Data Envelopment Analysis and Big Data

Dariush Khezrimotlagh , Joe Zhu , Wade D. Cook , Mehdi Toloo

PII: S0377-2217(18)30912-3
DOI: https://doi.org/10.1016/j.ejor.2018.10.044
Reference: EOR 15436

To appear in: *European Journal of Operational Research*

Received date: 9 May 2018
Accepted date: 26 October 2018

Highlights

- A new framework is proposed to deal with large-scale data envelopment analysis.
- Decreases the computational time to measure the performance scores of big data sets.
- The fastest available data envelopment analysis technique for big data.
- A framework of choice to evaluate large-scale data envelopment analysis.
- Our method is easy to implement.

Data Envelopment Analysis and Big Data

Dariush Khezrimotlagh
Department of Mathematics, Pennsylvania State University - Harrisburg, PA, USA
dk@psu.edu

Joe Zhu*
College of Auditing and Evaluation, Nanjing Audit University, Nanjing, Jiangsu Province, 211815, China
Foisie Business School, Worcester Polytechnic Institute, Massachusetts, USA
jzhu@wpi.edu

Wade D. Cook
Schulich School of Business, York University, Ontario, Canada
wcook@schulich.yorku.ca

Mehdi Toloo
Department of Systems Engineering, VŠB-Technical University of Ostrava, Czech Republic,
mehdi.toloo@vsb.cz

*Corresponding Author

Oct 2018

Data Envelopment Analysis and Big Data

Dariush Khezrimotlagh, Joe Zhu, Wade D. Cook, and Mehdi Toloo

Abstract

In the traditional Data Envelopment Analysis (DEA) approach for a set of *n* Decision Making Units (DMUs), a standard DEA model is solved *n* times, one for each DMU. As the number of DMUs increases, the running-time to solve the standard model sharply rises. In this study, a new framework is proposed to significantly decrease the required DEA calculation time in comparison with the existing methodologies when a large set of DMUs (e.g., 20,000 DMUs or more) is present. The framework includes five steps: (i) selecting a subsample of DMUs using a proposed algorithm, (ii) finding the best-practice DMUs in the selected subsample, (iii) finding the exterior DMUs to the hull of the selected subsample, (iv) identifying the set of all efficient DMUs, and (v) measuring the performance scores of DMUs as those arising from the traditional DEA approach. The variable returns to scale technology is assumed and several simulation experiments are designed to estimate the running-time for applying the proposed method for big data. The obtained results in this study point out that the running-time is decreased up to 99.9% in comparison with the existing techniques. In addition, we illustrate the essential computation time for applying the proposed method as a function of the number of DMUs (cardinality), number of inputs and outputs (dimension), and the proportion of efficient DMUs (density). The methods are also compared on a real data set consisting of 30,099 electric power plants in the United States from 1996-2016.

Keywords: Data envelopment analysis (DEA); Big data; Performance evaluation, Simulation.

1. Introduction

Data envelopment analysis (DEA) was developed by Charnes *et al*. (1978) to assess the performance of a set of homogeneous decision-making units (DMUs), with multiple inputs and multiple outputs. Using linear programming (LP), DEA classifies DMUs into two mutually exclusive and collectively exhaustive groups (efficient DMUs and inefficient DMUs) and measures the performance score of each DMU. As the number of DMUs increases (e.g., a Veterans Affairs' study of over 20,000 hospitals), a large number of LPs should be solved.

There are several studies in the DEA literature, starting from the study of Ali (1990, 1993), to provide a theoretical framework to reduce the computation time for applying DEA models. For instance, Dulá and Helgason (1996) developed a procedure to decrease the size of LPs for a finite set of DMUs to no more than the number of extreme DMUs in the convex hull (see also Dulá *et al*., 1998). Barr and Durchholz (1997) proposed a hierarchical decomposition (HD) procedure to decrease the running-time for solving large-scale DEA problems. Their suggestions in software implementations were to partition DMUs into smaller groups, and gradually drop the corresponding decision variables of a known inefficient DMU from the subsequent problem. Dulá (2008) addressed the impact of three parameters on the computation time for applying a DEA model, namely the number of DMUs involved (cardinality), number of inputs and outputs (dimension), and the proportion of efficient DMUs (density). Soon after, Dulá (2011) proposed an affective procedure, called build-hull (BH), to decrease substantially the running-time for finding all efficient DMUs. Some related studies on large-scale DEA can also be found in Sueyoshi and Chang (1989), Dulá and Thrall (2001), Chen and Cho (2009), Dulá and López (2013), Chen and Lai (2017), and others.

The main idea behind most proposed frameworks for partitioning DMUs into smaller groups is aimed at finding the set of efficient DMUs first, and after that calculating the performance scores of the inefficient DMUs (e.g. see Barr and Durchholz, 1997, Korhonen and Siitari, 2009). While the key in reducing the computation time when large data sets are present is to identify all efficient DMUs, it happens that as the dimension increases, the sample's density increases, and the running-time to find the set of efficient DMUs increases as well (e.g. see Dulá 2008, Chen and Cho, 2009). Thus, the running-time to calculate the DEA scores of all DMUs can be large.

The goal of this study is to develop a framework to substantially decrease the computation time for finding the set of all efficient DMUs in comparison with the existing methodologies and hence decrease the running-time to measure the performance scores of all DMUs. In other words, we develop a new method to significantly decrease the computation time needed to solve large-scale DEA problems. Our contribution is to select a subsample of DMUs and after that find the exterior DMUs of the hull of the selected subsample. We will next find the set of all efficient DMUs and measure the performance scores of all inefficient DMUs.

To demonstrate the effectiveness of our proposed methods, we use the variable returns to scale (VRS) technology (see Section 2 for more details about selecting VRS). We use Matlab2017a (student version) and a single laptop with an Intel® Core™ i7-7820HK CPU @2.90 GHz, 16 GB memory and a 64-bit Windows 10 operating system. Since different ways of programming, different software used, different devices, different settings, installed apps and the preferences in a system used can affect running-times in measuring the performance of a sample of DMUs, we show the rate of decreasing the running-time of our method relative to that of the traditional DEA approach, HD (hierarchical decomposition) and BH (build-hull). According to our knowledge, HD and BH are the most powerful and effective available techniques in the DEA literature. Since Dulá (2011) shows the power of BH versus HD, in this study we only show the power of our method versus BH, using a single computer with a small number of 8 processors.

