IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 16, 2021, accepted January 20, 2021, date of publication January 25, 2021, date of current version February 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054006

Heuristic and Backtracking Algorithms for Improving the Performance of Efficiency Analysis Trees

MIRIAM ESTEVE, JESÚS JAVIER RODRÍGUEZ-SALA, JOSÉ JUAN LÓPEZ-ESPÍN, AND JUAN APARICIO

Center of Operations Research (CIO), Miguel Hernández University of Elche (UMH), 03202 Elche, Spain

Corresponding author: Miriam Esteve (miriam.estevec@umh.es)

This work was supported in part by the Spanish Ministry of Science and Innovation and the State Research Agency under Grant PID2019-105952GB-I00/AEI/10.13039/501100011033, and in part by the Spanish Ministry of Science, Innovation and Universities under Grant FPU17/05365.

ABSTRACT In the literature of Economics, Engineering and Operations Research, the estimation of production frontiers is a current hot topic. Many parametric and nonparametric methodologies have been introduced for estimating technical efficiency of a set of units (for example, firms) from the production frontier. However, few of these methodologies are based upon machine learning techniques, despite being a rising field of research. Recently, a bridge has been built between these literatures, machine learning and production theory, through a new technique proposed in Esteve *et al.* (2020), called Efficiency Analysis Trees (EAT). The algorithm developed from EAT, based on the well-known Classification and Regression Trees (CART) machine learning technique, is a greedy technique that uses a particular heuristic for the selection of the next node to be split during the decision tree development process. Nevertheless, as we show in this paper, for different sample sizes and number of variables, the heuristic used by EAT is not capable of obtaining the tree with the minimum mean square error (MSE). For this reason, in this paper, a backtracking technique is implemented to improve the MSE obtained by the EAT algorithm. Additionally, a pair of new algorithms are introduced which combine the heuristic technique used by the standard EAT and the backtracking algorithm to enhance the reduction of the MSE, while decreasing the computation time. Our research is based on some simulated experiments. According to our computational results, the combination of the heuristic and the backtracking algorithm, in particular, that in which the tree growth starts with heuristics and ends with backtracking, has achieved an accuracy similar to that of backtracking and within a reasonable computational time. The contribution of the paper could be of special interest for industrial engineers interested in measuring efficiency and productivity of industrial processes in many sectors, such as energy, agri-food or service industries.

INDEX TERMS Efficiency, CART, backtracking, heuristic algorithm.

I. INTRODUCTION

The measurement of technical efficiency of any type of firm or non-for-profit organization is a topic of great interest in the literature in Economics, Engineering and Operations Research (see, for example, [1], [2] and [3]). Technical efficiency assessment is concerned with determining and comparing the performance of Decision Making Units (DMUs), entities that use several kinds of inputs to produce several types of outputs. An example of a DMU is a farm that uses land, labor and capital (inputs) to produce, for example, fruit (output). The measurement of technical efficiency is an area of interest for a great variety of sectors, from private firms producing goods to service industries, such as hospitals.

The main objective of such evaluation is to analyze the technical efficiency level of a set of DMUs, once we have an input-output observation of each, by comparing their performance with respect to the so-called production possibility set or technology, which is unknown and, in practice, must be estimated from a data sample. A key element in efficiency

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Zakarya.

analysis is the concept of production function. A production function represents the maximum product obtainable from each input combination according to current technical knowledge. From an econometrical viewpoint, a production function is a non-decreasing function that (upper) envelops all the observations (DMUs) in the input-output space, capturing the extreme behavior of the data. These characteristics contrast to those usually assumed by regression techniques, where the function to be estimated relates the response variable to the covariables in terms of the mean as opposed to the maximum. Once an estimation of the production function is obtained, technical efficiency is measured through the deviation of each DMU to the (upper) boundary of the technology, characterized by the production function. Indeed, given an observed input-output vector, the most usual measure of technical efficiency is the ratio of the actual produced output and the maximal producible output, i.e., the value of the production function at the corresponding observed input.

