried to increase. Without loss of generality, we may also consider a DEA-model by using a general directional vector $\boldsymbol{w} = \begin{pmatrix} \boldsymbol{w}^y \\ \boldsymbol{w}^x \end{pmatrix} \geq \boldsymbol{0}$, $\boldsymbol{w} \neq \boldsymbol{0}$ (discussion on directional distance functions, see Chambers *et al.* [1998]). Halme *et al.* [1999] called the model a general combined model. Input- and output-oriented models are the special cases of that model.

Consider the following lexicographic formulation of the general DEA-model in the so-called envelopment form (Korhonen and Siitari [2007]):

[2] Actually, the BCC model was first proposed by Afriat [1972, p.581], in a case of one output. The general version of the envelopment BCC model was independently also proposed, implemented and solved by Färe, Grosskopf and Logan [1983].

$$
\begin{aligned}
&\textit{lex max}\ \{\sigma,\ \boldsymbol{1}^T\boldsymbol{s}^+ + \boldsymbol{1}^T\boldsymbol{s}^-)\\
&\text{s.t.} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (2.4)\\
&\mathbf{Y}\lambda - \sigma \boldsymbol{w}^y - \boldsymbol{s}^+ = \boldsymbol{y}_0\\
&\mathbf{X}\lambda + \sigma \boldsymbol{w}^x + \boldsymbol{s}^- = \boldsymbol{x}_0\\
&\quad \lambda \in \Lambda\\
&\quad \lambda,\ \mathbf{s}^-,\ \mathbf{s}^+ \geq \boldsymbol{0}
\end{aligned}
$$

where $\boldsymbol{x}_0$ is the input-vector and $\boldsymbol{y}_0$ is the output-vector of a DMU under consideration and

$$
\Lambda = \begin{cases}
\{\lambda \mid \boldsymbol{1}'\lambda = 1, \lambda \geq \boldsymbol{0}\} \text{ for variable returns to scale model (Banker et al. [1984])}\\
\{\lambda \mid \boldsymbol{1}'\lambda \leq 1, \lambda \geq \boldsymbol{0}\} \text{ for non-increasing returns to scale model}\\
\{\lambda \mid \boldsymbol{1}'\lambda \geq 1, \lambda \geq \boldsymbol{0}\} \text{ for non-decreasing returns to scale model}\\
\{\lambda \mid \lambda \geq \boldsymbol{0}\} \text{ for constant returns to scale model (Charnes \textit{et al.} [1978]).}
\end{cases}
$$

Notation "*lex max*" refers to a lexicographic maximization problem. It means that we first solve (2.4) using σ as an objective function. In case, the optimal solution σ^* is not unique, we add the constraint $\sigma = \sigma^*$ into the model (2.4) and solve it by using $\boldsymbol{1}^T\boldsymbol{s}^+ + \boldsymbol{1}^T\boldsymbol{s}^-$ as the objective function.

The first three constraints for λ specifies one of the BCC - models, and the last one specifies the CCR-model. In the combined model, $\boldsymbol{w}^y = \boldsymbol{y}_0$ and $\boldsymbol{w}^x = \boldsymbol{x}_0$. In the input-oriented model $\boldsymbol{w}^y = \boldsymbol{0}$ and $\boldsymbol{w}^x = \boldsymbol{x}_0$, and in the output-oriented model $\boldsymbol{w}^x = \boldsymbol{0}$ and $\boldsymbol{w}^y = \boldsymbol{y}_0$. The value of σ - called an inefficiency score - at the optimum is denoted by σ^*. A DMU is efficient iff the optimal value σ^* and all slack variables $\boldsymbol{s}^-$, $\boldsymbol{s}^+$ equal zero. Otherwise, the DMU is inefficient (Charnes *et al.* 1994). For weakly-efficient solutions $\sigma^* = 0$ but $(\boldsymbol{1}^T\boldsymbol{s}^+ + \boldsymbol{1}^T\boldsymbol{s}^-) > 0$.

