Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2025 (43)** : 1–8.
ISSN-0037-8712
doi:10.5269/bspm.62238

A Method to Reduce Calculation Time of Data Envelopment Analysis Problems with Big Data

Roya Hosseinzadeh, Nima Azarmir Shotorbani, Yasser Jafari* and Javad Vakili

ABSTRACT: Data envelopment analysis problems with big data involve solving thousands of linear programs. This article describes a new method that significantly accelerates the solution of these problems. First, the units are divided into smaller categories, then all efficient units in the first category are identified, the next category is solved along with the efficient units from the previous category, and this process continues until the last category. In this way, the number of variables and the constraints in each problem is significantly reduced, and much less time is needed to solve the problems. Assuming the variable returns to scale, an algorithm is designed for big data to calculate run time using the proposed method, which involves a notable reduction in run time compared to existing techniques.

Key Words: Data Envelopment Analysis, Big Data.

Contents

1. Introduction

Data Envelopment Analysis (DEA) was developed by [4] to evaluation the performance of a set of homogeneous decision-making units (DMUs) with similar inputs and outputs. In the data envelopment analysis method, the problem is solved depending on the number of units to evaluate the efficiency of decision-making units. In data envelopment analysis models, the number of decision-making units and inputs and outputs determine the number of constraints and variables. Therefore, we face considerable problems in big data, which increase the computational load. Thus, with the increase in the number of units, the execution time to solve the standard models also increases significantly. There is much research in data envelopment analysis to provide a theoretical solution for reducing computational time using data envelopment analysis models. For example, [9] developed a technique to reduce LP size for a limited set of DMUs. [2] using a HD method for dividing DMUs into smaller groups of approximately equal length, also tried to reduce the computation time to measure the performance of all DMUs by using multiple processors in parallel. [10] propose an approach to identify the most miniature set of DMUs to develop the production possibilities set that leads to DEA results. [12] developed a method for dealing with returns to scale techniques. According to their method, the anti-reduction / anti-incremental returns to scale of efficient DMUs can be extracted from the variable return to scale of efficient DMUs, and from this set, a set of efficient DMUs with a fixed return to scale can be obtained. [7] points to the effect of three parameters when calculating data envelopment analysis models: the number of units involved, the number of inputs and outputs (dimension), and the ratio of efficient units (density). [14] proposed a hierarchical analysis method with lexicographic parametric programming to decrease the burden of DEA calculations when the dimension of the problem is insignificant. [5] also proposed a method to deal with problems with a variable return to scale when the problem dimension is small. Shortly afterward, [8] proposed an

* Corresponding Author

Submitted January 25, 2022. Published May 28, 2025
2010 *Mathematics Subject Classification:* 90C08, 68T09.

effective method, called build-hull (BH), to reduce the run time of all efficient units significantly. [11] used procedural analysis for big data; they provided a theoretical and specific method for big data in dynamic conditions, for when the data is constantly changing. [6] also proposed an algorithm to solve the LP size constraint problem for calculating efficiency when data is available on a large scale. Their LP algorithm can be solved in a fair number of iterations without considering the extra time to run. [15] use two algorithms for splitting large-scale DMUs into two modes: One input, one output, and multiple inputs and multiple outputs. [13] used a five-step method to separate efficient decision-making units and designed a simulation algorithm for estimating run time for big data, assuming a variable return to scale. This paper significantly reduces the DEA calculation time compared to existing methods using the proposed method. To test the proposed algorithm, the traditional method and the proposed method are implemented on MATLAB software and finally, the run time of the algorithms is calculated and compared. This study focuses on the input-oriented model with variable returns to scale(VRS). We use MATLAB 2018a and a computer with Intel (R) Core (TM) i5 CPU, 2.53GHz, 4 GB of memory, and a 64-bit operating system. This study is divided into five sections. The basic definitions of DEA and the Interpretations used are presented in 2. The method is detailed in 3. The running time of the proposed method is compared with the Existing methods in 4. Conclusions are given in 5.