The study is organized into eight sections. Section 2 presents the background of the study. Elementary definitions relating to DEA, as well as the three most effective methodologies in large-scale DEA, are introduced in Section 3. Section 4 introduces the used phrases. The method is developed in Section 5. Section 6 describes the preparation for applying our framework versus BH. The running-times of the proposed method versus those of the existing methodologies are presented in Section 7. Conclusions are given in Section 8.

2. Background

The standard DEA model originated by Charnes *et al*. (1978), is a radial constant returns-to-scale (CRS) model. In this study, we focus on the radial output-oriented variable returns-to-scale (VRS) model. The number of VRS-efficient DMUs is generally greater than the number of CRS-efficient DMUs, and the size of the VRS model is also larger than that of the CRS model. The VRS model (see model 1) has two forms, namely the envelopment and multiplier forms. The envelopment form contains $n+1$ decision variables, $m+s+1$ constraints, and n non-negativity restrictions; meanwhile, the multiplier form involves $m+s+1$ decision variables, n constraints, and $m+s$ non-negativity restrictions, where n is the number of DMUs and m and s are the numbers of inputs and outputs, respectively.

This study concerns the envelopment form in the first stage of the proposed method, because as quoted by Cooper *et al*. (2007) p. 52:

(i) The computational effort of LP is apt to grow in proportion to powers of the number of constraints. Usually, in DEA, n is considerably larger than $m+s$ (the rule of thumb), and hence it takes more time to solve the multiplier form than the envelopment form.

(ii) Since the memory size needed for keeping the basis (or its inverse) is the square of the number of constraints, the envelopment form is better fitted for memory saving purposes.

(iii) The multiplier form fails to provide a max-slack solution.

(iv) The interpretations of the envelopment form are more straightforward.

It is obvious that after the first stage of HD, BH, and our method, any DEA models (envelopment forms or multiplier forms) can be used to evaluate the performance of DMUs.

Moreover, the VRS model should be solved *n* times (once for each DMU) and consequently, as the cardinality increases, the number of decision variables increases, which leads to a significant increase in the number of required computations. Therefore, powerful computer systems in terms of memory capacity, CPU time, and advanced software, are needed to run a simple traditional DEA model such as the VRS model (see also Dulá, 2008).

Since the work of Ali (1993), several methods have been developed to initially find the set of efficient DMUs (called Stage 1) - in order to decrease the number of decision variables by removing the decision variables associated to inefficient DMUs - and, after that, apply the desired DEA model to measure the performance of the remaining inefficient DMUs. Ali (1993) suggests finding the non-dominated DMUs first and then updating the number of decision variables of a DEA model by gradually dropping the related decision variables to known inefficient DMUs. Barr and Durchholz (1997) followed the suggestions of Ali (1993) and used an HD procedure to partition DMUs into smaller groups, with approximately equal size. They also used several processors in parallel to decrease the computation time for measuring the performance scores of all DMUs. Dulá *et al*. (1998) proposed an approach to identify a minimal set of DMUs from which to extract the production possibility set that will allow one to obtain DEA results. Dulá and Thrall (2001) developed a method to deal with the four returns to scale technologies. From their approach, the non-decreasing/non-increasing returns to scale efficient DMUs can be extracted from the VRS-efficient DMUs, and from these three sets of efficient DMUs the set of CRS-efficient DMUs can be obtained. Dulá (2008) showed that the computation time to apply a traditional DEA model is nearly quadratic relative to the number of DMUs, and it is linear relative to the numbers of inputs and outputs (see also Dulá and López, 2009). Korhonen and Siitari (2009) proposed a hierarchical decomposition procedure with lexicographic parametric programming to decrease DEA computational burden when dimension is small. Chen and Cho (2009) also suggested an accelerating procedure to deal with large-scale VRS problems when the dimension is small. Dulá (2011) established a BH algorithm which can be used for all four returns to scale technologies to obtain DMUs' DEA scores in a timely manner. Dulá and López (2013) explored theoretical and computational challenges of frontier analysis when large sets of data are present. They provide a theoretical framework and specialized tools to deal with large-scale data in dynamic situations when data are constantly changed. Chen and Lai (2017) also proposed an algorithm to address the problem of LP size limitations for calculating the radial efficiency when large-scale data are present. From their algorithm LPs can be solved within a reasonable number of iterations without incurring extra running-time. The authors also highlighted that their algorithm is not designed to compete with other algorithms.

In this study, we propose a robust framework, including five simple steps, to substantially reduce the computation time of large-scale DEA, in comparison with the available techniques.

3. Introducing the used phrases

We first provide some elementary definitions to clarify the concepts and the meaning of the terminology used in this study. Suppose that a set $\mathfrak{D}$ includes n observations $\mathrm{DMU}_j, (j = 1, \dots, n)$, where each DMU_j has m non-negative inputs $\boldsymbol{x}_j = (x_{1j}, \dots, x_{mj})$ and s non-negative outputs $\boldsymbol{y}_j = (y_{1j}, \dots, y_{sj})$. We have the following definitions (Cooper *et al.*, 2007):

Definition 1: DMU_A dominates DMU_B if, and only if, (i) $\boldsymbol{x}_A \leq \boldsymbol{x}_B$ and (ii) $\boldsymbol{y}_A \geq \boldsymbol{y}_B$.

Definition 2: Let $\mathfrak{N}_A$ be the set of all DMUs that dominate DMU_A. DMU_A is a non-dominated DMU if $\mathfrak{N}_A = \{\mathrm{DMU}_A\}$, and the whole set of such DMUs is called the set of non-dominated DMUs, denoted by $\mathfrak{N}$.

