Pyry-Antti Siitari

Identifying Efficient Units in Large-Scale Dea Models

Using Efficient Frontier Approximation

HELSINGIN KAUPPAKORKEAKOULU
HELSINKI SCHOOL OF ECONOMICS

W-463

Pyry-Antti Siitari

Identifying Efficient Units in Large-Scale Dea Models Using Efficient Frontier Approximation

Department of Business Technology

February
2009

HELSINGIN KAUPPAKORKEAKOULU
HELSINKI SCHOOL OF ECONOMICS
WORKING PAPERS
W-463

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

ISSN 1235-5674
(Electronic working paper)
ISBN 978-952-488-313-9

Helsinki School of Economics -
HSE Print 2009

February 25, 2009

IDENTIFYING EFFICIENT UNITS IN LARGE-SCALE DEA MODELS USING EFFICIENT FRONTIER APPROXIMATION

Pyry-Antti Siitari

(February 2009)

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: pyry-antti.siitari@hse.fi

The research was supported by the Academy of Finland.

Acknowledgements: The author would like to thank Prof. Jose Dulá for providing the data files used in the performance tests. Special thanks are also due to Prof. Pekka Korhonen for useful discussions and guidance.

ABSTRACT

In this paper, we propose a computationally effective stepwise method to identify efficient units in large-scale Data Envelopment Analysis (DEA) models. The method is based on approximating the efficient frontier. We first identify a subset of efficient decision making units (DMUs) using lexicographic parametric programming. The subset of efficient units is then used to reduce the set of remaining units that are potentially efficient. Finally, the set of efficient units in the problem is recognized from the reduced set of potentially efficient units. The computational effectiveness of the proposed algorithm is demonstrated by solving a wide range of problems – including models with 100 000 units and twenty inputs and outputs. We also compare the algorithm to the methods represented in our previous papers and discuss their conceivable integration.

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

1 INTRODUCTION

Data Envelopment Analysis (DEA) is a non-parametric technique used in estimating the relative efficiency of different decision making units (DMUs). It is assumed that DMUs are performing the same task using similar multiple inputs to produce similar multiple outputs under similar conditions.

DEA models produce feasible input-output combinations based on observed inputs and outputs of the DMUs and their linear combinations. These feasible input-output combinations are called the production possibility set (PPS). The production possibility set determines how much a given DMU could increase its outputs or decrease its inputs, i.e. how efficient it is. Efficient frontier (surface) is the subset of the production possibility set where none of the inputs or outputs can be improved without worsening some other 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 efficient units.

To check the efficiency of a unit and to find the reference set and the efficiency score for inefficient units requires the solving an LP-model. The “standard” algorithm solves iteratively an LP-model for each unit separately. For each unit, the rhs - vector and one column (direction vector) in the coefficient matrix has to be updated. The optimal basis of the previous iteration is not valid for the next iteration as such. The approach is usable in small problems, but is computationally very ineffective in large scale problems.

When the number of units is large, let us say many ten thousands or even hundreds of thousands, computational aspects become important. Such problems appear when, for example, all high-schools or hospitals in Europe are evaluated, or when the efficiency analysis is made at an individual level. There exist also models which demand

efficiency calculation several times for each DMU. For example, a fuzzy DEA approach developed by Kao and Liu (2000) utilizes an α-cut concept in which each α-cut needs to solve a pair of ordinary DEA models. In these kinds of problems, the straightforward approach to formulate an LP-model for each unit with an unknown status becomes too time-consuming. Fortunately, the structure of the DEA-model makes it possible to develop special techniques for large-scale problems.

In this paper, we propose an efficient frontier approximation procedure to reduce the computation times of lexicographic parametric programming method by Korhonen and Siitari (2007). Since the approach in Korhonen and Siitari (2007) typically finds most of the efficient units at the very beginning of the computation, it is possible to get a good approximation by stopping the computation process early and using the efficient units found thus far as an approximated frontier. For example, in a problem with 10000 units and 10 inputs/outputs (solved by using a procedure described in Korhonen and Siitari (2007)) over 90% of all efficient units were found after running the procedure under 10% of the total computation time. Using a subset of efficient units (the units that form the approximated efficient frontier) as potential basic variables, it is possible to recognize most of the remaining inefficient units rapidly. During the solution procedure the subset of efficient units is supplemented with the units that are diagnosed super-efficient (Andersen and Petersen (1993)). Finally, the efficient units of the problem are recognized from this supplemented set.

The paper is given in eight sections. In the next section, we review some techniques that have been published to speed up DEA. In section 3, we illustrate the main idea of the article. In section 4, the necessary theoretical questions are discussed. In section 5, we represent the algorithm developed in this paper. Computational results are given and discussed in section 6. In section 7, we compare the solution technique developed in this paper to the dimensional decomposition technique of Korhonen and Siitari (2009). Section 8 concludes the paper with some remarks.

2 Overview of Existing Techniques

There are only few authors who have studied computational issues in DEA. Ali (1993, 1994), proposed an idea of "restricted basis entry" (RBE). The basis always consists of a set of existing efficient or unknown units. When a unit is diagnosed inefficient, the corresponding column is dropped from the set of potential basic vectors.

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 Dulá *et al.* (1998), (2001) and (2002). Dulá (2008) explores the impact of several LP enhancements and DEA specific accelerators. It reports the computational results of DEA problems with up to 100K DMUs.

Because the computing time as the function of the units increases more than linearly, Barr and Durchholz (1994) and (1997) proposed the partition of the set of the units. The efficient units are first identified in each partition of the 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.

Chen and Cho (2009) proposed an accelerating procedure that calculates the efficiency scores by identifying a set of neighborhood units for a DMU under consideration. The procedure performs well especially in problems where the proportion of efficient units is high.

Korhonen and Siitari (2007) used lexicographic parametric programming (Korhonen and Halme (1996)) to traverse from unit to unit along the efficient frontier. Lexicographic parametric programming was used to guarantee that the search will stay on the efficient frontier also in a case when a boundary is reached. On the way, it was also possible to early identify units efficient or inefficient. The units entering the basis were recognized efficient and the units dominated by an efficient facet were identified inefficient. The procedure is computationally efficient, when the number of inputs and outputs is small. This fact led Korhonen and Siitari (2009) to further develop the efficiency of the procedure by proposing the technique based on the idea of decomposing the problem dimensionally. The efficient units were identified in each sub-problem and further used as an initial approximation for the set of all units.

3 ILLUSTRATION OF THE MAIN IDEA OF THE ARTICLE

To illustrate the main idea of the procedure developed in this paper we use the following simple example. Assume we have six DMUs each using one identical input producing two outputs as depicted in Figure 1.

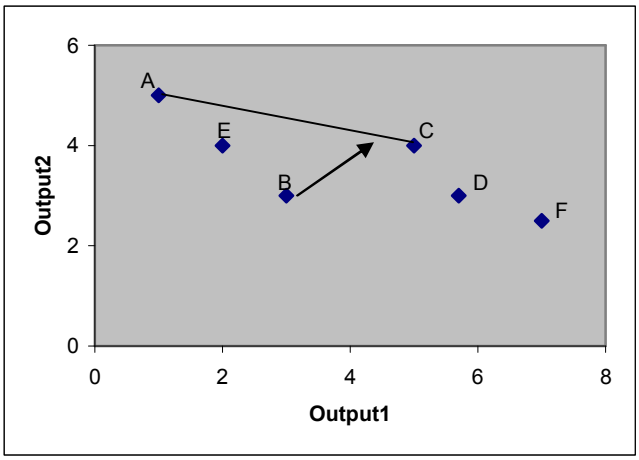

Figure 1. Constructing the Approximated Efficient Frontier (Step 1)

The search is started from A, which is found efficient. Next we check the efficiency of B. When we move from unit to unit we use lexicographic parametric programming (see, Korhonen and Siitari (2007)). During the search, unit C is recognized efficient. Unit B is found inefficient because it is projected[1] onto facet AC. Using the procedure

[1]In traditional DEA –models the projection is typically done radially. The term "radial" means that an efficient frontier is tried to reach either by proportionally increasing the values of the current outputs or decreasing the values of the current inputs. If we are only interested to find which units are efficient

described in Korhonen and Siitari (2007) we would next check the efficiency of units D, E and F. In the procedure used in this paper, we instead stop the solution process early and use the information available to construct an approximated efficient frontier. The approximated efficient frontier is constructed using the units that have been recognized efficient so far. The motive to use the approximation is to reduce the computational burden needed to evaluate the rest of the units. In this problem, we approximate the efficient frontier using efficient units A and C to classify the remaining unknown units E, D and F (Figure 2).

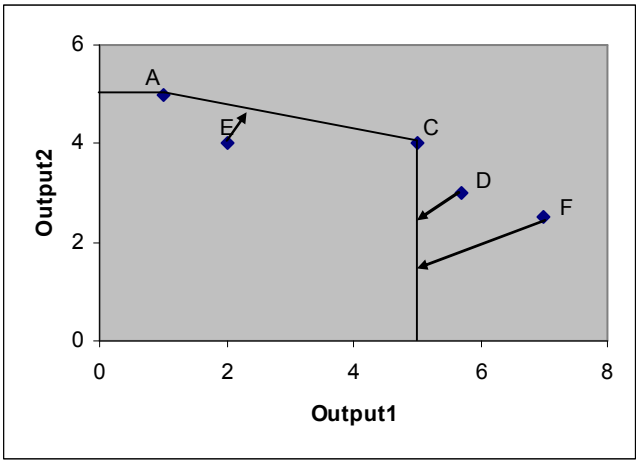

Figure 2. Using the Approximated Efficient Frontier to Identify Inefficient Units (Step 2)

Using the approximated efficient frontier, we find that unit E is inefficient and units D and F are super-efficient (see, Andersen and Petersen (1993)). Each pivot operation in this phase is less time consuming, because the coefficient matrix consists of only two units compared to six units at the previous phase (Figure 1). This way we can identify inefficient units with little computational effort.