2. Data envelopment analysis

Suppose that we have n DMUs and each of them consumes varying amounts of m different inputs to produce s different outputs. Specifically, consumes the amount of input i to produce the amount of output r. Define the respective input and output vectors. A popular DEA model for evaluating the efficiency value is the input-oriented BCC (Banker- Charles -Cooper) models proposed by [1], shown as the following model 2.1:

$$
\begin{aligned}
E_o^{input} = Max \quad & \frac{\sum_{r=1}^{s} u_r y_{ro} + u_o}{\sum_{i=1}^{m} v_i x_{io}} \\
s.t. \quad & \frac{\sum_{r=1}^{s} u_r y_{rj} + u_o}{\sum_{i=1}^{m} v_i x_{ij}} \leq 1, \quad j = 1, \ldots, n; \\
& u_r \geq 0, \quad r = 1, \ldots, s; \\
& v_i \geq 0, \quad i = 1, \ldots, m; \\
& u_o \quad free \ \ in \ \ sign.
\end{aligned}
\tag{2.1}
$$

In models 2.1, the subscript o denotes the DMU under evaluation. Values u_r and v_i are the input and output multipliers/weights, respectively, each $DMU_o (o = 1, \ldots, n)$ chooses its own optimal weights $u_r^\star$ and $v_i^\star$ to maximize its BCC efficiency. The variable u_o added in the numerator of models 2.1 reflects the variable returns to scale (VRS) assumption. Model 2.1 is non-linear program which can be transformed into the following linear models 2.2 (see, [3]):

$$
\begin{aligned}
E_o^{input} = Max \quad & \sum_{r=1}^{s} u_r y_{ro} + u_o \\
s.t. \quad & \sum_{r=1}^{s} u_r y_{rj} - \sum_{i=1}^{m} v_i x_{ij} + u_o \leq 0, \quad j = 1, \ldots, n; \\
& \sum_{i=1}^{m} v_i x_{io} = 1; \\
& u_r \geq 0, \quad r = 1, \ldots, s;
\end{aligned}
\tag{2.2}
$$

$$v_i \geq 0, \quad i = 1, \ldots, m;$$
$$u_o \quad free \ \ in \ \ sign.$$

The following model 2.3 is the dual of model 2.2:

$$\begin{aligned} E_o^{input} = Min \quad & \theta_o \\ s.t. \quad & \sum_{j=1}^{n} \lambda_j x_{ij} \leq \theta_o x_{io}, \quad i = 1, \ldots, m; \\ & \sum_{j=1}^{n} \lambda_j y_{rj} \geq y_{ro}, \quad r = 1, \ldots, s; \\ & \sum_{j=1}^{n} \lambda_j = 1; \\ & \lambda_j \geq 0, \quad j = 1, \ldots, n; \\ & \theta_o \quad free \ \ in \ \ sign. \end{aligned} \tag{2.3}$$

Definition 2.1 *If in the model 2.3, $\theta^\star = 1$ then DMU_o is efficient, otherwise it is inefficient.*

3. proposed method Algorithm

When calculating BCC efficiency values using models 2.3, we need to solve many different linear programming problems when the number of DMUs is extremely huge. model 2.3 has $n+1$ variables and $s+m+1$ constraints, typically with $n+1 \geq\geq s+m+1$ in the big data environment. Having more variables must increase the burden of calculation. Therefore, we must reduce the number of variables to reduce the burdens. We firstly propose the following Algorithm to address these issues.

Algorithm

Step I: Suppose $k = 0$, denote the set of all the DMUs $J = \{j | j = 1, \ldots, n\}$ Then, we divide J into l equal groups and we show the set of these groups as $J_1, J_2, \ldots, J_l$.

Step II: Suppose $k = k + 1$, calculate the efficiency values for set J_k of DMUs. The efficient DMUs of that group are called E_k. Let $J_{k+1} = E_k \bigcup J_{k+1}$, and so it continues until the last group.

In this case, E_l specifies the set of efficient units. By dividing DMUs into different groups and thus reducing the number of variables in each category, we increase the speed of the calculation process.