Definition 3: DMU_A is called an efficient DMU if and only if it is dominated neither by any other DMUs nor by any virtual DMUs (generated by linear combinations of the other DMUs). The set of efficient DMUs is denoted by $\mathfrak{F}$.

Obviously, the set of non-dominated DMUs is a subset of the set of all observations, i.e., $\mathfrak{N} \subseteq \mathfrak{D}$. The set of efficient DMUs is also a subset of the set of non-dominated DMUs, i.e., $\mathfrak{F} \subseteq \mathfrak{N}$. In addition, a non-dominated DMU is not necessarily an efficient DMU, i.e., $\mathfrak{N} \nsubseteq \mathfrak{F}$.

The standard VRS model (output-oriented) in envelopment form (Banker *et al.*, 1984) is as follows, where DMU_l $(l = 1, \dots, n)$ is under evaluation.

$$
\begin{aligned}
&\max \varphi_l \\
&\text{s.t.} \\
&\textstyle\sum_{j=1}^{n} \lambda_j x_{ij} \leq x_{il},\ i = 1, \dots, m, \\
&\textstyle\sum_{j=1}^{n} \lambda_j y_{rj} \geq \varphi_l y_{rl},\ r = 1, \dots, s, \qquad (1) \\
&\textstyle\sum_{j=1}^{n} \lambda_j = 1, \\
&\lambda_j \geq 0,\ j = 1, \dots, n.
\end{aligned}
$$

Assume that a subsample of size p, denoted by $\mathfrak{D}^S$, is given from the set of observations, i.e., $\mathfrak{D}^S \subseteq \mathfrak{D}$. In general, p can be any natural number less than or equal to n, i.e., $|\mathfrak{D}^S| = p \leq n$. We note that, $\mathfrak{D}^S \cap \mathfrak{F}$ can be an empty set. We let $\text{DMU}_k^S = (\boldsymbol{x}_k^S, \boldsymbol{y}_k^S)$, for $k = 1, \dots, p$, denote the selected DMUs in the subsample $\mathfrak{D}^S$. The radial VRS model, based upon the subsample $\mathfrak{D}^S$ where DMU_l $(l = 1, \dots, n)$ is under evaluation, is formulated as follows:

$$
\begin{aligned}
&\max \varphi_l^S \\
&\text{s.t.} \\
&\textstyle\sum_{k=1}^{p} \lambda_k x_{ik}^S \leq x_{il}, \, i = 1, \dots, m, \\
&\textstyle\sum_{k=1}^{p} \lambda_k y_{rk}^S \geq \varphi_l^S y_{rl}\,, \, r = 1, \dots, s, \\
&\textstyle\sum_{k=1}^{p} \lambda_k = 1, \\
&\lambda_k \geq 0, \, k = 1, \dots, p.
\end{aligned}
\tag{2}
$$

Definition 3: DMU_l is called *best-practice* if, and only if, $\varphi_l^{S*} = 1$. A best-practice DMU is not necessarily efficient. The set of best-practices in $\mathfrak{D}^S$ is denoted by $\mathfrak{B}^S$. We also call φ_l^{S*}, the sub-efficiency score of DMU_l^S.

The number of decision variables in model (2) is $p + 1$ in comparison with that of model (1) which has $n + 1$ variabes. Therefore, using model (2), where $p \ll n$, we deal with an LP problem with a much smaller number of decision variables, and therefore the running-time to solve the VRS model (2) decreases substantially. In general, increasing the number of decision variables in an LP results in increasing the running-time to solve that LP. For example, the running-time to solve model (2) with size 20,000 and 2+2 input-output is 0.5062 seconds, on average, whereas the running-time to solve model (2) with size 2,000 and 2+2 dimensions is 0.0085 seconds, on average. In theory, the smaller number of decision variables in model (2), the lower required running-time for estimating the performance of all DMUs (e.g., see Barr and Durchholz, 1997). When the subsample includes all efficient DMUs, model (2) is equivalent to the standard VRS model (1) (see Ali, 1993).

Ali (1993) suggests eliminating the corresponding lambdas of dominated DMUs from models (1) and (2). He mentions that there might be a small number of totally dominated DMUs in the sample, nonetheless, such preprocessing for large-scale data is worthwhile. For example, for a set of 20,000 DMUs with 10+10 dimensions where each input (output) is uniformly distributed on the interval [10, 20], there are more than 98.8% non-dominated DMUs in the sample, on average. We do not consider these enhancements for either of the used framework (including our framework) to have transparent comparisons between the existing frameworks and a better understanding of their performances. We now introduce the most well-known methods to decrease the running-time for large-scale data.

3.1. RBE (Restricted Basis Entry)

An RBE procedure was proposed by Ali (1993) to decrease the size of the VRS LP gradually. The process starts by evaluating a DMU using model (1) which has $n + 1$ decision variables (n non-negative variables $\boldsymbol{\lambda}$ and a free variable φ). If the evaluated DMU is inefficient, its corresponding variable lambda is removed from model (1); thus, the model has n decision variables in the next iteration. The size of the LP decreases gradually, and it is constant when all inefficient DMUs from the sample are removed. See Dulá (2008) for a detailed discussion on RBE and related enhancements.

The main weakness of RBE is that the LPs are not independent of one another, unlike in the traditional approach. In other words, RBE is useful when few resources are available because when evaluating a set of 100K DMUs, 100K computers are used in parallel, the traditional approach does not need any improvement. However, RBE is not comparable with BH, therefore, we do not compare our approach with this procedure in this study. It should be noted here that the aim of parallel processing is driving all computations among different processors attached to the same computer. In other words, in this approach, the running-time decreases while the number of computations is fixed. In contrast to this approach, the aim of our suggested approach is reducing the number of computations which leads to improving the required running-time. It is plain to verify that applying a parallel approach to solve the reduced computations in our approach will lessen the running-time as well.