2.3 Problem Decomposition

Let N be the index set of n DMUs (N = {1, 2, …, n}) in the original problem. Let I be the index set of the rows of X (I = {1, 2, …, m}) and let O be the index set of the rows of Y (O = {1, 2, …, p}). By dimensional decomposition we mean that the set of input variables I and the set of output variables O are divided into q subsets: $I = I_1 \cup I_2 \cup \ldots \cup I_q$, $I_i \neq \emptyset$, and $O = O_1 \cup O_2 \cup \ldots \cup O_q$, $O_i \neq \emptyset$. The sets are not necessarily disjoint. Each pair of sets (I_i, O_i) defines a DEA sub-problem A_i (i = 1, 2 ,…, q). Each sub-problem specifies a status for each DMU_j, j = 1, 2, …, n. The set of efficient DMUs in sub-problem A_i is denoted by K_i^E, the set of only weakly efficient DMUs by K_i^W and the set of super-efficient DMUs by K_i^S. We drop the subscript, when we refer to the original problem.

In the proposed algorithm, each $DMU_j, j \in N$ is classified to one sub-problem A_i (i = 1, 2, …, q). The index set N_i refers to the DMUs belonging to sub-problem A_i.

We denote $A_r \subseteq A_v$, if $I_r \subseteq I_v$ and $O_r \subseteq O_v$. In this case, problem A_v has at least all the variables that the problem A_r contains. The problem A_r is a dimensional chip of the problem A_v. Each unit in the problem A_r can be interpreted as a vector $\boldsymbol{q}_B$ in (2.3). Similarly the production possibility set of A_r relates to the production possibility set of A_v as the vector set Q_B relates to Q in (2.3). By Theorem 1, if $A_r \subseteq A_v$, then $K_r^E \subseteq (K_v^W \cup K_v^E)$.

By evaluating dimensionally decomposed problems we can expect to find several units that are efficient also in the original (full dimensional) problem. Using these units as potential basic variables we can classify all the other DMUs either inefficient or super-efficient with respect to our new frontier.

3 THE ALGORITHM

3.1 Decomposition algorithm

The problem with $m + p > 1$ variables can be decomposed many ways. It is important to choose the decomposition in such a way that we get "efficiency information" as much as possible. If e.g. $\Lambda = \{\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \geq \boldsymbol{0}\}$, then in case we choose all variables to be maximized or to be minimized, all units are inefficient. This kind of decomposition does not give us much useful information. In advance, it is hard to say which decomposition is best. If in each chip there is at least one input to be minimized and one output to be maximized, then each sub-problem has at least one efficient unit. In addition, it is reasonable to minimize the overlapping of the variables between the sub-problems. Motivation to decompose the problem dimensionally is to recognize different efficient DMUs of the original problem. That's why it is justified to try to have as diverse set of variables in each problem as possible.

In this paper we have used the following algorithm to divide the set of input variables I and the set of output variables O into q subsets $I_1, I_2, \ldots, I_q$ and $O_1, O_2, \ldots, O_q$.

```
Step 0: Initialization
        Select block-quantity $q$ ($1 \leq q \leq \max(m, p)$) which determines how many sub-
        problems will be created. The value for $q$ will be determined empirically.
        Define two temporary index sets $I_t$ and $O_t$.

Step 1: Decomposing input variables
        Set $I_t := I$. Define $k := 1$.
        Set allInputsUsed := false and allSubsetsUsed := false.

        Step 1a:
                Select the first variable $i$ from the index set $I_t$. Redefine $I_k := I_k + i$,
                $I_t := I_t - i$ and $k := k + 1$.
        Step 1b:
                If ($I_t = \varnothing$) (if all input variables have been used at least once)
                        Redefine $I_t := I$. Set allInputsUsed := true.
                If ($k = q$) (if all subsets have at least one input)
                        Redefine $k := 1$. Set allSubsetsUsed := true.
                If (allInputsUsed = true) and (allSubsetsUsed = true)
                        Go to Step 2.
                Else
```

```
            Go to Step 1a.

Step 2: Decomposing output variables
    Set $O_t$ := O. Define $k := q$.
    Set allOutputsUsed := false and allSubsetsUsed := false.

    Step 2a:
        Select the first variable $o$ from the index set $O_t$. Redefine $O_k := O_k + o$,
        $O_t := O_t - o$ and $k := k - 1$.
    Step 2b:
        If ($O_t = \varnothing$) (if all output variables have been used at least once)
            Redefine $O_t$ := O. Set allOutputsUsed := true.
        If (k = 0) (if all subsets have at least one output)
            Redefine $k := q$. Set allSubsetsUsed := true.
        If (allOutputsUsed = true) and (allSubsetsUsed = true)
            Stop.
        Else
            Go to Step 2a.
```