The estimation, in practice, of production functions as enveloping surfaces of the observations satisfying certain properties (such as monotonicity), started with the seminal papers by [4], [5] and [6]. Nowadays, the estimation of production functions is based on very different methodologies, mainly organized into parametric and non-parametric techniques. In the parametric world, the Stochastic Frontier Analysis (SFA) stands out as the most relevant methodology for estimating production functions ([7] and [8]). In this model, the production function must be parametrically described and a stochastic component that describes random shocks affecting the production process is added. Regarding non-parametric techniques, Data Envelopment Analysis (DEA) ([9] and [10]) and Free Disposal Hull (FDH) [11] are highlighted as the most important methodologies. In the case of DEA, the estimator is a piece-wise linear function, while in the case of FDH, the estimator of the production function is a step function. These last two techniques are based on the satisfaction of a set of axioms or postulates in microeconomics: monotonicity, in the case of DEA and FDH, and concavity of the production function, in the case of DEA. Other recent alternative non-parametric techniques for estimating production frontiers are those that apply Kernel-based approaches and local regression techniques. See, for example, [12], where the authors propose a kernel smoothing method that can handle multiple shape constraints (e.g., monotonicity) for multivariate functions, by generalizing [13]. Another interesting contribution is that of [14], who showed how constraint weighted bootstrapping may be applied to impose smoothness conditions on linear estimates. In particular, these authors estimated an input distance function both parametrically and non-parametrically, resorting in the latter to local linear generalized kernel regression. See also [15]. Other interesting approaches are [16], [17] and [18], where the authors introduce how to determine confidence intervals for the efficiency score of each DMU through adapting the bootstrapping methodology by [19] to the context of FDH and DEA. More recently, [20] and [21] have shown that DEA may be reinterpreted as non-parametric least-squares regression subject to shape constraints on the production frontier and sign constraints on residuals.

However, none of the above methodologies for estimating production functions addresses the problem through standard machine learning techniques, as for example Regression Trees, despite being one of the hot topics in the literature on modern data analytics. One recent exception is [22], where Classification and Regression Trees (CART) by [23] were adapted for estimating production functions through step functions, there by competing against the well-known FDH technique. The new approach was named Efficiency Analysis Trees (EAT). Esteve *et al.* [22] showed how EAT clearly outperforms FDH both in terms of mean squared error and bias. See also [24], [25] and [26] to read about the recent interest of the non-parametric community in bridging the gap between FDH and DEA and data science, machine learning and big data.

In particular, the algorithm used by EAT is based upon a heuristic technique for selecting the next node to be split during the growing process of the corresponding decision tree. However, as will be shown in this paper, this heuristic could not always yield the minimum mean squared error among all the possible trees that could be developed. Accordingly, one of the main objectives of this paper is to improve the accuracy of the estimator of the production function generated from EAT by resorting to backtracking techniques ([27] and [28]). In particular, we will combine the idea behind the heuristic approach with the potentiality of backtracking ([29] and [30]) to enhance the quality of the estimator of the production function based on EAT. Additionally, through our approach, we will be able to decrease the computational burden of standard backtracking techniques applied to the decision tree methodology based on EAT, as we will show in our simulated experiences. As far as we are aware, our paper is the first one that contemplates computational aspects of the Efficiency Analysis Trees technique [22]. On the one hand, there are some previous papers devoted to efficiency measurement and related to computational features of the problem (see, for example, [31] and [32]). However, they exploit characteristics linked to the structure of the problem as a Mathematical Programming model. Consequently, these contributions are far removed from the tree structure studied in this paper. On the other hand, there are other previous papers that focus on machine learning techniques, such as decision trees, and efficiency. However, these contributions do not really combine these two facets but rather apply each one in a different stage: in a first stage, technical efficiency is determined and, in a second stage, efficiency is used as a response variable with decision trees or Random Forest, to give details of factors related to technical efficiency (see, for example, [33] and [34]). In contrast, Efficiency Analysis Trees, which is the technique that we work with, methodologically connects these two key topics, i.e., decision trees and efficiency measurement.

In comparison with previous literature, the main contribution of this paper is the improvement of the accuracy of the estimator of the production function generated from EAT by resorting to backtracking and heuristics, while decreasing the computational burden of the technique. From a practical viewpoint, our contribution may be interesting for industrial engineers devoted to improving and monitoring the performance of manufacturing processes within firms. Our algorithms allow production frontiers of the technology associated with industrial processes to be determined and, consequently, a technical efficiency score from the production frontier to be calculated. In this sense, this paper is also in in line with some of the previous papers, namely [35], [36] and [37].