We need one more phase to recognize the final status of the remaining two potentially efficient units D and F (Figure 3). In the final phase, only efficient (A, C) and potentially efficient (D, F) units must be included in the coefficient matrix. During the final step as well as in the first step any unit that is recognized inefficient can be immediately dropped from the coefficient matrix (see, Ali (1993) and (1994)).

(or inefficient) we can use any non-zero projection vector (see, discussion in Korhonen and Siitari (2007)).

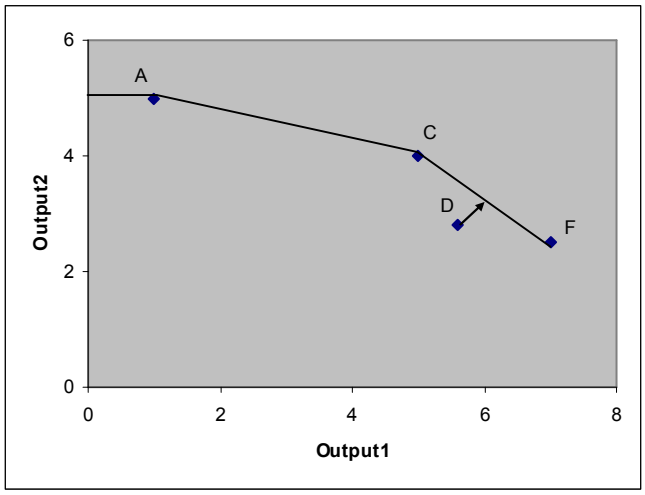

Figure 3. Recognizing all Efficient Units (Step 3)

In the last step, unit F is found efficient. Unit D is found inefficient because it is projected onto facet CF. The procedure has identified units A, C and F as efficient and units B, D and E as inefficient.

4 THEORETICAL CONSIDERATIONS

4.1 Definitions

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 $\boldsymbol{y} \in \Re_+^p$. Now we can define the production possibility set (PPS):

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

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 production possibility set P.

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

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

When $(\boldsymbol{y}^*, \boldsymbol{x}^*) \in$ P is not efficient, it is *inefficient.*

4.2 Basic Data Envelopment Models

Assume we have n DMUs each consuming m inputs ($m \neq 0$), and producing p outputs ($p \neq 0$). Let D be the index set of n DMUs (D = {1, 2, …, n}). Let **X** be an ($m \times n$) - matrix and **Y** be a ($p \times n$) - matrix consisting of non-negative elements, containing observed inputs and outputs of DMUs, respectively.

Consider the following lexicographic formulation[2] of general DEA in the so-called envelopment form (Korhonen and Siitari (2007)):

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

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

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

The first three constraints for λ specifies one of the BCC - models, and the last one the CCR-model. In the combined model, $\boldsymbol{w}^y = \boldsymbol{y}_j$ and $\boldsymbol{w}^x = \boldsymbol{x}_j$. In the input-oriented model $\boldsymbol{w}^y = \boldsymbol{0}$ and $\boldsymbol{w}^x = \boldsymbol{x}_j$, and in the output-oriented model $\boldsymbol{w}^x = \boldsymbol{0}$ and $\boldsymbol{w}^y = \boldsymbol{y}_j$. Input and output sets **X** and **Y** define the production possibility set P = $\{(\boldsymbol{y}, \boldsymbol{x}) \mid \boldsymbol{x} \geq \mathbf{X}\boldsymbol{\lambda}, \boldsymbol{y} \leq \mathbf{Y}\boldsymbol{\lambda}, \boldsymbol{\lambda} \in \Lambda\}$[3].

The value of σ_j , called an inefficiency score for unit j, at the optimum is denoted by σ_j^*. Notation "*lex max"* refers to a lexicographic maximization problem. It means that we first solve (1) using σ_j as an objective function. In case, the optimal solution σ_j^* is not unique, we add the constraint $\sigma_j = \sigma_j^*$ into the model (1) and solve it by using $\boldsymbol{1}^T\boldsymbol{s}_j^+ + \boldsymbol{1}^T\boldsymbol{s}_j^-$ as the objective function. Let´s for simplicity define the directional vector[4] $\boldsymbol{w} = (\boldsymbol{w}^x, \boldsymbol{w}^y)$ and $\boldsymbol{s}_j = (\boldsymbol{1}^T\boldsymbol{s}_j^+, \boldsymbol{1}^T\boldsymbol{s}_j^-)$.

---

[2]Lexicographic formulation is used to guarantee that weakly efficient units are not chosen into the basis (see, discussion in Korhonen and Siitari (2007)).

[3]For a textbook introduction into DEA see, for example, Charnes et al. (1994).

[4]For a discussion on directional distance functions see, Chambers et al. (1998).

Theorem 1. *The following results hold for model (1):*

1. $(\boldsymbol{x_j}, \boldsymbol{y_j})$ is efficient iff the value of σ_j at the optimum is $\sigma_j^* = 0$ and if the sum of slacks $\boldsymbol{1}^T\boldsymbol{s_j}$ at the optimum is $\boldsymbol{1}^T\boldsymbol{s_j}^* = 0$. This result holds for all $\boldsymbol{w} \geq \boldsymbol{0}$, $\boldsymbol{w} \neq \boldsymbol{0}$.
2. $(\boldsymbol{x_j}, \boldsymbol{y_j})$ is inefficient iff the value of σ_j at the optimum is $\sigma_j^* > 0$ or $\sigma_j^* = 0$ and $\boldsymbol{1}^T\boldsymbol{s_j}^* > 0$. This result holds for all $\boldsymbol{w} \geq \boldsymbol{0}$, $\boldsymbol{w} \neq \boldsymbol{0}$.

Proof:

see, Korhonen and Luptacik (2004).

If model (1) is solved for all units $j \in D$ we can define $E = \{j \in D \mid \sigma_j^* = 0, \boldsymbol{1}^T\boldsymbol{s_j}^* = 0\}$ to be the set of the indices of the efficient units in D. Accordingly, we can define I = D - E to be the set of inefficient units in D.

Let B_j^* be the set of basic variables associated with an optimal basis for model (1) of DMU j.

Theorem 2. *Early identification of efficient units:*

For all $j \in D$, $B_j^* \subseteq E$.

Proof:

see, for example Ali (1993).

Theorem 2 states that all the units in the optimal basis for some unit are efficient. Actually, if the problem (1) is maintained lexicographically optimal, a unit that enters the basis at any pivot operation is efficient (see, Korhonen and Siitari (2007)).

Any unit that is found inefficient can be dropped from the set of potential basic variables. This way, we can drop the columns of the inefficient units from **X** and **Y** matrices during the solution process ("restricted basis entry", RBE, see Ali (1993, 1994)) and diminish the complexity of pivot operations that follows.

4.3 Problem Approximation

Let $D_h \subseteq D$ ($D_h \neq \emptyset$) be the index set of the units used to approximate the production possibility set of the model (1). Let a ($a = |D_h|$) be the number of units used in the approximation. Let $\mathbf{X_h}$ be an ($m \times a$) - matrix and $\mathbf{Y_h}$ be a ($p \times a$) - matrix consisting of the inputs and outputs of the DMUs $h \in D_h$.

Let $D_k \subseteq D$ ($D_k \neq \emptyset$) be the index set of units whose efficiency status we are interested to check with the approximated problem. Sets D_h and D_k need not be disjoint[5]. Neither must their union equal to set D[6].

The model (1) can be approximated by:

[5]However, in practise when the approximated problem (2) is used in the algorithm represented in section 5 D_h and D_k are always disjoint (i.e. $D_h \cap D_k = \emptyset$).

[6]In the algorithm of section 5 the union of D_h and D_k is usually not equal to D (i.e. $(D_h \cup D_k) \neq D$). However, in some rare cases when the frontier construction phase did not find any inefficient units it is possible that their union equals to D (i.e. $(D_h \cup D_k) = D$).

$$
\begin{aligned}
&\textit{lex max } \{\sigma_k ,\ \mathbf{1}^T \mathbf{s}_k^+ + \mathbf{1}^T \mathbf{s}_k^-) \\
&\text{s.t.} \qquad (2)\\
&\mathbf{X_h}\boldsymbol{\lambda} + \sigma_k \boldsymbol{w}^x + \mathbf{s}_k^- = \boldsymbol{x}_k \\
&\mathbf{Y_h}\boldsymbol{\lambda} - \sigma_k \boldsymbol{w}^y - \mathbf{s}_k^+ = \boldsymbol{y}_k \\
&\qquad \boldsymbol{\lambda} \in \boldsymbol{\Lambda} \\
&\qquad \boldsymbol{\lambda},\ \mathbf{s}_\mathbf{k}^- ,\ \mathbf{s}_\mathbf{k}^+ \geq \mathbf{0}
\end{aligned}
$$

where $\boldsymbol{x}_k$ is the input-vector and $\boldsymbol{y}_k$ is the output-vector of a DMU $k \in D_k$ under consideration. Input and output sets $\mathbf{X_h}$ and $\mathbf{Y_h}$ define an approximated production possibility set $P_h = \{(\boldsymbol{y}, \boldsymbol{x}) \mid \boldsymbol{x} \geq \mathbf{X_h}\boldsymbol{\lambda}, \boldsymbol{y} \leq \mathbf{Y_h}\boldsymbol{\lambda}, \boldsymbol{\lambda} \in \Lambda\}$.