3.2. HD (Hierarchal Decomposition)

Barr and Durchholz, (1997) proposed HD as a mechanism to partition a given sample of n DMUs into Q blocks. HD needs three parameters to be introduced by the user, denoted by b, β and γ. The size of each block is at most b. The method is started by finding best-practices of each block, and then is repeated for the set of all found best-practices. In each run, the number of blocks is decreased, while the size of the blocks is increased using parameter γ. The process is stopped when the proportion of best-practices within all blocks is greater than the parameter β. The obtained best-practices in the last run are aggregated into one block in order to identify efficient DMUs. Now, the remaining DMUs are evaluated. For more details about the performance of HD see Barr and Durchholz, (1997), Dulá (2011) and Zhu *et al*. (2018).

Although more than n LPs might be solved in HD implementation, the procedure is very useful to decrease the running-time of evaluating large-scale DMUs where several processors are used in parallel. In other words, unlike RBE and similar to the traditional approach, the LPs in any block are independent of those in other blocks. Therefore, having several processors (or workers) in parallel substantially decreases the running-time and HD can be much faster than RBE. In addition, both HD and RBE can be combined to increase the performances of each other when few resources are available. See Dulá (2011) for the performance of HD versus BH.

3.3. *BH (Build-Hull)*

As quoted by Dulá and López (2013), BH is one of the fastest methods to find all efficient DMUs. BH can be introduced as the inverse of RBE, as the size of the used LP is gradually increased until all efficient DMUs are determined. BH uses model (3) and its dual (Dulá, 2011, p. 286).

$$
\begin{aligned}
&\min \alpha_l \\
&\text{s.t.} \\
&\textstyle\sum_{k=1}^{p} x_{ik}\lambda_k - \alpha_l \leq x_{il},\ i = 1, \dots, m, \\
&\textstyle\sum_{k=1}^{p} y_{rk}\lambda_k + \alpha_l \geq y_{rl},\ r = 1 \dots, s, \\
&\textstyle\sum_{k=1}^{p} \lambda_k = 1, \\
&\lambda_k \geq 0,\ k = 1, \dots, p, \\
&\alpha_l \geq 0.
\end{aligned}
\tag{3}
$$

The number of variables in the first LP in BH is 2 and is, at most, equal to the number of efficient DMUs in the sample. BH uses a minimal number of DMUs to envelop all DMUs and finds efficient DMUs one at a time, where each LP is dependent on the other LP in each iteration (similar to LPs in RBE and unlike HD). BH can be used when the dimension (the number of inputs and outputs) and/or the density (the proportion of efficient DMUs) are large or when there are not enough available resources.

In general, in the first iteration, $p = 1$ and model (3) has only two decision variables (i.e., a lambda and an alpha). The first DMU is selected using Ali's (1993) theorem which says "a DMU with the largest value of the first output lies on the efficient frontier". After that, a DMU is evaluated. If the DMU belongs to the corresponding hull generated by the first selected efficient DMU, then $\alpha^* = 0$ in model (3). Otherwise, the evaluated DMU might lie on the efficient frontier, that is, the evaluated DMU is in an unresolved situation. Now, the dual LP is solved for the unresolved DMU and based upon its optimal solutions and a Dulá's (2011) theorem, an efficient DMU is obtained and its corresponding lambda is added to model (3). If the earlier unresolved DMU does not belong to the new generated hull, the process is continued and another efficient DMU is added to model (3); otherwise, the unresolved DMU is an inefficient DMU, and the same process is continued for a new DMU. The process stops when there is no DMU to be evaluated.

In this study, we propose a new method which is significantly faster than all existing procedures, including BH, and without the need for a supercomputer or a set of parallel computers.

4. The proposed framework

We suggest the following algorithm to introduce our framework.

```
1. Start,
2. $\mathfrak{D} \longleftarrow$ get a sample of DMUs,
3. $\mathfrak{D}^S \longleftarrow$ select a subsample of $\mathfrak{D}$,
4. $\mathfrak{B}^S \longleftarrow$ find best-practices in $\mathfrak{D}^S$,
5. $\mathfrak{E} \longleftarrow$ find exterior DMUs in $\mathfrak{D} \backslash \mathfrak{D}^S$ respect to the hull of $\mathfrak{B}^S$,
6. If $\mathfrak{E} = \{\ \}$, then $\mathfrak{F} = \mathfrak{B}^S$ and all DMUs are already evaluated (go to 7).
   Otherwise,
         6.1. $\mathfrak{F} \longleftarrow$ find best-practice DMUs in $\mathfrak{D}^S \cup \mathfrak{E}$,
         6.2. Evaluate DMUs in $\mathfrak{D} \backslash (\mathfrak{D}^S \cup \mathfrak{E})$ respect to the $\mathfrak{F}$'s hull,
7. End.
```

We note that when the sample's density is small, the procedure can be ended in Step 6 above; hence decreasing the running-time substantially. In such a case, the selected subsample $\mathfrak{D}^S$ includes all efficient DMUs, i.e., $\mathfrak{F} \subseteq \mathfrak{D}^S$. Theoretically, when $\mathfrak{F}$ is not an empty set and its cardinality is large, we can expect a large density and presume a longer time to evaluate DMUs.

Our framework can also be introduced as an HD procedure with the difference being that instead of partitioning DMUs into several blocks and evaluating each block, we only use a subsample of DMUs (i.e., one block) and evaluate all other DMUs (i.e., DMUs in other blocks) based upon the selected subsample. The impact of this simple difference is profound in comparison with the existing procedures, such as BH. In addition, our procedure has all of the abilities of HD, but not vice versa. For example, our procedure has the ability of parallel processing, the most useful ability of HD. In contrast, HD always needs to evaluate remaining inefficient DMUs after finding all efficient DMUs, unlike our procedure which can be completed in Stage 6 above, without executing Steps 6.1 and 6.2. Moreover, for our procedure, the required number of efficient DMUs to build the hull is always less than or equal to that of HD. Indeed, HD uses all efficient DMUs to build the hull, even if they lie on a line-segment.