The paper is organized as follows. Section 2 is devoted to briefly introducing the algorithm behind the so-called Efficiency Analysis Trees technique. In section 3, four algorithms developed for estimating production functions are explained. Their computational performance and accuracy are investigated via experimental computations in section 4. Finally, section 5 concludes.

II. EFFICIENCY ANALYSIS TREES

Efficiency Analysis Trees (EAT) [22] is a nonparametric technique to estimate the current level of efficiency of a set of entities based on a tree structure. EAT generates a production frontier which satisfies fundamental postulates of microeconomics, such as free disposability, and shares some similarities with the Free Disposal Hull (FDH) [11] and Classification and Regression Trees (CART) [23]. The operation of EAT is relatively simple and based on a tree structure, offering the possibility of visual representation of rules for data. An EAT tree is built generating binary recursive partitioning of data until no further meaningful division is possible or a stopping rule holds.

Specifically, giving a data sample $[X|Y] \in R^{n\times(m+1)}$, EAT has the target of predicting the response variable y through the predictors x_j, where $j = 1, \ldots, m$. To split the data contained in a node t into two child nodes, t_L and t_R, the algorithm selects a predictor variable x_j, $1 \leq j \leq m$, and a threshold $s_j \in S_j$, where S_j is the set of possible thresholds for the variable j. The splitting of the node t is designed by obtaining the best combination (x_j, S_j) which minimizes

$$R'(t) = R(t_L) + R(t_R) = \frac{1}{n}\sum_{y_i\in t_L}((y_i - y(t_L))^2 + \sum_{y_i\in t_R}((y_i - y(t_R))^2 \quad (1)$$

where n is the sample size and $y(t_L)$ and $y(t_R)$ are the estimates of response variable y in nodes t_L and t_R, respectively, being t_L and t_R the so-called left and right children nodes of t.

Given a node t, the support of t is a region in the predictor space delimited by two points: a^t and b^t. Fig. 1 shows the supports corresponding to nodes t and t'. The EAT algorithm guarantees that the estimated production function is

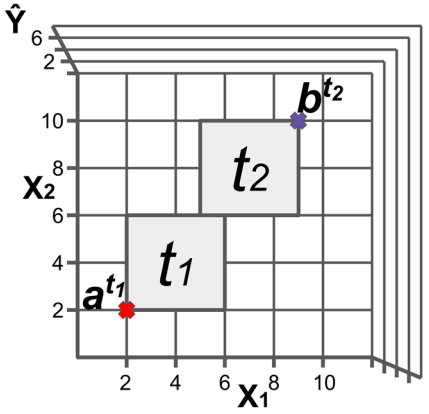


FIGURE 1. Example of the situation of two nodes in which the data associated with t' Pareto-Dominate those of t.

monotonically non-decreasing (the free disposability axiom). To do that, the algorithm resorts to the notion of Pareto-dominance between nodes. In particular, for a node t, the set of nodes in the tree that Pareto-dominate node t is denoted as $I_{\tilde{T}}(t)$. Fig. 1 shows an example where node t' Pareto-dominates node t since $a^{t'} = (2, 2) < b^t = (8, 10)$. So, $y(t)$ must be equal or higher than t' node. In this sense, the predictor function is guaranteed to be non-decreasing. In the EAT algorithm, the predictor of $y(t_R)$ is always the predictor of its parent node. Fig. 2 shows a comparison of the estimated production function derived from FDH and EAT with respect to the theoretical frontier. As can be seen, the FDH estimate is closer to the data and the EAT estimate is closer to the theoretical frontier, which represents the objective function that must be determined. In general, while the FDH approach suffers overfitting, the EAT approach does not.

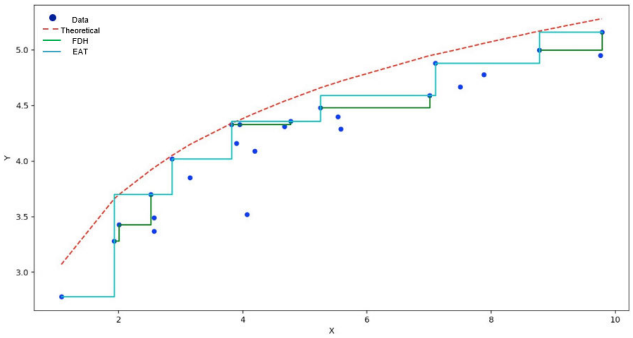


FIGURE 2. Example of Theoretical, EAT and FDH frontiers.