4. The running time of the proposed method

We first analyzed the proposed method for data with a small number of decision-making units. The difference in the run time of our method with the traditional method in the data set with a small number of units and small dimensions is insignificant; as we stated, the difference in run time is significant when we are faced with more DMUs and more significant dimensions. We run the problem for 1000 units with different dimensions:

Table 1: Comparison of the run time of the traditional method and the proposed, for 1000 units with 20 categories for problems with different dimensions.

Number of units	Number of categories	Dimensions	Traditional method run time	proposed method run time	Number of efficient units	Decrease in run time	Density
1000	20	$1+1$	28.5553	18.9044	5	33.8%	0.5%
		$3+3$	32.3555	25.9483	110	19.8%	11%
		$5+5$	40.6594	39.4970	328	2.86%	32.8%

We run 1000 DMU with 20 categories (Batch of 50); the difference in the run time of the proposed method with the traditional method is slight, especially for higher dimensions; this difference is subtler. Density also means the number of efficient DMUs divided by the total number of DMUs.

Now we run the problem for 5000 units with a different number of categories and dimensions:

Table 2: Comparison of the run time of the traditional method and the proposed method for 5000 units with different numbers of categories and dimensions.

Number of units	Number of categories	Dimensions	Traditional method run time	proposed method run time	Number of efficient units	Decrease in run time	Density
5000	50	$1+1$	758.6	83.1645	7	89.04%	0.14%
	50	$3+3$	1678.2	193.6647	224	88.46%	4.48%
	25	$3+3$	1678.2	124.5788	224	92.58%	4.48%
	10	$3+3$	1678.2	123.0772	224	92.67%	4.48%
	5	$3+3$	1678.2	166.4384	224	90.08%	4.48%
	50	$5+5$	1839.6	688.1625	953	62.6%	19.06%
	25	$5+5$	1839.6	475.9283	953	74.13%	19.06%
	10	$5+5$	1839.6	293.8556	953	84.03%	19.06%
	5	$5+5$	1839.6	259.8207	953	85.88%	19.06%
	2	$5+5$	1839.6	604.9155	953	67.12%	19.06%

As the number of DMUs increases, the run time of the proposed method decreases significantly compared to the traditional method. For example, for 5000 units with $1+1$ dimensions with 50 categories (Batch of 100), the reduction in the run time of the proposed method compared to the traditional method is 89.04%. Also, for the $3+3$ dimensions, the most significant decrease has occurred in the case of 10 categories (Batch of 500), which is faced with a 92.67% decrease in run time. Moreover, for the $5+5$ dimensions, the most significant reduction has occurred in the case of dividing the data into five categories (Batch of 1000), which is faced with a decrease of 85.88%, which shows the significant effect of the proposed method in reducing the run time. Also, the classification of the units itself effectively reduces the run time, especially in higher dimensions. For example, for 5000 units with $5+5$ dimensions in two different categories, the reduction in run time is obtained as follows: For 50 categories (Batch of 100) 62.6 and for five categories (Batch of 1000) 85.88%, which shows a difference of 27% at the run time only based on the number of each category. (see, figure 1).

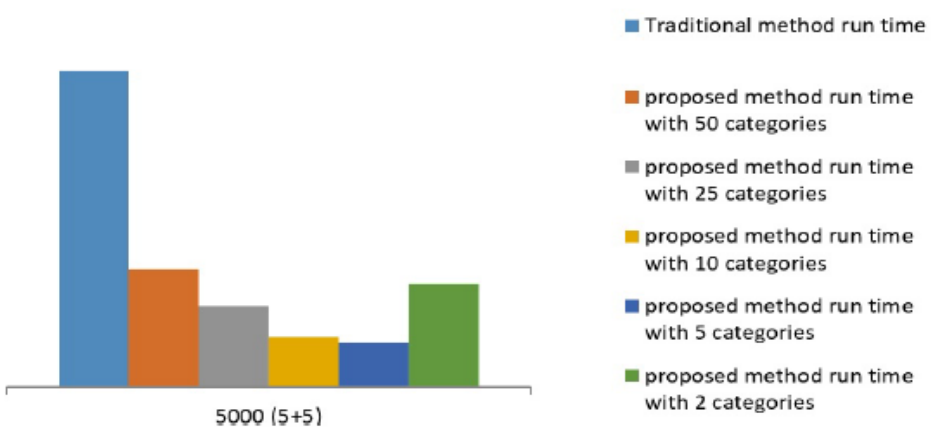

Figure 1: Comparison of the run time of the traditional method and the proposed method for 5000 units with dimensions of $5+5$, with different numbers of categories.