3.2 Classification algorithm

In this section we describe the proposed algorithm for solving the decomposed problem.

Step 0: Initialization

Divide the set of input variables I and the set of output variables O into q subsets I_1, $I_2,\ldots,I_q$ and O_1, $O_2,\ldots,O_q$ as described in the decomposition algorithm. For each pair of subsets, form a new dimensionally decomposed DEA sub-problem A_i (i = 1, 2, …, q).

Step 1: Dimensional Decomposition

Solve each of the sub-problems A_i. Get the union of efficient units K^{E*} in each sub-problem $K_1^E \cup K_2^E \cup \ldots \cup K_q^E = K^{E*}$

Step 2: Reduction of Number of Potentially Efficient Units

In the set N - K^{E*}, identify inefficient units using all inputs and outputs. Use only the units DMU_i, $i \in K^{E*}$ as basic units in (2.4). Define a new index set K^{super} that includes the set K^{E*} and the units that were found either efficient or super-efficient.

Step 3: Recognition of Efficient Units

In the set K^{super}, identify inefficient units using all variables. The units that are efficient in this problem form the set of efficient units in the original problem.

4 COMPUTATIONAL RESULTS

We tested our decomposition procedure using simulated problems, which we received from Prof. Jose Dulá. He has also used these problems in his own tests. The parameters of the problems are the number of units n, the number of inputs/outputs

$m+p$, and the density of the efficient units d. The number of units we used was 5000, 10000, 15000, 20000, 25000, and 50000. The number of inputs/outputs was 20. We classified the models into three density categories in such a way that the average densities in each class were 6.4%, 10.0%, and 22.9%. The computing times are reported using five different block-quantities q. The problems are decomposed dimensionally into two (q = 2), three (q = 3), four (q = 4) and five (q = 5) sub-problems. Computing times without decomposition (q = 1) are also reported. The tests are run with a PC-computer with one 2.4 GHz processor. The DEA-model used is CCR.

The test results are reported in Tables 1-4 in each combination of the parameters. In Table 1, we have reported the computing times when the number of sub-problems is two (q = 2). In Table 2 the number of sub-problems is three (q = 3) and in Tables 3 and 4 the number of sub-problems is four (q = 4) and five (q = 5) respectively. Tables 1-4 represent the computing times in each steps together with the number of efficient (Step 1 and Step 3) or super-efficient (Step 2) units found in each step. Column *No Part* refers to the time (in seconds) needed to solve the problem without decomposition (q = 1). *Step 1 / Time* reports the time required to solve the sub-problems. Column *Step 1 / # of Alt* represents the number of efficient units found in sub-problems. Column *Step 2 / Time* states the time needed to calculate the super-efficiencies. *Step 2 / # of Alt* presents the number of super-efficient units found (together with the units found in Step1). Column *Step 3 / Time* states the time that is needed to calculate the exact set of efficient units. *Step 3 / # of Alt* presents the number of efficient units found in the problem. Column *Total / Time* reports the total time needed to solve the problem by dimensional decomposition approach. *Total / Ratio* compares the required total time by decomposition with the time without partition.