III. ALGORITHM SCHEMES DEVELOPED FOR ESTIMATING PRODUCTION FUNCTIONS

In this section, we analyze the performance of four algorithms linked to the EAT methodology with respect to computational issues and accuracy.

A. A HEURISTIC ALGORITHM BASED ON THE EAT METHOD

The tree determined by EAT is a binary tree structure. Each node t has associated a matrix of data $[X|Y]_t \in R^{n_1\times m}$ with

$1 \leq n_1 \leq n$, which is a submatrix of $[X|Y] \in R^{n\times(m+1)}$. In each of the nodes, the predictor x_j, the set S_j, the mean square error (denoted by $R(t)$), a pair of vectors a^t and b^t vectors and the prediction of $y(t)$ are calculated.

Basically, EAT is an algorithm that obtains an estimation of the production frontier, fulfilling the property of free disposability, through the construction of a binary tree. The estimator of the production function is monotonically non-decreasing, like the step functions that appear in Fig. 2. Therefore, when the two descendants of a node are obtained in the tree construction, the matrix of data associated to this node is divided into two parts and both parts are assigned to the two children, denoted by t_L and t_R, respectively. Those descendants are obtained by minimizing the sum of their mean square errors (MSE) (see Eq. 1). The tree is developed by repeating the splitting process in each node until the ending condition is reached. This condition is related to the minimum number of rows in the data matrix associated to the nodes, which traditionally is at least 5 [23]. In this paper, this value is denoted by *numStop*.

The scheme of the splitting function for a basic split algorithm is shown in Algorithm 1. The prediction of $y(t)$ in the root node is defined as the maximum value of y in the data matrix. When this root node is split, its right node directly takes the parent prediction, i.e. $y(t_R) = y(t)$. In the case of the left node t_L, the set $I_{\tilde{T}}(t_L)$ comes into play. Then, the prediction $y(t_L)$ is set to be the largest value between the maximum value of y in the nodes belonging to $I_{\tilde{T}}(t_L)$ and the maximum value observed in t_L. In this way, we ensure that the predictive function returned by the EAT fulfils the property of free disposability (see Fig. 2).

At this stage, the whole process is repeated until the leaf nodes are reached. The construction of the tree starts from the root node, but as the tree is developed, it is necessary to establish a criterion to select the next t node to be split. The order in which the node is expanded is crucial because the final result depends on it. In the standard EAT algorithm [22], a node t is selected to be split when it is not a leaf node and it has few nodes in $I_{\tilde{T}}(t)$. Usually, this condition is satisfied for the left nodes. So, the splitting of the tree always starts with the left nodes that are not leaves and, once they are all expanded, continues with the right nodes. This procedure describes the particular heuristic followed by the standard EAT algorithm. The splitting process continues until some segmentations are not possible or a predefined stopping rule is satisfied, i.e. *numStop*.

Fig. 3 shows an example of how this heuristic builds the tree. The light grey nodes are the intermediate nodes, except t_0, which is the root node, and the dark grey nodes are the leaf nodes. The numbers within each node indicate the order in which they have been created.

B. A BACKTRACKING ALGORITHM BASED ON THE EAT METHOD

A backtracking algorithm is proposed in this work in order to obtain the tree which gives us the minimum values of