Then We run 10,000 units with different number of categories and dimensions:

Table 3: Comparison of the run time of the traditional method and the proposed method for 10,000 units with different categories and dimensions.

Number of units	Number of categories	Dimensions	Traditional method run time	proposed method run time	Number of efficient units	Decrease in run time	Density
10000	100	1 + 1	5466.5	156.9	9	97.13%	0.09%
	100	3 + 3	13433	657.8	286	95.11%	2.86%
	20	3 + 3	13433	297.6	286	97.79%	2.86%
	10	3 + 3	13433	357.1	286	97.34%	2.86%
	100	5 + 5	14473	4773.9	1323	67.2%	13.23%
	50	5 + 5	14473	2970.1	1323	79.48%	13.23%
	20	5 + 5	14473	1500.5	1323	89.63%	13.23%
	10	5 + 5	14473	1027.0	1323	92.91%	13.23%
	2	5 + 5	14473	3802.5	1323	73.73%	13.23%

For 1000 units with dimensions of 1+1 and 3+3, the reduction of the run time of the proposed method compared to the traditional method with any category is over 95%, which indicates the increase in the effectiveness of the proposed method by increasing the number of data. For the 5+5 dimensions, the most considerable decrease in the 10 categories (Batch of 1000) is equal to 92.91%. Comparing the two tables above, we find that by increasing the number of DMUs to achieve a better result and reducing more time in the run time of the proposed method than the traditional method, there is a need to categorize DMUs into a greater number of categories. Nevertheless, if the number of DMUs is constant, By increasing the inputs and outputs, the fewer categories, the better the result. In fact, increasing the number of DMUs is directly related to the number of categories, but increasing the dimensions is inversely related to the number of categories. Also, the higher the density, the less reduced run time of the proposed method will be compared to the traditional method, and it seems that with increasing density, it is better to divide the units into fewer categories. (see, figure 2).

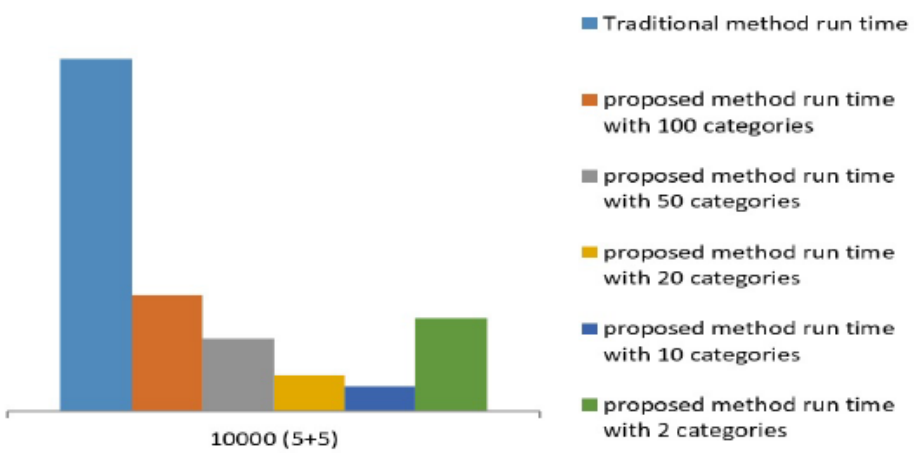

Figure 2: Comparison of the run time of the traditional method and the proposed method for 10,000 units with dimensions of 5 + 5, with different numbers categories.

We run the problem for 15,000 and 20,000 units with a different number of categories and dimensions:

Table 4: Comparison of the run time of the traditional method and the proposed method for 15,000 and 20,000 units with different numbers of categories and dimensions.