4.1. Selecting a subsample

Assume that for a given set of n DMUs, $\mathfrak{D}$, we have $p = \lfloor\sqrt{n}\rceil$, where $\lfloor . \rceil$ is the rounding function which rounds the number to the nearest integer. To construct a subsample with size p, we use the following theorem proposed by Ali (1993); that is, DMU_l is located on the VRS frontier if one of the following equations holds: $x_{il} = \min\{x_{ij} | j = 1, \dots, n\}$, for $i = 1, \dots, m$, and $y_{rl} = \max\{y_{rj} | j = 1, \dots, n\}$, for $r = 1, \dots, s$. In this study, we first select such DMUs (i.e., $m + s$ DMUs at most); that is, for each input a DMU with minimum value (total of m DMUs), and for each output, a DMU with maximum output value (total of s DMUs) are selected. To select the rest of DMUs, we use the following pseudocode to assign a pre-score to remaining DMUs of $\mathfrak{D}$, in a very negligible running-time.

For example, for a set of 20,000 DMUs with 10+10 dimensions, the average computation time of applying the following pseudocode (4) is only 0.31 seconds. Even if we consider a sample of 100,000 DMUs with 2+2 (or 10+10) dimensions and assume that all DMUs are efficient (100% density), the computation time to apply pseudocode (4) is 0.38 (or 1.87) seconds, on average.

```
For each unselected $\text{DMU}_j$,
{$u = 0$,
    {For each $i$ and $k_i$
        If $x_{ij} \le k_i$-th percentile of $\boldsymbol{x}^i$ then
          $u = u + 1$}                                        (4)
    {For each $r$ and $k'_r$
        If $y_{rj} \ge k'_r$-th percentile of $\boldsymbol{y}^r$ then
          $u = u + 1$}
pre-score_j = u }
```

In the above pseudocode, $\boldsymbol{x}^i$ and $\boldsymbol{y}^r$ represent the i-th inputs and the r-th outputs of all DMUs, $i = 1, \dots, m$ and $r = 1, \dots, s$; for instance $\boldsymbol{x}^1 = (x_{11}, x_{12}, \dots, x_{1n})$. The indexes, k_i and k'_r, are changed from 0-100 and refer to the percentiles of $\boldsymbol{x}^i$ and $\boldsymbol{y}^r$, respectively.

We sort DMUs in descending order by the assigned pre-scores, and the remaining DMUs (to construct a subsample with size p) are selected as those having the greatest pre-scores. By such an approach, DMUs which have higher values than the third quartiles of output values and lesser values than the first quartiles of input values have more chances to be included into the subsample.

We note that, in this study, our goal is not to improve the power of pseudocode (4) because its current power is enough to show the strength of our procedure in comparison with the available procedures and to decrease the running-time when a large-scale data is present.

It is important to recognize that out of a sample of 10,000 DMUs, we only select 100 ($=\sqrt{10{,}000}$) DMUs. Of course, when the density is large (e.g. 10% or more), we are aware that the subsample with size 100 does not include most of efficient DMUs, but the procedure is robust enough to determine most of inefficient DMUs in a very short period of time. In other words, this approach can decrease the original sample size to a subsample with much smaller size. Therefore, the search for finding all efficient DMUs can be continued on a subsample with high density, but with smaller size. In this step, we can also use BH, but we do not apply such an enhancement to show the power of our procedure versus BH.

When the dimension is large (e.g., 10+10) we suggest letting $p=\left\lfloor \min\{\sqrt{mn}, n/2\}\right\rfloor$ or $p=\left\lfloor \min\{\sqrt{(m+s)n}, n/2\}\right\rfloor$. However, in this study, we only let $p=\lfloor\sqrt{n}\rfloor$ in all situations to illustrate the power of pseudocode (4).

5. Preparation to apply our procedure

5.1. Computer and software

Although we can use several computers in parallel to have a faster procedure, we used only a single computer to show the power of our framework versus BH approach. In other words, it is obvious that if we use two (or more) computers in parallel, the recorded running-time for our proposed method will be much smaller than when we use one single computer.

The computer used is a single laptop with an Intel® Core™ i7-7820HK CPU @2.90 GHz, 16 GB memory and a 64-bit Windows 10 operating system. The CPU is a quad-core which has eight logical processors (i.e., 8 workers). We utilized Matlab2017a (student version) without combining CPLEX, and this is the same for BH as well. MATLAB allows parallel processing using a par-for-loop instead of the traditional for-loop. A par-for-loop executes a series of calculations in the loop body in parallel to decrease the running-time of calculations. For more information see https://www.mathworks.com/help/matlab/ref/parfor.html. We can change the number of workers and preferences from 1 to 8 workers. These preferences are also the same for both approaches.

We note that to execute the iterations in parallel, all interaction should be independent from one another. After finding the set of efficient DMUs, all procedures use the par-for-loop to measure the performance scores of inefficient DMUs. Therefore, the running-times to evaluate the inefficient DMUs are about the same for both procedures, except for our procedure when the algorithm is ended in Step 6 above, without executing Steps 6.1 and 6.2.

5.2. Datasets

In this study, we use more than 150 samples including 102 real datasets and generated datasets using a uniform approach on the interval [10, 20]. We use a uniform distribution because neither the possible correlation between inputs and outputs nor the way of generating datasets affect the performance of our method versus BH. The following are the list of the used datasets in this study:

Dataset_1) A real data set of power plants including 30,099 DMUs with 2+4 dimensions (2 inputs, one good output and three bad outputs). The number of VRS efficient DMUs in this data set is 2,450.

Dataset_2) 100 samples of size 10,000, which were randomly chosen from Dataset_1. This dataset is examined to compare the methods on 100 real-datasets when density can be varied from one sample to another.

Dataset_3) The real data set of Plant Energy including 30,099 DMUs with 2+1 dimensions (2 inputs and one good output).

Dataset_4) Random samples generated by a uniform approach, for each 1,000, 5,000, 10,000, 15,000, 25,000, 50,000 cardinalities and 1+1 to 10+10 dimensions. This dataset is considered to compare the methods when cardinalities and dimensions increase.

For all of the above data sets, using only a single computer, the results confirm that our method has a much smaller running-time in comparison with that of BH, and the power of our method is not represented by a chance. In other words, the differences between our proposed method and BH are remarkable and, regardless of the data set, our method performs faster.