Algorithm 1 Splitting Function Scheme

Input: t, $[X|Y]_t \in R^{n_1\times m}$
Output: t_L, t_R
FOR $j := 1$ **TO** m
  $S_j := \{x_j \in t\}$
  FOR EACH s_j **IN** S_j
    Create t_L and t_R
    $t_L := \{x_j < s_j | x_j \in [X|Y]_{t_L}\}$
    $y(t_L) := \max\{y_i \in t_L\}$
    $t_R := \{x_j \geq s_j | x_j \in [X|Y]_{t_R}\}$
    $y(t_R) := y(t)$
    IF
    $I_{\tilde{T}}(t_L) \neq \emptyset$
      $y(I_{\tilde{T}}(t_L)) := \max\{y(t) \in I_{\tilde{T}}(t_L)\}$
      IF $y(I_{\tilde{T}}(t_L)) > y(t_L)$
        $y(t_L) := y(I_{\tilde{T}}(t_L))$
      END IF
    END IF
    $R'(t) = R(t_L) + R(t_R) =$
    $\frac{1}{n}\sum_{y_i\in t_L}((y_i - y(t_L))^2 + \sum_{y_i\in t_R}((y_i - y(t_R))^2$
  END FOR
  $s_j := \min(R'(t))$
END FOR
$S_j := \min(s_j)$
$a^{t_L}[x_j] := S_j$
$b^{t_R}[x_j] := S_j$

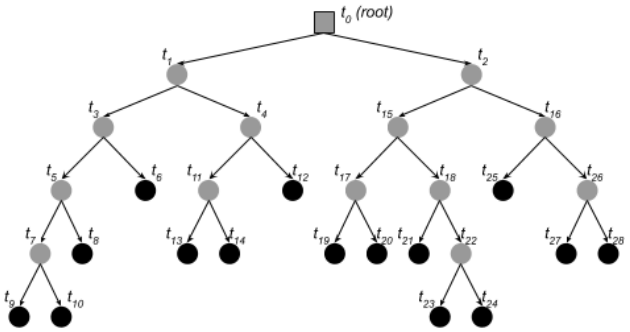


FIGURE 3. Example of tree obtained by the standard EAT algorithm.

the sum of minimum square error (MSE) from the different combinations of (x_j, S_j). The heuristic tree, linked to the standard EAT algorithm, only explores one feasible solution while backtracking explores all of them, obtaining the best one.

The development of the backtracking algorithm in the present work considers the tree decomposition structure in the split function, obtaining this tree structure according to the principle of free disposability [38]. The tree is developed in a similar way to the previous subsection, although the depth of the tree is marked by the number of nodes that can be created by splitting each of its nodes.

C. THE COMBINATION OF HEURISTIC AND BACKTRACKING TECHNIQUES

The heuristic algorithm proposed in A obtains suitable values for MSE. However, these values are worse than those

obtained by the backtracking algorithm B, and at the expense of using a high computational cost. So, new algorithms with both schemes are proposed in this subsection to obtain appropriate values within a reasonable time frame. Indeed, in general, the computational cost consumed by the backtracking algorithm is unfeasible. In this sense, our proposals combine both approaches to obtain better results than using both separately.

Two approaches are studied: the first one combines heuristics in first place and backtracking in second. The tree is constructed by the heuristic until a condition (denoted by *numStopH*) is achieved. When this part of the tree is finished, the backtracking continues to build the tree until an ending condition (denoted by *numStopB*) is reached. In that case, *numStopH* must be higher than *numStopB*, otherwise the tree will be completed, and the backtracking will not be able to continue its expansion. The fulfilment of these conditions for *numStopH* and *numStopB*, are determined by the data matrix associated with the node. Thus, when the ending conditions are reached in the splitting process, the nodes obtained are marked as leaves nodes. For this reason, when the first of them is fulfilled, i.e., *numStopH*, the nodes that are leaves need to be redefined, and then, the state of the tree allows it to continue expanding the nodes.

The second approach is similar to the previous one but involves changing the order of the backtracking and heuristic algorithms. In this case, the tree begins building up through backtracking until *numStopB* is reached and continues expanding through heuristics until the condition linked to *numStopH* is satisfied.

Fig. 4 shows the proposed algorithms: Algorithm A (the heuristic), Algorithm B (the backtracking), Algorithm C (first heuristics and then backtracking), and, finally, Algorithm D (first backtracking and then heuristics). Circled nodes mean that they have been constructed following the heuristic approach. Triangled nodes are all the possible nodes that can be constructed by each of them using the backtracking approach. The numStopH and numStopB indicate the condition that the nodes must fulfill, both those represented by circles and those by triangles, to complete the construction of the tree by means of the heuristic or backtracking approach, respectively.