Table 1: Computing Times (s) and the Number of Alternatives when the Problem is Decomposed into Two Parts (*q*=2).

		No Part.	Step1		Step2		Step3		Total	
Density	# of Alt.	Time(s)	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	Ratio
	5 000	354	25	229	16	733	5	665	46	0.129
	10 000	1686	99	263	41	855	8	806	148	0.088
6.4 %	15 000	3559	239	274	72	956	11	829	322	0.091
	20 000	8589	431	295	127	997	14	888	571	0.067
	25 000	12342	705	341	172	1147	16	1011	893	0.072
	50 000	51753	3851	456	525	1642	39	1376	4415	0.085
	5 000	555	38	297	32	600	3	586	73	0.132
	10 000	2935	195	430	97	1089	12	1038	304	0.103
10.0 %	15 000	6864	778	549	235	1590	53	1498	1066	0.155
	20 000	11425	1781	641	363	2055	56	1930	2200	0.193
	25 000	19209	1978	707	489	2384	100	2384	2567	0.134
	50 000	106172	12776	1087	1559	4956	523	4607	14858	0.140
	5 000	675	85	509	65	1253	16	1209	166	0.245
	10 000	3264	383	712	197	2441	78	2346	659	0.202
22.9 %	15 000	8053	1475	952	438	3590	233	3447	2146	0.266
	20 000	15756	2921	1058	734	4770	541	4517	4196	0.266
	25 000	25089	4911	1280	1152	5921	900	5596	6963	0.278
	50 000	115585	21511	1802	3073	11670	4033	10783	28617	0.248

Table 2: Computing Times (s) and the Number of Alternatives when the Problem is Decomposed into Three Parts (*q*=3).

		No Part.	Step1		Step2		Step3		Total	
Density	# of Alt.	Time(s)	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	Ratio
	5 000	354	2	115	8	870	8	665	17	0.048
	10 000	1686	3	124	18	1271	18	806	39	0.023
6.4 %	15 000	3559	12	134	32	1388	21	829	64	0.018
	20 000	8589	13	134	42	1551	30	888	85	0.010
	25 000	12342	21	163	69	1599	33	1011	124	0.010
	50 000	51753	54	206	174	2697	123	1376	351	0.007
	5 000	555	2	151	16	721	5	586	23	0.041
	10 000	2935	5	167	32	1301	20	1038	57	0.020
10.0 %	15 000	6864	17	212	71	1799	41	1498	128	0.019
	20 000	11425	30	230	107	2395	77	1930	215	0.019
	25 000	19209	35	257	136	2952	140	2384	312	0.016
	50 000	106172	103	330	380	5643	747	4607	1231	0.012
	5 000	675	3	197	21	1328	18	1209	42	0.062
	10 000	3264	9	261	63	2612	94	2346	166	0.051
22.9 %	15 000	8053	28	293	100	3858	300	3447	428	0.053
	20 000	15756	51	328	151	5129	598	4517	799	0.051
	25 000	25089	60	323	184	6451	1008	5596	1251	0.050
	50 000	115585	210	445	562	12606	4889	10783	5661	0.049

Table 3: Computing Times (s) and the Number of Alternatives when the Problem is Decomposed into Four Parts (q=4).