5.3. Our procedure versus BH

Clearly, the LPs in BH are not independent from one iteration to another and the algorithm waits until one integration is finished prior to starting the next iteration. In contrast, while BH solves one LP at a time, our procedure allows for solving several LPs at a time, and hence we expect to have a smaller running-time.

On the other hand, to see the power of our framework versus BH, assume that a sample of 10,000 DMUs with 20 known efficient DMUs (i.e., 0.2% density) is given. Also assume that there are no more efficient DMUs in the sample, but none of the procedures are aware of this situation. Our robust method simply uses the given 20 efficient DMUs, measures the performance of all inefficient DMUs, and completes the task. In contrast, BH builds the hull of the given 20 efficient DMUs first. Then, it needs to check the remaining DMUs to find whether or not there are other efficient DMUs in the sample. This means that BH evaluates each inefficient DMU one at a time until it finds out that the first given 20 efficient DMUs are the exact efficient DMUs in the sample. After this elapsed time, BH still needs to evaluate the inefficient DMUs. As a result, we can expect BH to have a weaker performance than our method.

6. The running-time of the proposed framework

In this section, we record the running-times to find the set of efficient DMUs or the minimal number of DMUs to build the VRS hull, denoted “Stage 1”, and the total running-times to measure the performance scores of all DMUs, denoted “Total”. BH finds the minimal number of DMUs to build the hull. We utilize different datasets to illustrate the superiority of our method over BH.

It is obvious that, as the number of DMUs increases, the running-time for the traditional approach rises sharply (see also Dulá 2008). For example, the average running-times of the traditional approach for a set of 16,000 and 20,000 DMUs are 5,294 seconds (88 minutes) and 10,123 seconds (168 minutes or almost 3 hours), respectively. In contrast, the average running-times when applying our method are 63 seconds and 78 seconds (1.3 minutes), respectively.

It is already proven that BH perform faster than the traditional approach. Dulá (2011) also shows the advantages of BH versus HD. In this study, we show that our proposed method (using a single computer only) is much faster than the fastest exiting methodology.

6.1. Comparing methods on Dataset_1:

The utilized data source for Dataset_1 is the Emissions and Generation Resource Database (E-GRID). E-GRID is a comprehensive source of data on the environmental characteristics of all electric power generated in the United States. These environmental characteristics include air emissions for nitrogen oxides, sulfur dioxide, carbon dioxide, methane, and nitrous oxide. This information can be used in DEA analysis of a plant’s efficiency and its impact on its carbon foot print. GRID2016 was released in February 2016, and includes data from 1996 through 2016, encompassing more than 4,600 power plants. The data for some years and some particular DMUs were not available. Overall, we could select 30,099 power plants (DMUs) and the following DEA inputs and outputs. The two inputs are (1) plant annual total heat used to generate the electricity, measured Millions of British Thermal Units (MMBtu), and (2) net generator (nameplate) capacity measured in Megawatt (MW) units. The outputs include (1) good output: the net electricity generated in MW hours (MWh), and (2-4) three bad outputs: annual NOx, SO2, and CO2 emissions, all measured in tons emitted, respectively. Each bad output is treated by subtracting the maximum value of the corresponding year with the recorded output value so that they can be used as DEA outputs (see Seiford and Zhu, 2005).

There are 2,450 VRS efficient DMUs in the sample of 30,099 DMUs. BH illustrates that the minimum number of efficient DMUs to build the hull is 84, and then measures the efficiency of DMUs with the LP problem of size 85 (84 lambdas and 1 alpha). The running-time of our method in Stage 1 (Total) is 66.16 (128.27), whereas the running-time of BH in Stage 1 (Total) is 397.45 (442.80) (see also Figure 2). It can be seen that BH uses the minimum number of DMUs to build the hull; however, it can only solve one problem at a time in Stage 1. In contrast, our method using a single computer decreases the running-time of BH by 83% in Stage 1 and 71% in Total in this example and both stages are completed before BH finds the 84 efficient DMUs to build the hull. If we use the available power of a single computer with 12 or more processors or multiple computers in parallel, the running-time of our approach sharply decreases. On the other hand, although the absolute improvement time on this dataset is 314.528 seconds (5.24 minutes), under our DEA context, assume one wants to run a bootstrapping method on Dataset_1. The difference of 5.24 minutes will become 524 minutes (almost 9 hours) for 100 runs, and 87 hours (almost 4 days) for 1000 runs.

6.2. Comparing methods on Dataset_2:

Our method and BH are compared on Dataset_2, 100 real-datasets when density can be varied from one sample to another. The average running-time of our method in Stage 1 is 24.27 seconds, where the median and the third quartiles are 20.83 and 23.03 seconds, respectively, representing several outliers (see Figure 1). The average number of efficient DMUs to build the hull is 293.35 for our method. Even if we consider the outliers in this experiment, such as the maximum running-time of Stage 1 (Total) for our method (representing the weakest performance of pseudocode 4), the running-time of Stage 1, using a single computer, is just 50.62 seconds with the total of 63.04 seconds, which is still less than 84.33 (98.33) seconds, the minimum running-time of BH in Stage 1 (Total).

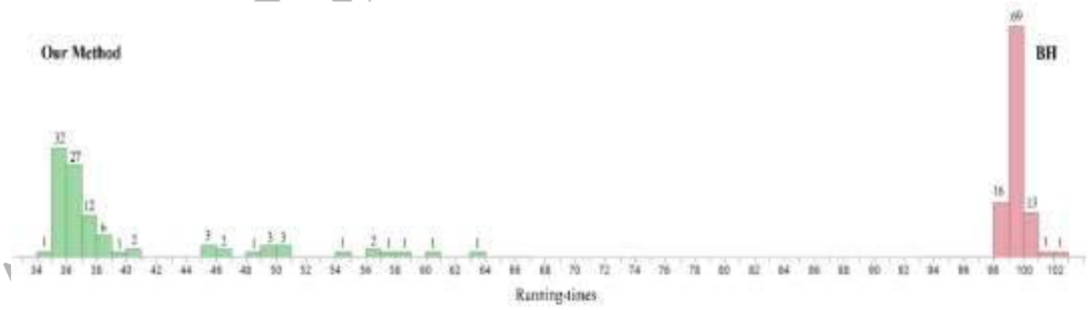

Figure 1: The total running-times of our method versus BH on Dataset_2.