In the model (2) units $h \in D_h$ form the set of potential basic variables. A DMU $k \in D_k$ does not usually belong to the set D_h. If it does not belong to the set D_h it is not included in $\mathbf{X_h}$ and $\mathbf{Y_h}$. Thus, we can define the following three efficiency concepts for the model (2):

Theorem 3. *The following results hold for the model (2):*

1. $(\boldsymbol{x}_k , \boldsymbol{y}_k) \in P_h$ is efficient in the set P_h iff the value of σ_k at the optimum is $\sigma_k^* = 0$ and the sum of slacks $\mathbf{1}^T\mathbf{s}_k$ at the optimum is $\mathbf{1}^T\mathbf{s}_k^* = 0$. This result holds for all $\boldsymbol{w} \geq \mathbf{0}$, $\boldsymbol{w} \neq \mathbf{0}$.
2. $(\boldsymbol{x}_k , \boldsymbol{y}_k) \in P_h$ is inefficient iff the value of σ_k at the optimum is $\sigma_k^* > 0$ or $\sigma_k^* = 0$ and $\mathbf{1}^T\mathbf{s}_k^* > 0$. This result holds for all $\boldsymbol{w} \geq \mathbf{0}$, $\boldsymbol{w} \neq \mathbf{0}$.
3. $(\boldsymbol{x}_k , \boldsymbol{y}_k) \in P - P_h$ is super-efficient in set the P_h iff the value of σ_k at the optimum is $\sigma_k^* < 0$. This result holds for all $\boldsymbol{w} > \mathbf{0}$.

Proof:

Points 1. & 2: see, Korhonen and Luptacik (2004).

Point 3: Assume $(\boldsymbol{x}_k , \boldsymbol{y}_k)$ is super-efficient. Assume that, at the optimum of the problem, σ_k^* would be greater than or equal to zero ($\sigma_k^* \geq 0$) using a directional vector $\boldsymbol{w} > \mathbf{0}$. That would imply that there exists another point $(\boldsymbol{x}, \boldsymbol{y}) \in P_h$ such that $\boldsymbol{x} = \mathbf{X_h}\boldsymbol{\lambda}^* \leq \boldsymbol{x}_k$ and $\boldsymbol{y} = \mathbf{Y_h}\boldsymbol{\lambda}^* \geq \boldsymbol{y}_k$ contracting the initial assumption that $(\boldsymbol{x}_k , \boldsymbol{y}_k)$ is super-efficient.

If model (2) is solved for all units $k \in D_k$ they will be divided into three mutually exclusive and exhaustive sets $E_k = \{k \in D_k \mid \sigma_k^* = 0, \mathbf{1}^T\mathbf{s}_k^* = 0\}$, $S_k = \{k \in D_k \mid \sigma_k^* < 0\}$, and $I_k = D_k - E_k - S_k$. The sets E_k, S_k and I_k refer respectively to units in D_k that are efficient, super-efficient and inefficient in the model (2).

Theorem 4. *If a unit is inefficient in the approximated model (2) it is also inefficient in the original model (1):*

i.e. for all $k \in D_k$ ($D_k \subseteq D$), if $k \in I_k$ then also $k \in I$

Proof:

Unit k corresponds to a point $(\boldsymbol{x}_k , \boldsymbol{y}_k)$. Because it is inefficient in the model (2) there exists another point $(\boldsymbol{x}^*, \boldsymbol{y}^*) \neq (\boldsymbol{x}_k , \boldsymbol{y}_k)$ such that $\boldsymbol{x}^* = \mathbf{X_h}\boldsymbol{\lambda}^* \leq \boldsymbol{x}_k$ and $\boldsymbol{y}^* = \mathbf{Y_h}\boldsymbol{\lambda}^* \geq \boldsymbol{y}_k$. Because $D_h \subseteq D$ there exists $\boldsymbol{\lambda_2}^*$ in the model (1) such that $\mathbf{X}\boldsymbol{\lambda_2}^* = \mathbf{X_h}\boldsymbol{\lambda}^* = \boldsymbol{x}^* \leq \boldsymbol{x}_k$ and $\mathbf{Y}\boldsymbol{\lambda_2}^* = \mathbf{Y_h}\boldsymbol{\lambda}^* = \boldsymbol{y}^* \geq \boldsymbol{y}_k$ implying the unit k that corresponds to

the point ($\boldsymbol{x}_k$, $\boldsymbol{y}_k$) is inefficient also in the original problem (1) ($\boldsymbol{\lambda}_2$* could be constructed so that its components that correspond to units $k \in D_k$ are the same than in $\boldsymbol{\lambda}$* while all the other components in $\boldsymbol{\lambda}_2$* are zero).

Theorem 4 states that if the unit k under evaluation in the approximated model (2) is inefficient, it is also inefficient in the original model (1). However, if the unit is efficient or super-efficient in the approximated model (2), it could still be inefficient in the original model (1). Since the matrices $\mathbf{X_h}$ and $\mathbf{Y_h}$ in the approximated problem (2) are smaller, the pivot operations[7] are also less time consuming compared to the original problem (1). By solving the approximated problem first we can reduce the computing times when evaluating the efficiency of all units $j \in D$, because every unit that is diagnozed inefficient in the approximated problem (2) can be dropped from the problem (1) when identifying the set of efficient units.

5 DEVELOPMENT OF THE PROCEDURE

The following algorithm will be used in section 6 to solve a set of simulated problems. The algorithm consists of three main steps. First we search for the units that will be used to construct the approximated efficient frontier (Step 1 in the procedure below). In principle, any set of units could be used to approximate the efficient frontier. However, using the formulation (1) we can find units that are known to be efficient. Thus, it is certain that a minimal number of units are used to form a given approximated production possibility set P_h (i.e. there is not inefficient units in the index set D_h). This is important, because inefficient units would make every pivot operation computationally more demanding in the next phase (Step 2 in the procedure below). According to formulation (2) and Theorem 4, the approximated efficient frontier can be used to identify units that are inefficient in the original problem (Step 2). Finally, we need to check the efficiency status of the units that were found either efficient or super-efficient in Step 2 to identify the set of efficient units in the original problem (Step 3). The more inefficient units there were found in Step 2, the less demanding every pivot operation is in the final phase and the less there are units whose efficiency status is to be checked.

The only parameter in the algorithm whose value must be determined beforehand is a fraction-parameter b ($0 < b \leq 1$). It determines the proportion of units to be evaluated in the approximation phase (Step 1). Parameter b determines how accurate the approximation of the efficient frontier is. The larger the selected value for b, the closer the approximation of the efficient frontier is to the original frontier. For example, if we select $b = 1$ the procedure is guaranteed to identify the efficiency status of all the units in Step 1. This is obvious, because in that case the approximated frontier equals the original frontier. On the other hand, small values for b will cause

[7] The complexity of a pivot operation is dependant on the number of rows and columns in the DEA model. In model (1) the complexity is approximately $c*(m+p)*n$, in model (2) the complexity is $c*(m+p)*a$, where c is some constant, m is the number of inputs, p is the number of outputs, n is the number of DMUs in model (1) and a is the number of DMUs that are used to construct the frontier in model (2). If the "restricted basis entry" (RBE, see Ali(1993) and (1994)) is used, then in model (1) n is only at the beginning of the procedure equal to the number of DMUs. After finding some inefficient units it is equal to the number of unknown and efficient units.

the procedure to spend more time in Step 3 and less time in Steps 1 and 2. The optimal value for b will be determined empirically in section 6.

Step 0: Initialization

Select a value for the fraction-parameter b. Select a directional vector[8] $\boldsymbol{w} > 0$. The same directional vector is used for all the units. Define sets U = D, E = ∅ and I = ∅ to describe units that have been identified unknown, efficient and inefficient respectively in the original problem (1).

Step 1: Constructing the Approximated Efficient Frontier

Step 1.0: Formulate the problem as model (1). Use units U ∪ E as potential basic variables in (1). Use early identification of efficient units and RBE (Theorem 2).

Step 1.1: Evaluate the efficiencies of b*|U| (rounded up to the next integer) units in the set U. Append new efficient and inefficient units that are found into the sets E and I respectively. Redefine U := U – E - I.

If (U = ∅)

E is the set of efficient units in D. The procedure can be stopped.

Else

Go to Step 2.

Step 2: Using the Approximated Efficient Frontier to Identify Inefficient Units

Step 2.0: Formulate the problem as model (2). Use only units E, that were found efficient in the Step 1, as potential basic variables (i.e. D_h in section 4.3 is equal to E (D_h = E)).

Step 2.1: Evaluate the super-efficiencies of all the units U whose efficiency status are still after Step 1 unknown (i.e. D_k in section 4.3 is equal to U (D_k = U)). Append all inefficient units $k \in D_k$ found in the approximated problem (2) into set I_k. Using Theorem 4 we can redefine I := I + I_k and U := U – I_k.

Step 3: Recognizing all Efficient Units

Step 3.0: Formulate the problem as model (1). Use units (U ∪ E) as potential basic variables in (1). Use early identification of efficient units and RBE (Theorem 2).