		No Part.	Step1		Step2		Step3		Total	
Density	# of Alt.	Time(s)	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	Ratio
6.4 %	5 000	354	0	70	5	1127	13	665	18	0.051
	10 000	1686	1	87	13	1467	24	806	38	0.023
	15 000	3559	1	70	16	1847	42	829	58	0.016
	20 000	8589	4	99	30	2337	72	888	106	0.012
	25 000	12342	4	113	44	2212	71	1011	120	0.010
	50 000	51753	14	112	97	4496	510	1376	621	0.012
10.0 %	5 000	555	0	91	8	823	7	586	16	0.028
	10 000	2935	2	113	21	1400	24	1038	47	0.016
	15 000	6864	3	121	38	2136	68	1498	109	0.016
	20 000	11425	6	135	67	2815	149	1930	222	0.019
	25 000	19209	6	122	59	4061	436	2384	501	0.026
	50 000	106172	19	165	171	6817	1442	4607	1631	0.015
22.9 %	5 000	675	1	121	12	1420	22	1209	34	0.051
	10 000	3264	2	160	32	2736	109	2346	143	0.044
	15 000	8053	7	190	65	4119	424	3447	496	0.062
	20 000	15756	8	161	73	5584	864	4517	945	0.060
	25 000	25089	13	198	107	6884	1420	5596	1540	0.061
	50 000	115585	31	204	221	13678	7190	10783	7442	0.064

Table 4: Computing Times (s) and the Number of Alternatives when the Problem is Decomposed into Five Parts (q=5).

		No Part.	Step1		Step2		Step3		Total	
Density	# of Alt.	Time(s)	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	Ratio
6.4 %	5 000	354	0	45	3	1363	17	665	20	0.058
	10 000	1686	0	48	7	2476	75	806	82	0.049
	15 000	3559	0	50	10	2694	91	829	101	0.028
	20 000	8589	1	60	20	3770	258	888	279	0.032
	25 000	12342	1	53	25	3329	235	1011	261	0.021
	50 000	51753	2	56	52	6978	1497	1376	1551	0.030
10.0 %	5 000	555	0	54	4	997	12	586	16	0.029
	10 000	2935	0	59	10	1762	44	1038	54	0.018
	15 000	6864	1	68	17	2623	115	1498	133	0.019
	20 000	11425	1	57	20	3940	400	1930	421	0.037
	25 000	19209	1	67	29	4268	493	2384	522	0.027
	50 000	106172	2	74	68	8162	2360	4607	2430	0.023
22.9 %	5 000	675	0	58	5	1576	30	1209	35	0.052
	10 000	3264	0	57	10	3264	207	2346	217	0.067
	15 000	8053	2	67	24	4618	542	3447	567	0.070
	20 000	15756	1	60	25	6321	1165	4517	1191	0.076
	25 000	25089	1	75	36	7773	1991	5596	2028	0.081
	50 000	115585	2	81	86	15847	10900	10783	10988	0.095

Tables 1-4 show that the dimensional decomposition approach decreases computation times dramatically. Computation times were smallest with the block-quantities q = 3 and q = 4 in all problems. The difference in computation times between these two block-quantities is generally small. In the biggest problems with 50000 units the block-quantity q = 3 was somewhat faster. We can also see from the Tables 1-4 that the smaller the block-quantity the more time is spent in Step 1 and Step 2 and the more potentially efficient units are found in Step 1. On the other hand, the smaller the block-quantity the smaller the number of super-efficient units found in

Step 2 and the less time is spent in the Step 3 to calculate the exact set of efficient units.

The computing times are illustrated in Figures 1-3 in each efficiency category using each five block-quantities. In Figure 1, the average efficiency density is 6.4%. In Figure 2 the average efficiency density is 10.0% and in Figure 3 it is 22.9%.