6.3. Comparing methods on Dataset_3:

In this experiment, dimensions are 3 and the number of efficient DMUs in the sample is 16. The minimum number of DMUs to build the hull is 15, as BH identifies.

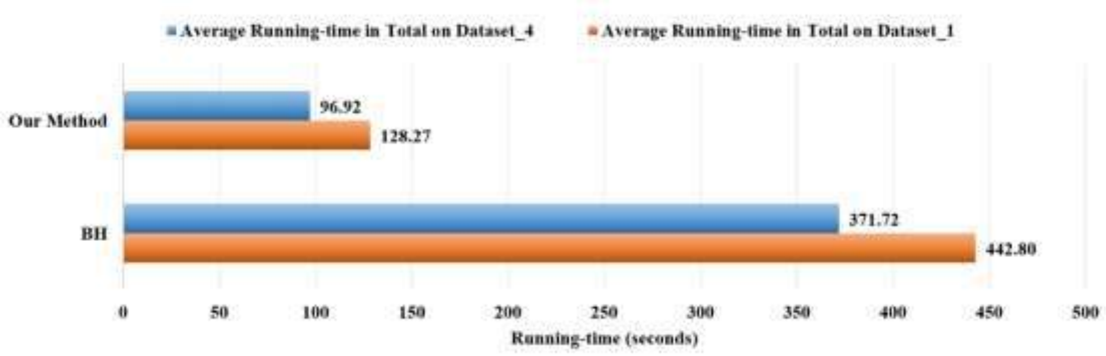

Figure 2: Comparing the results of methods on Datasets_1 & 3.

As shown in Figure 2, our method substantially performs better than BH. The total running-time for our model on Dataset_1 with 2+4 dimensions is still less than the total running-time for BH on Dataset_4 with 2+1 dimensions.

The running-times of our method in Stage 1 and Total are 52.34 and 96.92, in comparison with the running-times of BH in Stage 1 and Total 327.53 and 371.72, respectively. In other words, where our method needs only 52.3 seconds to find the efficient DMUs, BH requires 327.5 seconds to finish Stage 1. This represents that our method decreases the running-time of BH in Stage 1 by 84%. We can clearly see the advantages of our method in this situation using a single computer. Indeed, we just use the available power of the computer without adding any additional computers/processors, whereas BH solves one problem at a time. As the cardinality increases the weak performance of BH versus our approach is more transparent.

6.4. Comparing methods on Datasets_4:

In the previous subsections, we illustrate the power of our method versus BH on one real dataset with 2+1 and 2+4 dimensions. We now show that the power of our method is not determined based upon one real dataset. In addition, we compare the methods on Dataset_3 to study the effects of the cardinality/dimension on their running-time. In this situation, the larger the cardinality, the smaller the density; and the larger the dimension, the larger the density.

As shown in Figure 3, the result summary is that, BH is very slower than our method. For instance, when dimensions are 5+5 and the cardinality is 50,000, the running-time of our method in Stage 1 is 312 seconds with Total of 1,039 seconds, whereas that of BH is 2,432 seconds in Stage 1 with Total of 3,179 seconds. This means that our method decreases the total running-time of BH in Stage 1 by 87% and in Total by 67%. The absolute difference between the running-times is also 2,140 (i.e., 35.7 minutes). Similarly, when the dimensions are 9+9 with 50,000 cardinality, the running-time of our method in Stage 1 is 78,963.60 seconds with Total of 59,994.77 seconds (16 hours), whereas that of BH is 100,551.99 seconds in Stage 1 and Total is 131,863.43 seconds (one day and 12 hours). In this case, the absolute difference between running-time of our method versus BH is 71,869 seconds, representing 1,197 minutes which is almost 20 hours.

Whether the cardinality/dimension increases, our method performs much faster than BH. We emphasize that these results are measured when a single computer with the small number of 8 processors is used, and if we use a computer with more number of processors or we use two computers (or more) in parallel, BH can never be comparable with our method.

7. Conclusion

We have proposed a framework to substantially decrease the running-time to deal with large-scale DEA problems. In the method, we determine the DEA scores in a very short period of time in comparison with the existing approaches. We discuss the pros and cons of the available methodologies and show that the proposed procedure should be the framework of choice to evaluate the performance of large-scale DEA. We have developed our method for the radial VRS model however, in a similar manner, the method can be adjusted for the other DEA models. For future research, pseudocode (4) can be improved to increase the speed of our approach. The combination of our procedure and BH can also be studied for future research.