Step 3.1: Evaluate the efficiencies of remaining unknown units U. Append new efficient and inefficient units that are found in U into sets E and I respectively. E is the set of efficient units in D.

In Steps 1 and 3 the procedure identifies new efficient and inefficient units using the formulation (1). When the problem is formulated as in (1) the units under evaluation form a subset of the set of potential basic variables. Thus, the unit under evaluation in these steps can not be super-efficient. Steps 1 and 3 use units U and E as potential

[8] To avoid computational difficulties, we select a vector such that each element is of the same magnitude as the corresponding inputs and outputs.

basic variables. Step 1 evaluates the efficiencies of some fraction b of all the units. Step 3 evaluates the efficiencies for all the remaining units in U. Steps 1 and 3 always use early identification of efficient units and RBE. If the number of input and output variables is low also early identification of inefficient units should be used[9] (see, details in Korhonen and Siitari (2007)). By using RBE in these steps, we can remove every inefficient unit that is found from the set of potential basic variables. This way we can diminish the complexity of every pivot operation that follows.

Step 2 reduces the number of potentially efficient units in the original problem. When the problem (2) is formulated as in Step 2 the unit $k \in D_k$ under evaluation does not belong to the set of potential basic variables D_h. Thus, units under evaluation in this step can be super-efficient. The units that are identified efficient or super-efficient in Step 2 can be inefficient in the original problem, so in Step 2 we are mostly interested in the units that are found inefficient. Inefficient units found in the approximated problem of Step 2 are also inefficient in the original problem. In Step 2 we do not use early identification of efficient units or RBE, because all the units in the set of potential basic variables are beforehand known to be efficient.

The sequence of the units to be checked has also an effect on the performance of the procedure. In this paper, we use a random sequence in all the steps. However, the efficiency of the procedure might be better if a more sophisticated selection of the sequence was used. This might be especially important for Step 3, because the procedure typically spends most of time in it (see, section 6). Also, in Step 2 we could collect some information about the differences in the optimal basis solutions of efficient and super-efficient units. This information could be used in Step 3 to select a sequence that reduces the total number of pivot operations. [10]

6 ANALYSIS AND NUMERICAL RESULTS

6.1 Computational Results

We tested the approximation procedure using simulated problems, which we received from Prof. Jose Dulá. These problems are also the same that we have used in our previous papers (Korhonen and Siitari (2007) and (2009)). The parameters of the problems are the number of units n, the number of inputs and outputs $v = m + p$, and the density of the problem d. The number of units we used was 5000, 10000, 15000, 20000, 25000, and 50000. The number of inputs and outputs (v) was 5, 10, 15 and 20. We classified the models into three density categories (low, average and high). These categories represent the average densities in the models. The computing times were calculated using twenty different values for parameter b from 0.0025 to 0.05 with

[9] If we use the early identification of inefficient units we check, during a pivot operation, whether the values in the column of the simplex tableau are all non-negative. This checking increases the complexity of pivot operations. If the dimension of the problem is small, the increased complexity of pivot operations is more than compensated by the diminished number of units to be checked. In Korhonen and Siitari (2007), it was reported that the early identification of inefficient units is generally useful if the number of variables ($m + p$) is not greater than ten.

[10] For example, we could select the sequence in Step 3 so that the units whose optimal basis is similar in Step 2 are near each other in the sequence.

increment 0.0025.[11] Computing times with $b = 1$ represent a case where the problem is solved in one phase (Korhonen and Siitari (2007)). In this case, the approximation procedure is not used and it serves as a reference to illustrate the time savings achieved by efficient frontier approximation approach. The procedure was implemented in Java and it was run using Java runtime environment 1.6.0_07. The tests were run with a PC-computer with one 2.4 GHz processor and 1 GB RAM. The DEA-model used was CCR.

The test results are reported in Tables 1-4. In Table 1, we have reported the computing times when the number of inputs and outputs is twenty. In Table 2 the number of inputs and outputs is fifteen, in Table 3 ten and in Table 4 five. 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 *b = 1* refers to the time (in seconds) needed to solve the problem in one phase. Column *Best b* represents the value for parameter *b* that produced the smallest total computing time in each problem.[12] *Step 1 / Time* reports the time required to construct the approximated efficient frontier in Step 1. Column *Step 1 / # of Alt* represents the number of efficient units that were used to approximate the frontier. Column *Step 2 / Time* states the time needed to calculate the super-efficiencies when reducing the set of potentially efficient units (Step 2 of the procedure in section 5). *Step 2 / # of Alt* presents the number of super-efficient units found in Step 2. Column *Step 3 / Time* states the time that is needed to calculate the set of efficient units in Step 3. *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 the efficient frontier approximation approach. *Total / Ratio* compares the required total time using the approximation procedure with the time using the direct calculation of efficient units (i.e. the case when $b = 1$).

[11] This interval was selected empirically. All the problems had a single best value with computation times increasing with a smaller or larger values around the best value (see, Figure 4. which illustrates the computation times as a function of *b* in a problem with v=20, n=20000 and d=22.9%).

[12] We use the term "best *b*" to refer to the value that produced the smallest computation time of all the parameter values tested. It is not a mathematically proven optimum, but should be a good enough value for practical purposes.

Table 1. Computing Times (s) and the Number of Alternatives When the Number of Inputs and Outputs is Twenty ($v = 20$).

Density	# of Alt.	$b = 1$ Time (s)	*Best b*	Step1 Time (s)	Step1 # of Alt.	Step2 Time (s)	Step2 # of Alt.	Step3 Time (s)	Step3 # of Alt.	Total Time (s)	Total Ratio
6.4 %	5,000	311	0.0125	1	64	4	925	9	665	14	0.045
	10,000	1450	0.0125	4	113	15	1039	17	806	36	0.024
	15,000	2931	0.0100	7	131	28	993	15	829	50	0.017
	20,000	6334	0.0075	8	134	41	1672	35	888	84	0.013
	25,000	10838	0.0075	15	166	71	1306	35	1011	121	0.011
	50,000	46695	0.0051	31	231	201	1970	89	1376	321	0.007
10.0 %	5,000	490	0.0175	1	87	7	819	9	586	17	0.034
	10,000	2357	0.0175	4	176	32	1266	33	1038	69	0.029
	15,000	6010	0.0125	5	184	52	1940	80	1498	137	0.023
	20,000	10209	0.0100	10	210	76	2678	154	1930	240	0.024
	25,000	17525	0.0125	15	305	145	2936	205	2384	365	0.021
	50,000	95605	0.0075	28	368	383	5793	845	4607	1256	0.013
22.9 %	5,000	615	0.0275	1	153	12	1315	33	1209	46	0.075
	10,000	2665	0.0250	5	257	46	2437	111	2346	162	0.061
	15,000	7186	0.0225	11	333	101	3829	322	3447	434	0.060
	20,000	14106	0.0225	19	454	195	4808	532	4517	745	0.053
	25,000	22241	0.0175	17	443	233	6053	855	5596	1105	0.050
	50,000	103604	0.0125	47	641	800	11932	3650	10783	4497	0.043

In Table 1 we have classified the models into 6.4%, 10.0% and 22.9% average density categories. We can see from Table 1 that the best value for parameter b tends to increase when the efficiency density increases. On the other hand, b is smaller when the number of DMUs increases. Computation time is decreased drastically in all the problems. The largest proportional time savings are achieved in low density problems with a high number of DMUs. For example, in a problem with 50 000 units in a density category 6.4% the computation time needed is 321 seconds which is less than one percent of the reference case.

From Table 1 we can also see that in Step 3 we actually solve a problem with high efficiency density, because in Step 2 the procedure is able to recognize most of the inefficient units of the original problem. For example, in a problem with 50 000 units in a density category 22.9% the number of potentially efficient units in Step 2 is 11932 (including 641 units that are known to be efficient from Step 1) and the total number of efficient units is 10783. In that problem, Step 3 could be interpreted as a new high efficiency density problem with over 90% of units being efficient. Thus, we could reduce computing times even further by using in Step 3 an approach that performs especially well in high efficiency density problems (for example, a procedure that is similar to the one described in Chen and Cho (2009) might be suitable).

Table 2. Computing Times (s) and the Number of Alternatives When the Number of Inputs and Outputs is Fifteen ($v = 15$).

Density	# of Alt.	$b = 1$ Time (s)	Best b	Step1 Time (s)	Step1 # of Alt.	Step2 Time (s)	Step2 # of Alt.	Step3 Time (s)	Step3 # of Alt.	Total Time (s)	Ratio
3.7 %	5,000	176	0.0125	0	56	3	561	2	394	5	0.029
	10,000	692	0.0150	2	95	8	636	3	489	13	0.019
	15,000	1882	0.0050	2	71	10	1072	9	474	21	0.011
	20,000	3925	0.0050	3	92	19	1132	12	526	34	0.009
	25,000	5145	0.0050	4	104	23	1464	19	535	45	0.009
	50,000	26228	0.0050	18	201	104	1429	26	866	148	0.006
9.1 %	5,000	303	0.0175	1	87	5	715	4	520	9	0.031
	10,000	1334	0.0125	2	133	13	1345	19	971	34	0.025
	15,000	3684	0.0125	4	184	32	1969	47	1427	83	0.023
	20,000	6251	0.0100	6	211	51	2515	77	1858	134	0.021
	25,000	8886	0.0075	5	206	50	2715	84	2054	139	0.016
	50,000	42483	0.0075	25	373	238	5270	387	3885	650	0.015
21.7 %	5,000	370	0.0275	1	171	9	1282	16	1169	25	0.068
	10,000	1768	0.0275	3	305	37	2266	53	2271	93	0.053
	15,000	4195	0.0175	4	265	48	3643	134	3357	186	0.044
	20,000	7588	0.0150	6	303	79	5199	323	4385	408	0.054
	25,000	11976	0.0150	11	372	116	5562	387	4729	514	0.043
	50,000	57602	0.0150	40	748	588	11658	1814	10473	2442	0.042

Table 3. Computing Times (s) and the Number of Alternatives When the Number of Inputs and Outputs is Ten ($v = 10$).