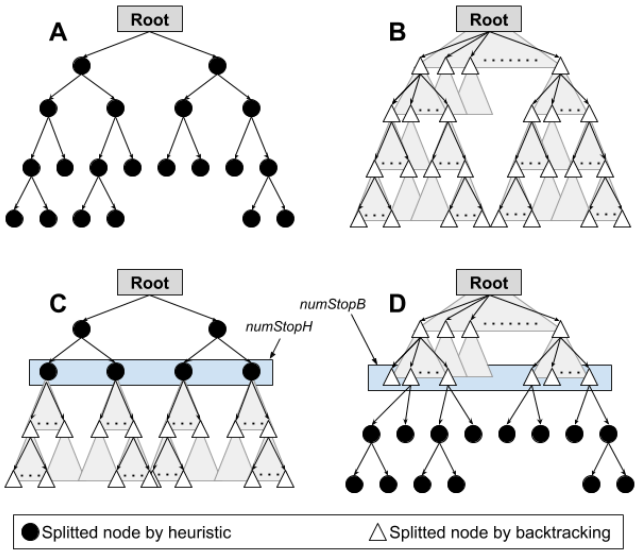


FIGURE 4. The development of trees through diverse approaches.

IV. EXPERIMENTAL RESULTS

The experiments are carried out on a computer with Intel(R) Core (TM) i7-10510U with CPU 1.80 GHz and RAM 16 GB. The code was written in C. Some experiments are carried out to tune certain parameters of the algorithm. In all of them, the data are simulated from a Cobb-Douglas theoretical production function [39]. The input data are randomly sampled from $Uni[0, 1]$ independently for each input and observation. Then, the 'efficient' output level is calculated as $f(x_1, \ldots, x_n) = x_1^{\alpha_1} + \ldots + x_n^{\alpha_n}$, being $\sum_{i=1}^{n} \alpha_i = 0.5$. Finally, a term of 'inefficiency' is randomly determined as $y = x_1^{\alpha_1} + \ldots + x_n^{\alpha_n} \cdot e^{-u}$, where $u \sim Exp\left(\frac{1}{3}\right)$. In this paper the experiments have been carried out using the values shown in Table 1.

The experiments were executed for 100 trials for each combination of the dataset size and number of inputs. The second experiment studies the execution cost when varying the dataset size, denoted by n, and the number of inputs. In these experiments, *numStopH* and *numStopB* take the value 2.

TABLE 1. Theoretical production functions used in the experiments depending on the number of inputs.

Input size	$f(x)$
2	$f(x) = x_1^{0.1} + x_2^{0.4} + e^{-u}$
3	$f(x) = x_1^{0.1} + x_2^{0.1} + x_3^{0.4} + e^{-u}$
4	$f(x) = x_1^{0.1} + x_2^{0.1} + x_3^{0.2} + x_4^{0.2} + e^{-u}$
5	$f(x) = x_1^{0.05} + x_2^{0.1} + x_3^{0.05} + x_4^{0.1} + x_5^{0.2} + e^{-u}$
6	$f(x) = x_1^{0.05} + x_2^{0.1} + x_3^{0.05} + x_4^{0.05} + x_5^{0.1} + x_6^{0.15} + e^{-u}$
7	$f(x) = x_1^{0.05} + x_2^{0.05} + x_3^{0.05} + x_4^{0.05} + x_5^{0.1} + x_6^{0.05} + x_7^{0.15} + e^{-u}$
8	$f(x) = x_1^{0.05} + x_2^{0.05} + x_3^{0.05} + x_4^{0.05} + x_5^{0.1} + x_6^{0.05} + x_7^{0.05} + x_8^{0.1} + e^{-u}$

Table 2 reports the average execution time and the average of the sum of MSE from leaf nodes in the different experiments considered. In particular, the first two columns indicate the sample size and the number of inputs. The rest of the columns show the execution time and the mean of the sum of MSE in the leaf nodes obtained in the four algorithms proposed.