Number of units	Number of categories	Dimensions	Traditional method run time	proposed method run time	Number of efficient units	Decrease in run time	Density
15000	150	1 + 1	19028	233.4481	5	98.77%	0.033%
	150	2 + 2	42240	506.4647	87	98.8%	0.58%
	30	2 + 2	42240	324.5341	87	99.23%	0.58%
	15	2 + 2	42240	471.3423	87	98.89%	0.58%
20000	100	1 + 1	47816	319.1830	7	99.33%	0.035%

With the increase in the number of units, we see a substantial reduction in the run time of the proposed method compared to the traditional method, a decrease of over 98%, which is proof of the effectiveness of our proposed method. Now, using the proposed method, we run several examples with a different numbers of units and dimensions; in these examples, we assume that the optimal number of categories is 500 categories:

Table 5: run time of the proposed method for different numbers of units and dimensions.

Number of units	Number of categories	Dimensions	proposed method run time	Number of efficient units	Density
15000	30	1 + 1	249.2	7	0.046%
		2 + 2	324.5	87	0.58%
		3 + 3	505.2	321	2.14%
		4 + 4	1538.4	817	5.44%
		5 + 5	4749.5	1736	11.57%
20000	40	1 + 1	338.5	7	0.035%
		2 + 2	418.9	76	0.38%
		3 + 3	883.8	373	1.86%
		4 + 4	3131.9	985	4.92%
		5 + 5	10236	1854	9.27%
30000	60	1 + 1	570.0	6	0.02%
		2 + 2	666.5	103	0.34%
		3 + 3	1906.9	329	1.09%
		4 + 4	9407.2	1202	4.006%
40000	80	1 + 1	749.5	8	0.02%
		2 + 2	1045.0	74	0.185%
		3 + 3	4573.6	449	1.12%
		4 + 4	24335	1153	2.88%
50000	100	1 + 1	828.5	9	0.018%
		2 + 2	1258.0	85	0.17%
		3 + 3	7110.6	452	0.904%

Comparing the run time of the proposed method with the traditional method, for example, for 20,000 units, the run time of the proposed method for 20,000 units with dimensions of 5 + 5, 10236 seconds, while according to Table 4, the run time of the traditional method for 20,000 units with Dimensions 1 + 1, 47816 seconds, this time is 4.67 times the run time of 20,000 units with dimensions of 5+5, which means 78.59% less run time. Due to the differences in the dimensions of these two samples, this amount of difference at run time is significant. Also, comparing Tables 3 and 5, we find that the run time of the proposed method, for example, with 50,000 units with 3 + 3 dimensions, is less than the run time of

the traditional method with $10,000$ units with $3+3$ dimensions, and this shows the effectiveness of our method. It should be noted that the system RAM of this computer was not able to run the traditional method for the examples in Table 5 and was faced with a shortage of memory. (see, Figure 3).

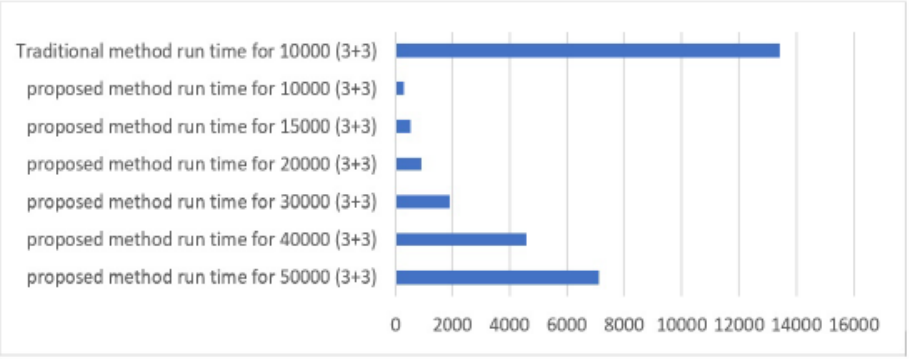

Figure 3: Comparison of the run time of the traditional method and the proposed method for the $3+3$ dimensions, with the number of different units.