Density	# of Alt.	$b = 1$ Time (s)	Best b	Step1 Time (s)	Step1 # of Alt.	Step2 Time (s)	Step2 # of Alt.	Step3 Time (s)	Step3 # of Alt.	Total Time (s)	Ratio
1.7 %	5,000	35	0.0100	0	44	1	236	0	168	1	0.037
	10,000	156	0.0100	1	46	2	481	1	166	3	0.021
	15,000	427	0.0050	1	55	4	931	3	236	7	0.017
	20,000	751	0.0075	1	60	6	743	2	242	9	0.012
	25,000	1570	0.0075	2	78	10	1291	7	317	18	0.012
	50,000	6022	0.0050	4	111	30	1124	5	529	39	0.006
7.9 %	5,000	99	0.0150	0	74	2	616	2	430	4	0.039
	10,000	531	0.0100	1	89	4	1378	9	793	14	0.026
	15,000	1304	0.0100	1	132	12	1883	20	1198	33	0.025
	20,000	2839	0.0075	2	152	19	2232	29	1562	50	0.018
	25,000	4192	0.0075	3	187	25	2390	29	1892	56	0.013
	50,000	18816	0.0075	12	354	122	5234	205	3663	339	0.018
18.4 %	5,000	151	0.0225	0	104	3	1344	7	1010	11	0.070
	10,000	771	0.0225	2	213	14	2296	30	1926	45	0.058
	15,000	1812	0.0175	3	246	25	3444	71	2766	98	0.054
	20,000	3850	0.0125	3	249	31	4425	117	3555	151	0.039
	25,000	6147	0.0125	5	314	50	5863	223	4448	278	0.045
	50,000	29393	0.0125	20	615	238	10955	901	8404	1158	0.039

Table 4. Computing Times (s) and the Number of Alternatives When the Number of Inputs and Outputs is Five ($v = 5$).

Density	# of Alt.	$b = 1$ Time (s)	Best b	Step1 Time (s)	Step1 # of Alt.	Step2 Time (s)	Step2 # of Alt.	Step3 Time (s)	Step3 # of Alt.	Total Time (s)	Ratio
0.6 %	5,000	0.3	0.0025	0.0	10	0.1	165	0.0	35	0.1	0.564
	10,000	0.5	0.0025	0.1	9	0.1	3040	0.1	66	0.2	0.470
	15,000	1.4	0.0025	0.1	23	0.3	274	0.0	85	0.5	0.346
	20,000	1.8	0.0025	0.1	18	0.4	1086	0.0	108	0.5	0.278
	25,000	3.7	0.0025	0.1	13	0.3	2677	0.3	129	0.7	0.182
	50,000	8.6	0.0025	0.5	27	0.9	3517	0.8	207	2.1	0.247
3.3 %	5,000	1.1	0.0025	0.0	22	0.1	464	0.1	216	0.3	0.285
	10,000	4.5	0.0025	0.1	34	0.3	2829	0.8	363	1.2	0.267
	15,000	7.8	0.0025	0.2	33	0.8	4684	2.1	462	2.6	0.333
	20,000	16.3	0.0025	0.3	53	0.8	2992	2.2	633	3.3	0.202
	25,000	23.6	0.0025	0.4	54	0.9	4662	5.1	739	6.4	0.271
	50,000	80.8	0.0025	1.0	78	3.6	9878	22.7	1185	27.4	0.339
6.5 %	5,000	2.8	0.0125	0.1	36	0.3	1067	0.6	437	0.9	0.335
	10,000	11.5	0.0050	0.2	53	0.6	1929	2.1	702	2.9	0.255
	15,000	27.1	0.0175	0.3	52	1.0	3679	5.6	943	6.6	0.244
	20,000	44.3	0.0300	0.5	87	2.0	3835	8.0	1227	10.5	0.237
	25,000	68.8	0.0300	0.6	91	2.8	6511	21.7	1452	25.1	0.365
	50,000	246.2	0.0025	1.5	108	5.6	13418	92.9	2416	100.0	0.406

In Tables 1-4 we can see that the approximation procedure improves computation times more when the number of inputs and outputs (v) is large. For example, in Table 4 ($v = 5$) the computation times are usually reduced only by 40 – 80 %, compared to Table 1 ($v = 20$) where the computation times are reduced by more than 92 %. There is not a clear connection between the best b value and the number of inputs and outputs in Tables 1-4.

As is obvious, the computation time needed to solve the problem depends on the value of the parameter b selected. In addition to b, the complexity of the approximation procedure depends on n, v and d. To estimate the complexity of the procedure when the best b is used for each problem, the following regression model was used:

$$t = k \cdot n^a v^c d^f \varepsilon \qquad (3)$$

where ε is an error term whose logarithm is assumed to be normally distributed, and k, a, c and f are parameters to be estimated. The logarithm of the model (3) was taken and used in a linear regression to estimate the parameters. Table 5 represents the results.

Table 5. The Estimates of the Regression Coefficients for Computation Times in Tables 1-4 ($R^2 = 0.990$)

	Coefficients	*Standard Error*	*Lower 95%*	*Upper 95%*
$\hat{\log}(k)$	-7.220	0.166	-7.555	-6.885
$\hat{a}$	2.032	0.032	1.966	2.098
$\hat{c}$	1.457	0.084	1.288	1.627
$\hat{f}$	1.094	0.027	1.039	1.150

From Table 5 we can see that the model fitted very well with the data. The number of DMUs has the highest impact on computation times while the number of inputs and outputs has a clearly smaller effect. Interestingly, the relationship of density on computation times in these problems is almost linear. In Korhonen and Siitari (2007) the estimates for the regression coefficients were -9.175, 2.472, 1.667 and 0.435 for $\log(k)$, a, c and f respectively. This indicates that the computation times increase less with respect to growing n and v using the algorithm developed in this paper. However, the estimate of the regression coefficient for d is clearly higher in Table 5. This follows from the fact that the increased efficiency of the algorithm developed in this paper is dependant on finding inefficient units in Step 2. The fever there are inefficient units (the higher d), the smaller the usefulness of the algorithm is.[13]

6.2 Adaptively Selecting the Value for Parameter b

Tables 1-4 reports the computing times when the empirical best value for the parameter b is used. In practice, an optimal value for b depends on the density of the problem, which is not known beforehand. That is why we need a method to select a value for b before knowing the true efficiency density.

Figure 4 illustrates the computing times when the number of inputs and outputs is 20, the number of units is 20000 and the density category is 22.9%. Computing times varies as a function of parameter b.

[13] In a hypothetical problem where all the units were efficient, the sum of computation times in Steps 1 and 3 would approximately equal the total computing time using the approach described in Korhonen and Siitari (2007). The additional time spent in Step 2 calculating the super-efficiencies would be an extra burden, because all the units checked in Step 2 would be either efficient or super-efficient.

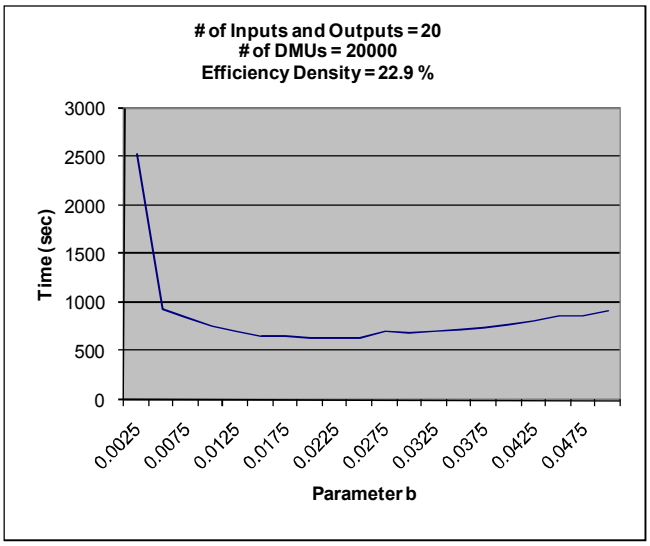

Figure 4. Computing Times (s) as a Function of Parameter *b*

In Figure 4 the best value for parameter b is 0.0225. With the best value of *b* the computing time needed to solve the problem is 745 seconds which is less than 6% of the reference case (see Table 1). Compared to the reference case, computing times are drastically smaller for all the *b* values used. Also, near "optimal" times are achieved using a wide range of *b*.

In Figure 4, computing times first decrease rapidly as *b* increases. After reaching a threshold value 0.005 the decrease in computing times becomes less rapid. On the other hand, after *b* exceeds the best value, the time needed to solve the problem increases only slowly. A similar behavior is found in all the problems studied. Based on this, we could select a sufficiently high value for *b* so that it is guaranteed that *b* is greater than the threshold value regardless of the true efficiency density. For example, if we selected $b = 0.03$ for all the problems represented in Table 1 (where the number of inputs and outputs is twenty) the smallest reduction in computation times compared to the reference case would be 89 %. On the other hand, the computation times would, on average, be about 100 % higher compared to the situation where a best value for *b* was used (see Figures 5-7 and Table 7 below).