Regarding the results, all methods are affected by the increase in complexity of the problem associated with the number of rows in the datasets, since we detected that the bigger the sample size, the bigger the value of R^2. In addition,

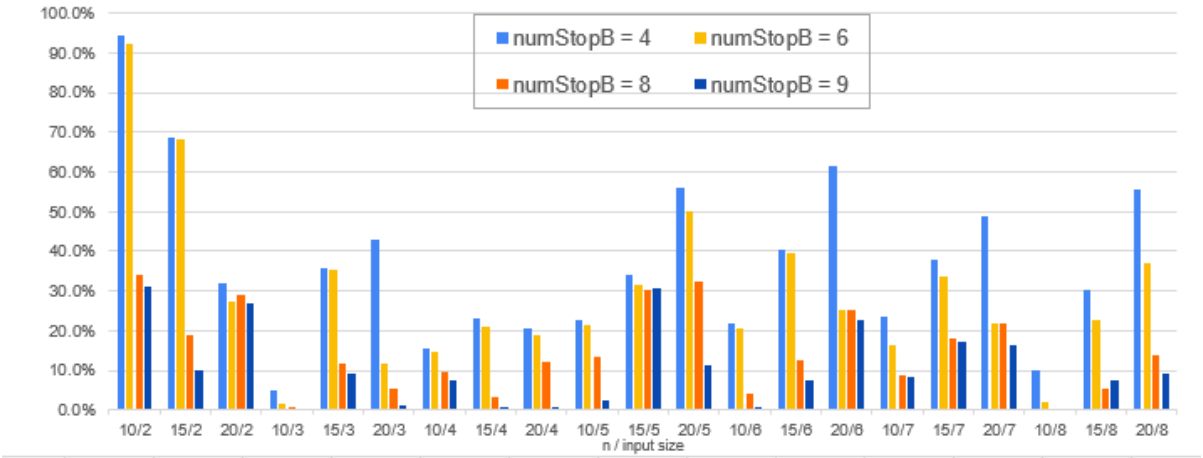


FIGURE 5. Percentage of improvement between Algorithm D and Backtracking (Algorithm B) when the size of the problem, number of inputs and numStopB vary.

the increase in complexity related to the number of inputs in the datasets affects all methods, since we observe that R^2 decreases. The time required to execute the trials is only affected by the increase in the number of rows.

Furthermore, Algorithm A presents the highest value for R^2, Algorithm B the lowest one and the others have values of R^2 between these two. On the one hand, Algorithm C, which has a higher execution cost than Algorithm A, has a value of R^2 that is close to that corresponding to Algorithm A. On the other hand, Algorithm D, which has a lower execution cost than Algorithm B, has a value of R^2 that is close to Algorithm B. Additionally, from Table 2, it seems that Algorithm D is the best algorithm to obtain a satisfactory R^2 in an acceptable execution time.

Algorithm D is the one that obtains the closest values of R^2 with respect to that observed for Algorithm B, which presents the minimum ones, in a feasible execution time.

New experiments were executed using Algorithm D and varying the first ending condition, i.e., *numStopB*. Table 3 reports similar results to those in Table 2, but only with Algorithm D. The first two columns are the problem size and the input size, and the remaining blocks of columns show the applied value of *numStopB*. Moreover, each one of these blocks shows three columns with the execution time, the R^2 value and the percentage of improvement with respect to Algorithm B.

As in Table 2, the values of R^2 increase when the problem size increases and these results are notoriously influenced by the growth of the first ending condition. This effect can be observed in column "*%B*", which represents the improvement of Algorithm D with respect to Algorithm B. The percentage of improvement increases from 30% to 94%. The execution cost is also affected by the increase in the *numStopB*, which is getting closer to the execution times obtained in Algorithm A. So, the best *numStopB* is the one that equally balances the work of backtracking and heuristics. In this way, an acceptable result can be obtained in a feasible execution time.

TABLE 2. Execution cost and precision error when the four algorithms are used for several problem and input size.