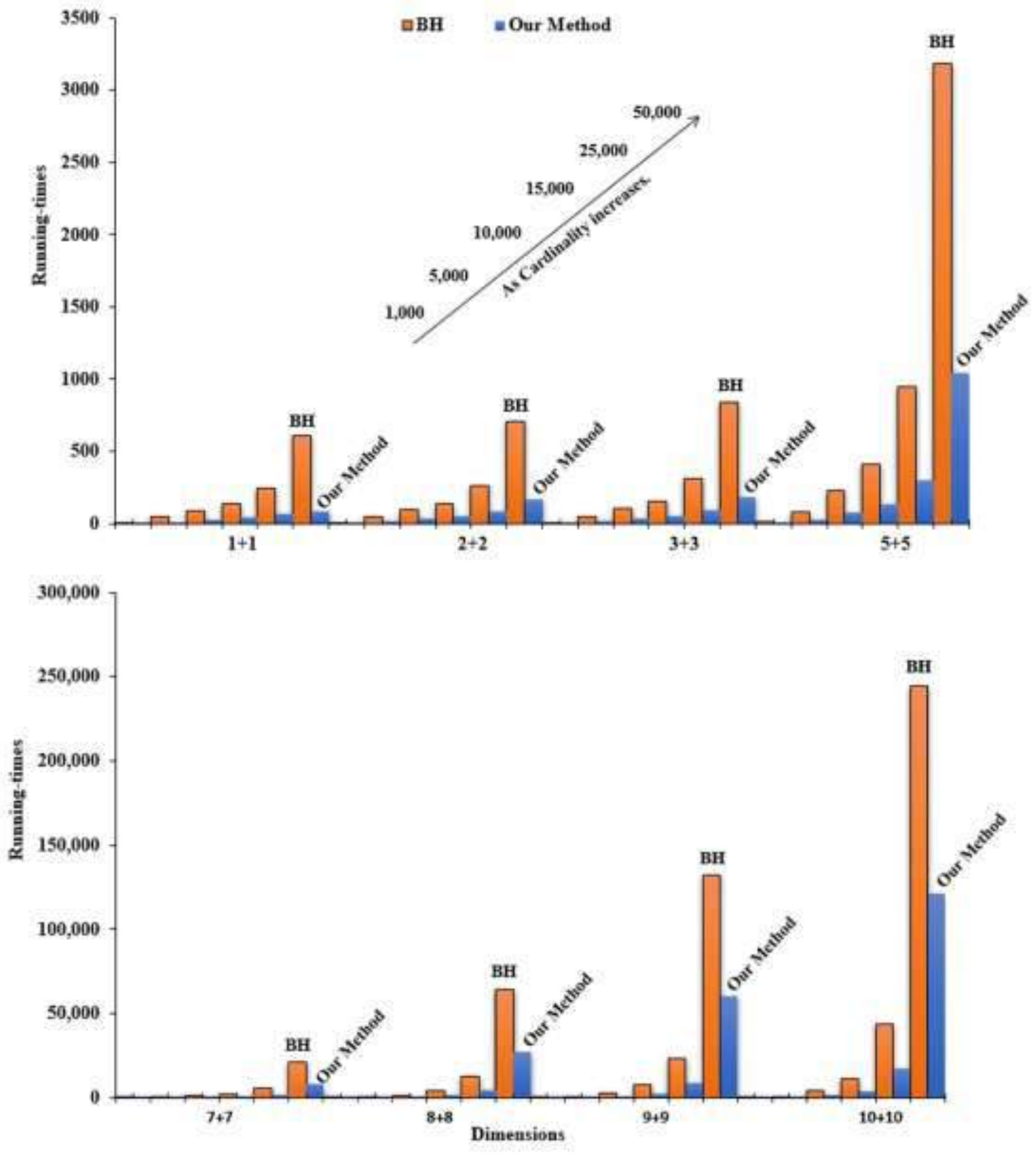

Figure 3: Comparing our method and BH with different dimensions and cardinalities.

Acknowledgments

We appreciate the valuable comments from three anonymous reviewers as they aided in significantly improving the clarity of our paper. This research is supported by Pennsylvania State University (No. 0206020 AC6ET), National Natural Science Funds of China (No. 71828101) and the Czech Science Foundation (GAČR 16-17810S). All support is greatly acknowledged.

References

1. Ali, A. I. (1990). Data envelopment analysis: computational issues. Computers, Environment and Urban Systems, 14(2), 157-165.
2. Ali, A.I. (1993). Streamlined computation for data envelopment analysis. European Journal of Operational Research, 64(1), 61-67.
3. Banker, R.D., Charnes, A., & Cooper, W. W. (1984). Some models for estimating technical and scale inefficiencies in data envelopment analysis. Management Science, 30(9), 1078-1092.
4. Banker, R. D., & Chang, H. (2006). The super-efficiency procedure for outlier identification, not for ranking efficient units. European Journal of Operational Research, 175(2), 1311-1320.
5. Barr, R.S., & Durchholz, M.L. (1997). Parallel and hierarchical decomposition approaches for solving large-scale data envelopment analysis models. Annals of Operations Research, 73, 339-372.
6. Charnes A., Cooper, W.W., & Rhodes, E. (1978). Measuring the inefficiency of decision making units. European Journal of Operational Research 2 (6) 429-444.
7. Chen, W.C., & Cho, W.J. (2009). A procedure for large-scale DEA computations. Computers & Operations Research, 36(6), 1813-1824.
8. Chen, W.C., & Lai, S.Y. (2017). Determining radial efficiency with a large data set by solving small-size linear programs. Annals of Operations Research, 250(1), 147-166.
9. Cooper, W.W., Seiford, L.M., & Tone, K., (2007). Data envelopment analysis: a comprehensive text with models, applications, references and DEA-solver software, 2nd edition. Springer, New York.
10. Dulá, J.H., & 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(2), 352-367.
11. Dulá, J.H., R.V. Helgason, and N. Venugopal. (1998). “An Algorithm for Identifying the Frame of a Pointed Finite Conical Hull.” INFORMS Journal on Computing 10(3).
12. Dulá, J. H., & Thrall, R. M. (2001). A computational framework for accelerating DEA. Journal of Productivity Analysis, 16(1), 63-78.
13. Dulá, J.H. (2008). A computational study of DEA with massive data sets. Computers & Operations Research 35.4: 1191-1203.
14. Dulá, J. H., & López, F. J. (2009). Preprocessing DEA. Computers & Operations Research, 36(4), 1204-1220.
15. Dulá, J.H. (2011). A method for data envelopment analysis. INFORMS Journal on Computing, 23(2), 284-296.

16. Dulá, J. H., & López, F. J. (2013). DEA with streaming data. Omega, 41(1), 41-47.
17. Korhonen, P. J., & Siitari, P. A. (2009). A dimensional decomposition approach to identifying efficient units in large-scale DEA models. Computers & Operations Research, 36(1), 234-244.
18. Seiford, L.M., and Zhu, J. (2005), A response to comments on modeling undesirable factors in efficiency evaluation, European Journal of Operational Research, 161(2), 579-581.
19. Sueyoshi, T., & Chang, Y. L. (1989). Efficient algorithm for additive and multiplicative models in data envelopment analysis. Operations Research Letters, 8(4), 205-213.
20. Zhu, Q., Wu, J., & Song, M. (2018). Efficiency evaluation based on data envelopment analysis in the big data context. Computers & Operations Research, 98, 291-300