To develop a method to select a value for *b* before the actual efficiency density is known we can use the information that is available from the previous problems solved. From Tables 1-3 we can see that *b* increases when the efficiency density increases. Also, we see that *b* decreases when the number of DMUs under evaluation increases. The effect of the number of inputs and outputs is unclear. To evaluate the relative importance of these variables (the number of DMUs *n*, the number of inputs and outputs *v* and the efficiency density *d*), on the best value for parameter *b*, we used the following regression model:

$$b = \alpha \cdot n^{\beta} v^{\gamma} d^{\delta} \varepsilon \qquad (4)$$

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 linear regression to estimate the parameters. The data to estimate the model was taken from the Tables 1-3 (i.e. the models where the number of inputs and outputs were greater than five) because for small dimensional problems

the reference method (a case where b = 1) is fast enough for most purposes. The results for model (4) are represented in Table 6.

Table 6. The Estimates of the Regression Coefficients for the Best Value of Parameter b (R^2 = 0.792)

	Coefficients	*Standard Error*	*P-value*	*Lower 95%*	*Upper 95%*
$\hat{\log(\alpha)}$	-0.043	0.259	0.870	-0.564	0.478
$\hat{\beta}$	-0.322	0.051	0.000	-0.425	-0.219
$\hat{\gamma}$	-0.066	0.131	0.616	0.198	-0.330
$\hat{\delta}$	0.443	0.043	0.000	0.357	0.529

From Table 6 we can see that the number of DMUs and the efficiency density has the highest impact on the best b. Constant α and the number of inputs and outputs v were not statistically significant using the data from Tables 1-3 (if α and v were dropped from the model, R^2 would have been 0.789)[14].

Using the regression model (4) and the estimates for the regression coefficients it is possible to develop an algorithm that adaptively selects b using the information available during Step 1.1 of the procedure developed in section 5. Step 1.1 could be modified as follows:

Step 1.1:

Step 1.1.0: Evaluate the first fifteen units and append the new efficient and inefficient units into the sets E and I respectively. Redefine U := U – E – I. Define $\hat{b} = \hat{\alpha} \cdot n^{\hat{\beta}} v^{\hat{\gamma}} \hat{d}^{\hat{\delta}}$, where $\hat{\alpha}$, $\hat{\beta}$, $\hat{\gamma}$ and $\hat{\delta}$ are the regression coefficients estimated in Table 6 and $\hat{d}$ is the proportion of units found efficient relative to units that have been identified efficient or inefficient. The estimate for d must be greater than zero so that $\hat{b}$ would be non-zero. That is why a lower bound of 0.005 is set for $\hat{d}$. Thus, $\hat{d} = \max(\frac{E}{E+I}, 0.005)$ is defined. Next, we define b to be the proportion of units that have been diagnosed either efficient or inefficient relative to all units i.e. $b = \frac{E+I}{D}$.

Step 1.1.1: Evaluate the efficiency of the next unknown unit. If the unit is found efficient or inefficient append it into sets E or I respectively. Recalculate U, $\hat{d}$, $\hat{b}$ and b.

Step 1.1.2:

If (U = ∅)

E is the set of efficient units in D. The procedure can be stopped.

Else If ($b \le \hat{b}$)

[14] Although α and v were not statistically significant we do not drop them from the model (4), because it is possible that they would become significant if a larger set of data were used.

Go to Step 1.1.1

Else

Go to Step 2.

The above modification of the approximation procedure starts by evaluating fifteen first units to get an initial approximation for the efficiency density $\hat{d}$. After the initialization, the procedure compares the current proportion of DMUs evaluated (b) to the estimated proportion of the units that, according to model (4), should be evaluated ($\hat{b}$). If an early identification of efficient units (Theorem 2) is used, the calculated value for $\hat{b}$ is actually a kind of an upper bound for parameter b because the procedure is biased to find efficient units during the initial evaluation rounds. This might be a desired phenomenon because having a too large b is a lesser problem than having a one that is below the threshold value (see, Figure 4).

If the procedure is frequently used to solve DEA –problems, it is possible to update the regression coefficients using new information available from the solved problems. This is especially useful option if computer idle time can be used to solve the problems using a wide range of parameter values b around an estimated best $\hat{b}$ value. In that case, the parameter value that in practice solves the problem fastest could be directly used in updating the estimates of the regression coefficients in the model (4).

Table 7 reports the computing times for the above presented adaptive parameter b selection procedure. The problems solved are from the Table 1 where the number of inputs and outputs is twenty. As a reference, Table 7 also reports the computation times for the best b value (as reported in Table 1) and a case where a constant parameter value $b = 0.03$ is used for all the problems.

Table 7. Computation Times (s) with a Different Parameter Value Selection ($v = 20$)

Density	# of Alt.	b = adaptive Time(s)	b = best Time(s)	b = 0.03 (constant) Time(s)
6.4 %	5,000	16	14	17
	10,000	40	36	55
	15,000	59	50	116
	20,000	84	84	252
	25,000	159	121	398
	50,000	338	321	1724
10.0 %	5,000	22	17	21
	10,000	69	69	86
	15,000	143	137	198
	20,000	270	240	350
	25,000	428	365	576
	50,000	1260	1256	3597
22.9 %	5,000	49	46	51
	10,000	166	162	164
	15,000	445	434	445
	20,000	827	745	778
	25,000	1121	1105	1273
	50,000	5296	4497	6239

From Table 7 we can see that the adaptive procedure performs relatively well against the best b selection. The biggest increase is found in a problem with 25000 units in a density category 6.4% where the computation time is increased by 31%. Also, the constant b selection (b = 0.03) performs quite well in all the other problems except the largest problems with 50000 units.

The computing times represented in Table 7 are illustrated in Figures 5-7. In Figure 5 the efficiency density is 6.4%, in Figures 6 and 7 the density is 10.0% and 22.9% respectively.