5. Conclusion

In traditional Data Envelopment Analysis (DEA) methods, it is important to note that evaluating each decision-making unit requires solving a linear programming problem; in other words, to evaluate all decision-making units, we need to solve as many linear programming problems as there are decision-making units. Furthermore, whenever the number of inputs or outputs or the number of decision-making units is high, the computational complexity for evaluating decision-making units will be high. In this paper, we managed to significantly accelerate the solving of these problems by using a new method. By dividing the units into smaller categories, identifying all efficient units in the first category, solving the next category along with the efficient units of the previous category, and continuing this process until the last category, the number of variables and constraints in each problem is considerably reduced, and much less time is needed to solve the problems. This method can also be adjusted for other DEA models, and finding the optimal number of categories in the proposed method can be studied in future research.

References

1. Banker, R. D., Charnes, A., Cooper, W. W., *Some models for estimating technical and scale inefficiencies in data envelopment analysis*, Manag. Sci. 30(9), 1078-1092 (1984).
2. Barr, R. S., Durchholz, M. L., *Parallel and hierarchical decomposition approaches for solving large-scale data envelopment analysis models*, Ann. Oper. Res. 73, 339-372 (1997).
3. Charnes, A., Cooper, W. W., *Programming with linear fractional functionals*, Nav. Res. Logist. 9(3-4), 181-186 (1962).
4. Charnes, A., Cooper, W. W., Rhodes, E., *Measuring the inefficiency of decision making units*, Eur. J. Oper. Res. 2(6), 429-444, (1978).
5. Chen, W. C., Cho, W. J., *A procedure for large-scale DEA computations*, Comput. Oper. Res. 36(6), 1813-1824 (2009).
6. Chen, W. C., Lai, S.Y. *Determining radial efficiency with a large data set by solving small-size linear programs*, Ann. Oper. Res. 250(1), 147-166 (2017).
7. Dula, J. H., *A computational study of DEA with massive data sets*, Comput. Oper. Res. 35(4), 1191-1203 (2008).
8. Dula, J. H., *An algorithm for data envelopment analysis*, INFORMS. J. Comput. 23(2), 284-296 (2011).
9. Dula, J. H., Helgason, R.V., *A new procedure for identifying the frame of the convex hull of a finite collection of points in multidimensional space*, Eur. J. Oper. Res. 92(2), 352-367 (1996).
10. Dula, J. H., Helgason, R. V., Venugopal, N., *An Algorithm for Identifying the Frame of a Pointed Finite Conical Hull*, INFORMS. J. Comput. 10(3), 323-330 (1998).
11. Dula, J. H., López, F. J., *DEA with streaming data*, Omega. 41(1), 41-47 (2013).

12. Dula, J. H., Thrall, R. M., *A computational framework for accelerating DEA*, J. Product. Anal. 16(1), 63-78 (2001).

13. Khezrimotlagh, D., Zhu, J., Cook, W. D., Toloo, M., *Data envelopment analysis and big data*, Eur. J. Oper. Res. 274(3) 1047-1054 (2018).

14. Korhonen, P. J., Siitari, P. A., *A dimensional decomposition approach to identifying efficient units in large-scale DEA models*, Comput. Oper. Res. 36(1), 234-244 (2009).

15. Zhu, Q. J., Wu, M., Song, M., *Efficiency evaluation based on data envelopment analysis in the big data context*, Comput. Oper. Res. 98, 291-300 (2017).

Roya Hosseinzadeh,
Department of Mathematics,
Tabriz Branch, Islamic Azad University,
Tabriz, Iran.
E-mail address: hosseinzadeh13@gmail.com

and

Nima Azarmir Shotorbani,
Department of Mathematics,
Tabriz Branch, Islamic Azad University,
Tabriz, Iran.
E-mail address: azarmir_nim@yahoo.com

and

Yasser Jafari,
Department of Mathematics,
Shabestar Branch, Islamic Azad University,
Shabestar, Iran.
E-mail address: Yassermath2006@gmail.com

and

Javad Vakili,
Department of Mathematics,Statistics and Computer Science,
University of Tabriz,
Tabriz, Iran.
E-mail address: j.vakili@tabrizu.ac.ir