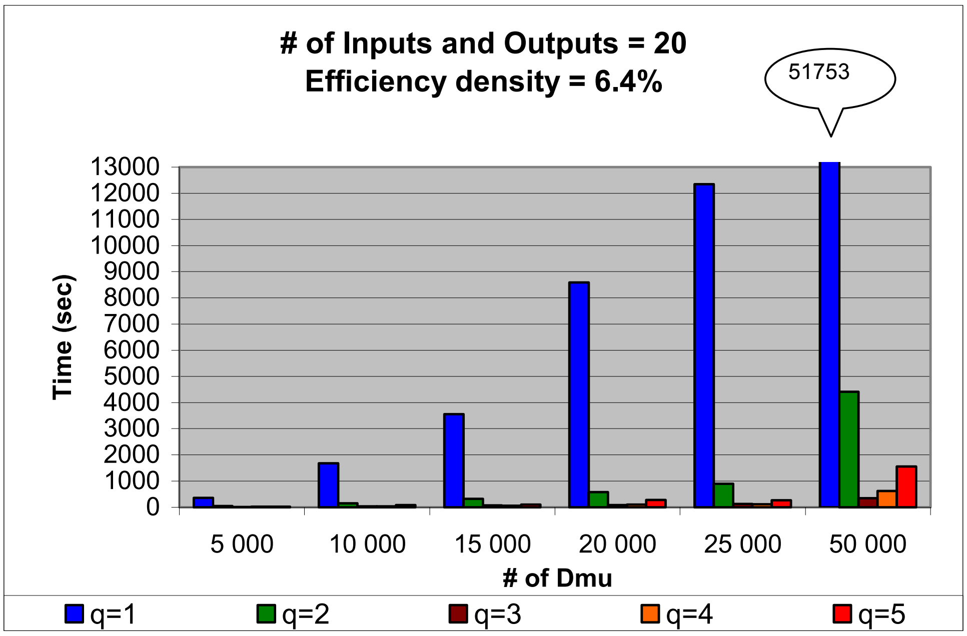

Figure 1: Computing Times (s) with Average Efficiency Density 6.4%

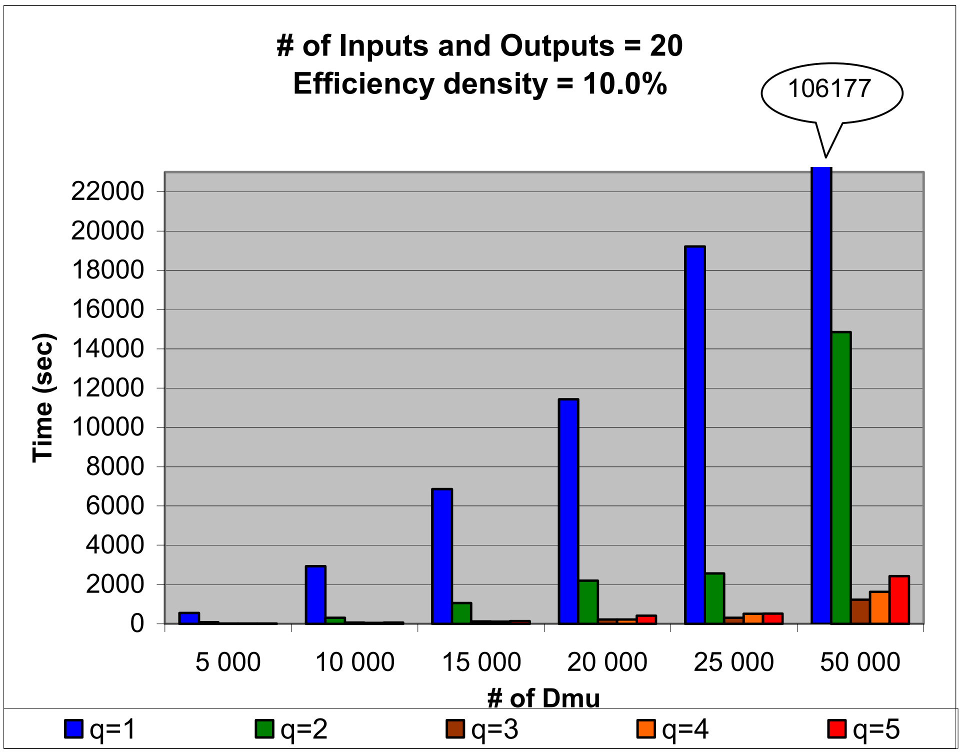

Figure 2: Computing Times (s) with Average Efficiency Density 10.0%

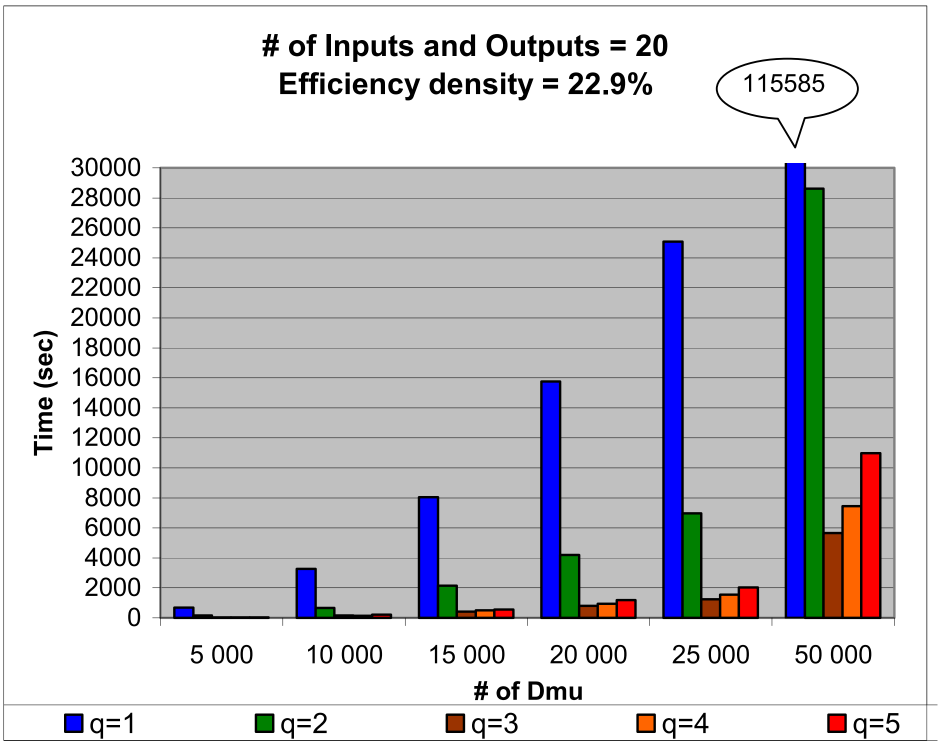

Figure 3: Computing Times (s) with Average Efficiency Density 22.9%

The complexity of the decomposition approach depends on *n*, *d*, and *q*. To estimate the effect on the computing times, we have used the following regression model to fit the data in Tables 1-4:

$t = \alpha n^{\beta} d^{\gamma} q^{\delta} \varepsilon$,

where ε is an error term and α, β, γ, and δ are parameters to be estimated. We took the logarithm of the model and used a linear regression to estimate the parameters. The results are represented in Table 5.

Table 5: The Estimates of the Regression Coefficients ($R^2 = 0.918$)

	Coefficients	*Standard Error*	*Lower 95%*	*Upper 95%*
log(*a*)	-5.229	0.397	-6.018	-4.440
b	2.366	0.098	2.171	2.560
c	1.058	0.108	0.842	1.274
d	-2.267	0.118	-2.501	-2.031

As can be seen from Table 5, the estimate for the parameter δ is distinctly negative. This parameter value gives us a rough estimate of the magnitude of the time savings using the dimensional decomposition approach. However, it should be emphasized that the relationship between the logarithm of the computation time and the logarithm of the block-quantity is not linear.

5 CONCLUSIONS

We have developed a dimensional decomposition procedure to reduce computation times when identifying efficient and inefficient units in the Data Envelopment Analysis (DEA). Usually the algorithms developed for large-scale problems are based on the idea to decompose the problem by units. First the efficient units are recognized in sub-problems and then those units are used in computing the efficiency scores for the inefficient units.