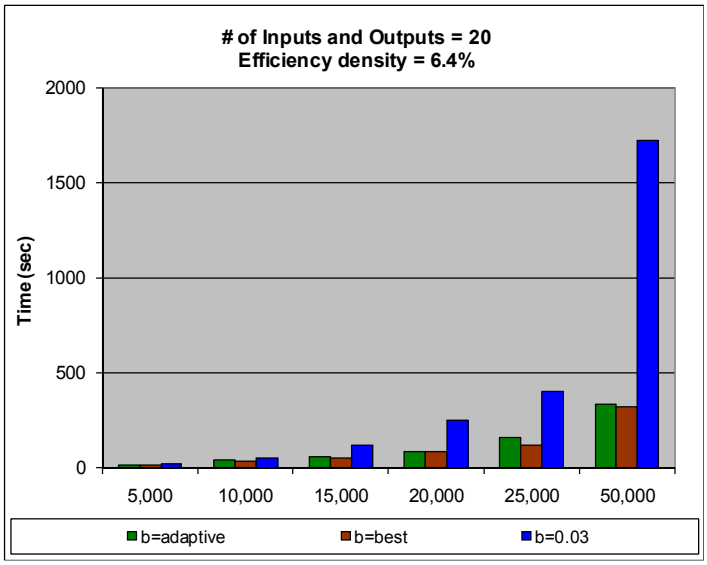

Figure 5. Computing Times (s) with an Average Efficiency Density of 6.4%

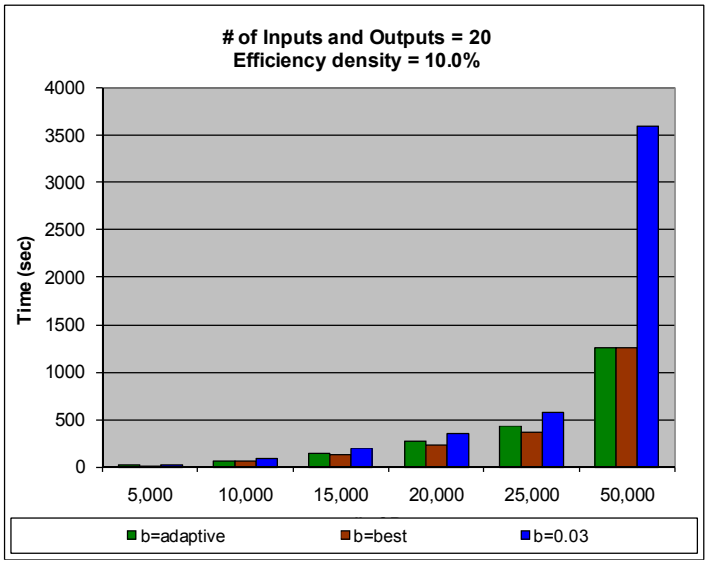

Figure 6. Computing Times (s) with an Average Efficiency Density of 10.0%

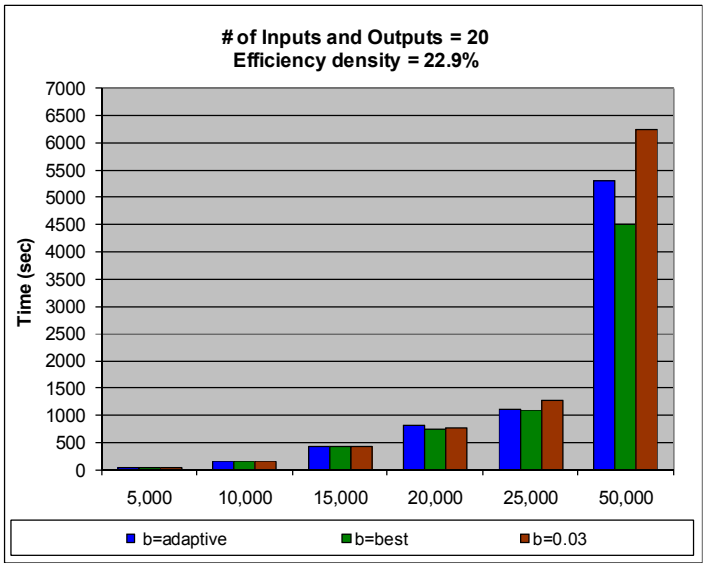

Figure 7. Computing Times (s) with an Average Efficiency Density of 10.0%

6.3 Using the Adaptive Selection of b with 100K Problems

Table 7 reported the computing times using the adaptive selection of parameter *b*. However, the adaptive selection of *b* did not completely reflect the real-world situation where a best value for *b* is not known beforehand, because the regression coefficients in model (4) were calculated using the known best values of *b* for these problems. To reflect a more realistic scenario, we used the adaptive procedure developed in section 6.2 with some new problems that were not used in the estimation of the regression coefficients. The problems had 100 000 units and twenty inputs and outputs. There was one problem in each density category (low, average and high). Table 8 reports the computing times of the three problems with 100 000 units. The interpretation of columns in Table 8 is the same than in Tables 1-4, except that the density column in Table 8 reports the true efficiency density of a problem (in contrast to average efficiency densities in Tables 1-4). Also the column that reports the value for *b* (*Adaptive b*) is not an empirically found best value but the value that the adaptive procedure of section 6.2 generated.

Table 8. Computing Times (s) of Problems with 100 000 Units ($v = 20$)

		Adaptive	Step1		Step2		Step3		Total
Density	# of Alt.	*b*	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)	# of Alt.	Time(s)
1.8%	100,000	0.0050	55	256	487	5096	612	1844	1155
8.8%	100,000	0.0065	67	402	827	14157	5528	8808	6423
20.7%	100,000	0.0071	77	492	1016	27439	21387	20695	22481

From Table 8 we can see that the adaptive procedure performed relatively well if we consider applying the method to real-world situations. Computing times were all feasible, if we consider the typical need to solve the problem only once: around 19 min, 1 h 47 min and 6 h 15 min for low, average and high density categories respectively. All the problems used most of the time in Step 3. This leaves a possibility to improve the computing times even further by selecting the sequence of units to be solved in

Step3 intelligently and/or using a procedure that is especially suitable for high efficiency density problems.

If we compare the *Adaptive b* values in Table 8 to the *Best b* values in Table 1, we can see that all the *Adaptive b* values are smaller than any of the *Best b* values. This reflects the influence of having a greater number of DMUs. However, compared to problems with 50 000 DMUs in Table 1 the *Adaptive b* values are fairly close to them (especially in the low density problem). This reflects the feature of the adaptive procedure to error on the upside when calculating the estimate for d if the early identification of efficient units is used. However, this is a desired phenomenon because we want to make sure that the adaptive b value is greater than the (unknown) threshold value.

7 Using the Approximation Procedure with the Dimensional Decomposition Approach

Korhonen and Siitari (2009) represented an algorithm where the identification of efficient units was accelerated by a dimensional decomposition procedure. The algorithm first partitions the original problem dimensionally into sub-problems and then identifies the efficient units of the sub-problems. The units (that are weakly efficient in the original problem) are then used as an approximation of the efficient frontier. Apart from building the approximated frontier the dimensional decomposition algorithm (henceforth procedure *Dim*) is similar to the approximation procedure presented in this paper (procedure *Fract*). Thus an interesting question is, if these approaches can be combined to further reduce the computation times. A necessary condition is that the methods tend to find different units to be used in the approximation of the efficient frontier.

Table 9 summarizes the approximation step of both algorithms for problems presented in Table 1. Column *# of Fract* states the number of units that are used to approximate the efficient frontier using the best value of parameter b for the procedure developed in section 5. *Column # of Dim* presents the number of units that are used to approximate the efficient frontier using the best number of decomposed problems (block-quantity $q = 3$) for the dimensional decomposition procedure (see, Korhonen and Siitari (2009)). Column *# of Distinct* states the number of distinct units found by the methods in the approximation step. (i.e. the cardinality of the union of the units found in the *Fract* and *Dim* approximation steps). *Ratio* compares the number of distinct units to the sum of units found in *Fract* and *Dim* approximation steps (i.e. *Ratio* = *# of Distinct* / (*# of Fract* + *# of Dim*)).

Table 9. Number of Distinct Units Found in the Approximation Steps ($v = 20$)

Density	# of Alt.	# of *Fract*	# of *Dim*	# of Distinct	Ratio
6.4 %	5,000	64	115	151	0.844
	10,000	113	124	191	0.806
	15,000	131	134	206	0.777
	20,000	134	134	204	0.761
	25,000	166	163	273	0.830
	50,000	231	206	364	0.833
10.0 %	5,000	87	151	205	0.861
	10,000	176	167	304	0.886
	15,000	184	212	358	0.904
	20,000	210	230	401	0.911
	25,000	305	257	467	0.831
	50,000	368	330	669	0.958
22.9 %	5,000	153	197	312	0.891
	10,000	257	261	496	0.958
	15,000	333	293	595	0.950
	20,000	454	328	740	0.946
	25,000	443	323	731	0.954
	50,000	641	445	1052	0.969

As can be seen form Table 9, procedures *Fract* and *Dim* tend to use different units to approximate the efficient frontier when best parameter values are used. In Table 9 around 90% of units that are found in the approximation steps are distinct. An explanation why these procedures tend to find different units in the approximation steps can be illustrated using the example first presented in section 3. In Figure 8 we have six DMUs each using one identical input to produce two outputs.

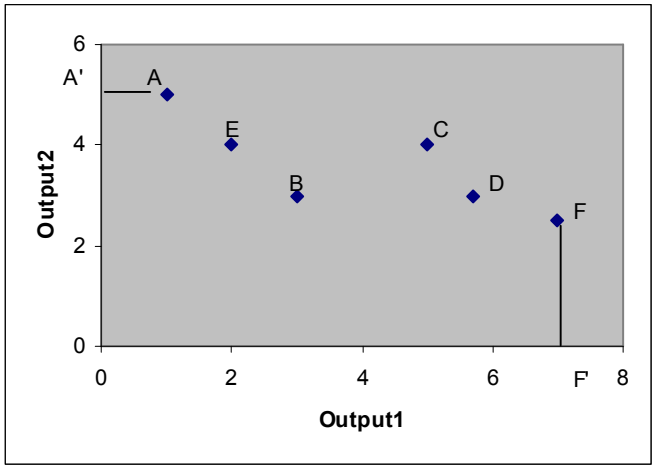

Figure 8. Illustrating the Approximation phase of Dimensional Decomposition Algorithm

In section 3 the *Fract* procedure started the search from efficient unit A and next evaluated B. During the search, unit C entered the basis and was diagnosed efficient. The procedure approximated the efficient frontier using units A and C. If the *Dim* procedure was used instead, the problem would have been decomposed dimensionally into two sub-problems. The first problem would have variables {Input, Output1}, the second problem would have variables {Input, Output2}. In the first sub-problem unit F would be found efficient (because its projection onto Output1 (F') is farthest from the origo). Similarly, the second sub-problem would have unit A as an efficient unit. Thus, procedure *Dim* would use units A and F to approximate the efficient frontier. The example illustrates the tendency of procedure *Dim* to select units that are performing well in some dimensionally decomposed sub-problem of the original problem (like units A and F). On the other hand, using procedure *Fract* the efficient units that belong to facets onto which many inefficient units are projected (like a unit C) have a higher likelihood to be selected.

The algorithm presented in section 5 could be modified to use also units that procedure *Dim* selects in its approximation step. To have these units (together with the units that unmodified procedure *Fract* uses) in the efficient frontier approximation, Step 2 could be rewritten as follows (procedure *Combined*):

Step 2: Using the Approximated Efficient Frontier to Identify Inefficient Units

Step 2.0: Initialization of *Dim* Procedure

Divide the set of input variables M and the set of output variables P into q subsets M_1, $M_2,\ldots,M_q$ and P_1, $P_2,\ldots,P_q$ as described in Korhonen and Siitari (2009). For each pair of subsets $\{M_i , P_i\}$ form a new dimensionally decomposed DEA sub-problem A_i (i = 1, 2, …, q); q is an integer valued block-quantity parameter whose value must be determined beforehand ($1 \leq q \leq \max(m, p)$).

Step 2.1: Selecting Units Using *Dim* Procedure

Solve each of the sub-problems A_i. Get the union of efficient units W in each sub-problem.

Step 2.2: Formulate the problem as model (2). Use only units $E \cup W$ as potential basic variables ($D_h = E \cup W$).

Step 2.3: Evaluate the super-efficiencies of all the units in the set $U - W$ ($D_k = U - W$). Append all inefficient units found in the approximated problem (2) into set I_k. Using Theorem 4 we can redefine $I = I + I_k$ and $U = U - I_k$.

Table 10 reports the computing times and empirically found best parameter values for procedures *Fract*, *Dim* and *Combined*. The problems solved are from Table 1 where the number of inputs and outputs is twenty. *Fract* and *Dim* have only one parameter whose value affects the computation times (b and q respectively). In the Combined procedure two parameter values have to be determined (b and q). For procedure *Fract* the computing times were calculated as for Table 1. For *Combined* the computing times were calculated using parameter values for b from 0.001 to 0.02 with increments of 0.0005. Parameter values for q in both *Dim* and *Combined* - procedures were 2, 3, 4, 5 and 6.

Table 10. Computation Times (s) Using Different Approximation Approaches ($v = 20$)

Density	# of Alt.	Fract		Dim		Combined		
		Time(s)	*b*	Time(s)	*q*	Time (s)	*b*	*q*
6.4 %	5,000	14	0.0125	16	3	13	0.0075	5
	10,000	36	0.0125	42	3	31	0.0050	4
	15,000	50	0.0100	63	3	48	0.0035	4
	20,000	84	0.0075	86	3	76	0.0050	5
	25,000	121	0.0075	118	3	107	0.0015	4
	50,000	321	0.0051	322	3	291	0.0035	5
10.0 %	5,000	17	0.0175	20	3	15	0.0025	5
	10,000	69	0.0175	58	3	50	0.0020	4
	15,000	137	0.0125	124	3	105	0.0015	4
	20,000	240	0.0100	203	3	172	0.0025	4
	25,000	365	0.0125	299	3	290	0.0050	5
	50,000	1256	0.0075	1205	3	1060	0.0025	5
22.9 %	5,000	46	0.0275	42	3	38	0.0100	5
	10,000	162	0.0250	169	3	146	0.0020	4
	15,000	434	0.0225	370	3	352	0.0100	5
	20,000	745	0.0225	680	3	651	0.0100	4
	25,000	1105	0.0175	1064	3	979	0.0100	5
	50,000	4497	0.0125	4341	3	4206	0.0075	4

From Table 10 we can see that the *Combined* -procedure was fastest in all the problems evaluated. Compared to *Fract* (*Dim*), *Combined* was on average 15% (13%) faster. Procedure *Fract* tends to perform slightly better relative to *Dim* in density category 6.4%. In higher density categories procedure *Dim* performs better than *Fract*. The best block-quantity parameter (q) value for *Dim* was 3 in all the problems evaluated. For procedure *Combined* the parameter b (q) values were smaller (higher) in all the problems compared to *Fract* (*Dim*).

The computing times represented in Table 10 are illustrated in Figures 9-11. In Figure 9 the efficiency density is 6.4%, in Figures 10 and 11 the density is 10.0% and 22.9% respectively.

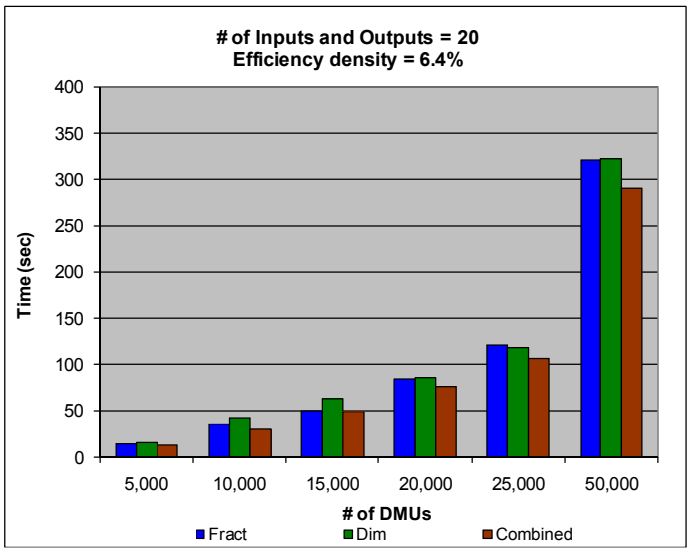

Figure 9. Computing Times (s) with an Average Efficiency Density of 6.4%

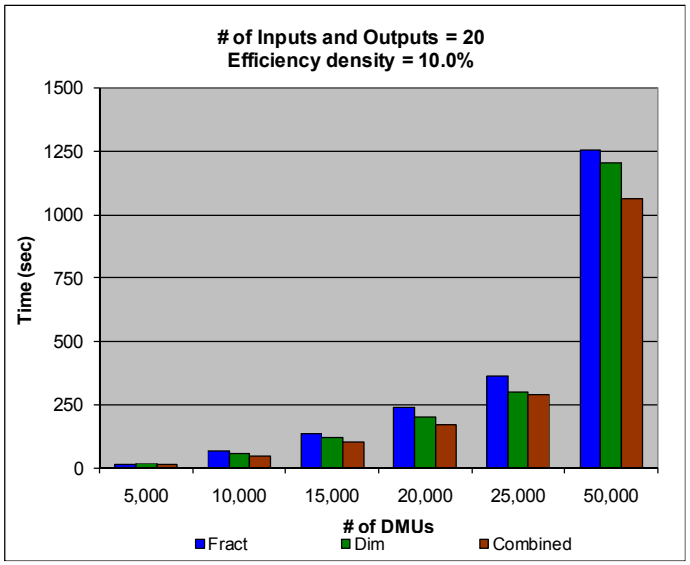

Figure 10. Computing Times (s) with an Average Efficiency Density of 10.0%

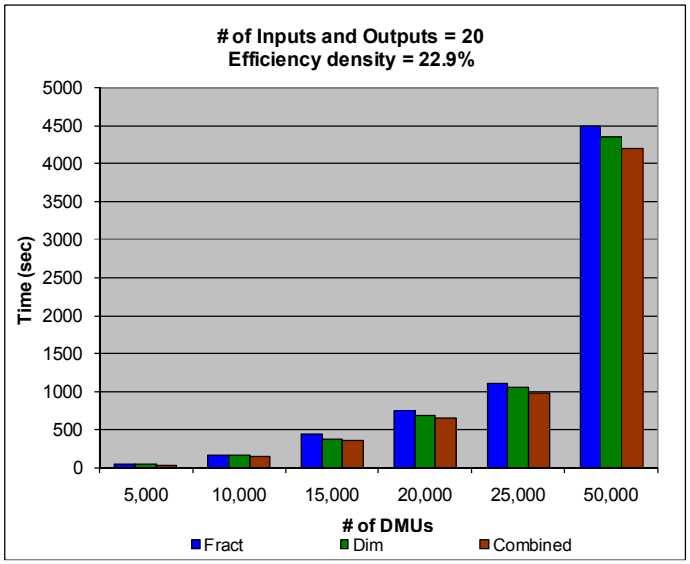

Figure 11. Computing Times (s) with an Average Efficiency Density of 22.9%

8 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we developed an efficient frontier approximation method to reduce the computation times when classifying decision making units (DMUs) inefficient and efficient in the Data Envelopment Analysis (DEA) approach. The algorithm developed in this paper first identifies a subset of efficient units. This subset of efficient units spans the approximated efficient frontier. The subset can be rapidly identified by using the fact that, if the problem is kept lexicographically optimal, every unit entering the basis is known to be efficient. Using an appropriately sized subset of efficient units, it is possible to recognize most of the inefficient units swiftly by calculating their (super-) efficiencies with regards to the approximated frontier.

By selecting properly the parameter value that determines the fraction of units to be identified in the frontier construction phase, the algorithm decreased drastically the

computing times in all the problems tested. The algorithm worked robustly with a wide range of parameter values. This fact enabled us to develop a simple adaptive procedure to select the parameter value during the solution process.

We compared the computational times achieved by the algorithm developed in this paper to the algorithms presented in our earlier papers Korhonen and Siitari (2007) and (2009). We also shortly represented and tested a combined algorithm that, in the approximated frontier construction phase, utilizes the dimensional decomposition idea presented in Korhonen and Siitari (2009).

In the future, our purpose is to study the effect of the sequence of the units to be checked. Selecting the sequence intelligently, it is probable that the computation times in the efficient unit identification problem discussed in this paper can be decreased further. The sequence of the units to be analyzed is also important when we calculate the efficiency scores using the set of efficient units.

REFERENCES

Andersen, P., and Petersen, N. C. (1993), "A Procedure for 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. (1994). "Parallel and Hierarchical Decomposition Approaches for Solving Large-Scale Data Envelopment Analysis Models", Technical Report 94-CSE-6, Southern Methodist University, Department of Computer Science and Engineering, Dallas, Texas.

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

Chen, W-C., Cho W-J. (2009), "A Procedure for Large-Scale DEA Computations", Computers & Operations Research, 36 1813-1824.

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. (1998). "An Algorithm for Identifying the Frame of a Pointed Finite Conical Hull", INFORMS Journal of Computing 10, 323-330.

Dulá, J.H., and Thrall, R.M. (2001), "A Computational Framework for Accelerating DEA", Journal of Productivity Analysis, 16, 63-78.

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.

Dulá, J. H. (2008), "A Computational Study of DEA with Massive Data Sets", Computers & Operations Research, 35 1191-1203.

Kao, C. and Liu, S.-T. (2000). "Fuzzy Efficiency Measures in Data Envelopment Analysis", Fuzzy Sets and Systems, 113 427–437.

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 Luptacik, M. (2004), "Eco-efficiency Analysis of Power Plants: An Extension of Data Envelopment Analysis", European Journal of Operational Research 154, 437-446.

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

Korhonen, P. and Siitari, P. (2009), "A Dimensional Decomposition Approach to Identifying Efficient Units in Large-Scale DEA Models", Computers & Operations Research 36, 234-244.