The algorithm developed in this paper divides the problem dimensionally by selecting a subset of variables into new sub-problems. The efficient units found in sub-problems are the weakly efficient units of the original problem. They provide an initial approximation for the set of all efficient units of the original problem. The set is supplemented during the solution procedure with the units that are diagnosed efficient or super-efficient. The efficient units of the original problem are recognized from this supplemented set. The computation times are reported using five different decomposition sizes. The dimensional decomposition approach decreases computation times drastically.

In the future, our purpose is to combine the dimensional decomposition approach with the other decomposition methods that have been proposed in the literature. We also need a method which enables us to select the optimal partition sizes and the sequence of the units to be checked. These issues are topics for further research.

REFERENCES

Afriat, S. (1972), “Efficiency Estimation of Production Functions”, International Economic Review, 568-598

Andersen, P., and Petersen, N. C. (1993), “A Procedure fo Ranking Efficient Units in Data Envelopment Analysis”, Management Science 39, 1261-1264.

Ali, A.I. (1993), “Streamlined Computation for Data Envelopment Analysis”,

European Journal of Operational Research 64, 61-67.

Ali, A.I (1994) “Computational aspects of Data Envelopment Analysis”, in Charnes, A., Cooper, W., Lewin, A.Y. and Seiford, L.M., (Eds.): Data Envelopment Analysis: Theory, Methodology, and Applications, Kluwer Academic Publishers, Norwell, 63 – 88.

Banker, R.D., Charnes, A. and Cooper, W.W. (1984), “Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment Analysis”, Management Science 30, 1078-1092.

Barr, R.S. and Durchholz, M.L. (1997). “Parallel and Hierarchical Decomposition Approaches for Solving Large-Scale Data Envelopment Analysis Models”, Annals of Operations Research 73, 339-372.

Chambers R. G., Y. Chung, and R. Färe (1998), “Profit, Directional Distance Functions, and Nerlovian Efficiency”, Journal of Optimization Theory and Applications 98, 351-364.

Charnes, A., Cooper, W.W. and Rhodes, E. (1978), “Measuring Efficiency of Decision Making Units”, European Journal of Operational Research 2, 429-444.

Charnes, A., Cooper, W.W. and Rhodes, E. (1979), “Short Communication: Measuring Efficiency of Decision Making Units”, European Journal of Operational Research 3, 339.

Charnes, A., Cooper, W., Lewin, A.Y. and Seiford, L.M. (1994**),** Data Envelopment Analysis: Theory, Methodology and Applications, Kluwer Academic Publishers, Norwell.

Dulá, J.H., and Helgason, R.V. (1996), “A New Procedure for Identifying the Frame of the Convex Hull of a Finite Collection of Points in Multidimensional Space”, European Journal of Operational Research 92, 352-367.

Dulá, J.H., Helgason, R.V., and Venugopal, N. (1997). “An Algorithm for Identifying the Frame of a Pointed Finite Conical Hull”, Journal of Computing 10, 323-330.

Dulá, J. H., and López, F. J. (2002), “Data Envelopment Analysis (DEA) in Massive Data Sets”, in Abello, J. , Pardalos, P., and Resende, M. (Eds.): Handbook of Massive Data Sets, Kluwer Academic Publisher, pp. 419-437.

Färe, R., S. Grosskopf, and J. Logan (1983), “The Relative Efficiency of Illinois Electric Utilities”, Resources und Energy 5, 349-367.

Halme, M., Joro, T., Korhonen, P., Salo, S. and Wallenius J. (1999), "A Value Efficiency Approach to Incorporating Preference Information in Data Envelopment Analysis", Management Science 45, pp. 103-115.

Korhonen, P. and Halme, M. (1996), "Using Lexicographic Parametric Programming for Searching a Nondominated Set in Multiple Objective Linear Programming", Journal of Multi-Criteria Decision Analysis 5, 291-300.

Korhonen, P. and Siitari, P. - A. (2007): "Using Lexicographic Parametric Programming for Identifying Efficient Units in DEA", Computers & Operations Research 34, pp. 2177-2190.