n	Input size	Algorithm A		Algorithm B		Algorithm C		Algorithm D	
		Time	R^2	Time	R^2	Time	R^2	Time	R^2
10	2	0.000	0.26	309,204	0.07	32,524	0.26	35.789	0.08
15	2	0.001	1.96	318.757	0.95	17.664	1.62	79.800	1.06
20	2	0.002	4.59	330,005	4.04	19.544	4.33	157.196	4.04
10	3	0.001	0.15	82.775	0.05	6.585	0.15	48.600	0.13
15	3	0.001	1.25	131.355	0.73	10.309	1.25	84.235	1.07
20	3	0.002	0.50	183.638	0.45	17.700	0.50	163.225	0.49
10	4	0.001	0.20	91.175	0.07	10.531	0.21	42.593	0.18
15	4	0.001	0.12	276.829	0.07	12.563	0.12	88.602	0.10
20	4	0.002	0.74	176.645	0.63	18.283	0.74	180.085	0.69
10	5	0.001	0.91	62.846	0.33	9.027	0.91	41.844	0.77
15	5	0.002	0.19	100.198	0.11	16.003	0.19	106.907	0.16
20	5	0.002	0.62	141.652	0.58	18.689	0.62	185.002	0.56
10	6	0.001	0.24	58.681	0.09	8.346	0.23	39.893	0.20
15	6	0.001	0.76	90.44	0.44	12.02	0.76	93.025	0.64
20	6	0.002	0.19	118.405	0.18	16.827	0.19	194.903	0.18
10	7	0.001	0.45	61.117	0.17	8.642	0.45	40.443	0.38
15	7	0.001	0.34	91.154	0.20	12.987	0.33	92.246	0.28
20	7	0.003	0.11	124.467	0.10	17.445	0.11	187.113	0.10
10	8	0.001	0.01	65.492	0.01	9.028	0.01	40.777	0.01
15	8	0.002	0.01	114.744	0.01	16.823	0.01	99.984	0.01
20	8	0.002	0.21	126.868	0.20	19.734	0.21	191.372	0.19

Fig. 5 shows the performance of Algorithm D for different numStopB problems when the problem size changes. The X-axis describes the sample size in relation to the number of inputs (n/input size) and the Y-axis shows the percentage of improvement of R^2 with respect to the backtracking algorithm. The figure reveals that the percentage of improvement is higher when numStopB is equal to 4 and 6.

In the case of real applications, our results could be useful for determining certain parameters, such as numStopB. In particular, the way to determine the most suitable value for the parameter numStopB to apply Algorithm D is directly

TABLE 3. Execution cost and precision error of Algorithm D when the first ending condition varies in 4, 6, 8 and 9.

		numStopB = 4			*numStopB* = 6			*numStopB* = 8			*numStopB* = 9		
n	Input size	Time	R^2	%B	Time	R^2	%B	Time	R^2	%B	Time	R^2	%B
10	2	24.434	0.08	94.6	13.115	0.08	92.2	5.390	0.20	33.9	4.002	0.20	31.3
15	2	57.733	1.26	68.7	35.471	1.27	68.3	9.183	1.77	18.9	5.374	1.85	10.2
20	2	124.601	1.41	31.8	73.625	1.44	27.4	26.403	1.43	28.9	9.264	1.44	27.1
10	3	23.855	0.14	5.2	15.813	0.15	1.5	6.209	0.15	0.8	4.328	0.15	0.4
15	3	68.680	1.07	35.7	46.821	1.07	35.2	8.381	1.19	11.7	5.763	1.21	9.1
20	3	134.692	0.48	43.0	104.128	0.50	11.7	24.974	0.50	5.2	9.634	0.50	1.2
10	4	26.775	0.19	15.5	15.810	0.19	14.6	6.154	0.20	9.7	4.836	0.20	7.6
15	4	72.738	0.11	23.2	65.130	0.11	21.0	15.746	0.12	3.4	5.897	0.12	0.7
20	4	139.782	0.72	20.5	105.389	0.72	18.7	20.193	0.72	12.3	10.006	0.74	0.8
10	5	27.374	0.78	22.9	17.479	0.79	21.3	10.842	0.83	13.3	4.982	0.90	2.5
15	5	81.548	0.16	34.2	50.558	0.16	31.5	12.384	0.16	30.3	6.008	0.16	30.7
20	5	149.646	0.59	56.1	114.125	0.59	50.1	19.193	0.60	32.3	10.235	0.61	11.4
10	6	31.683	0.20	21.9	16.266	0.20	20.5	13.573	0.22	4.2	5.002	0.23	0.7
15	6	68.658	0.63	40.3	51.365	0.63	39.4	28.197	0.72	12.7	6.735	0.73	7.4
20	6	139.837	0.18	61.6	112.087	0.19	25.2	32.645	0.19	25.2	10.935	0.19	22.8
10	7	27.936	0.38	23.6	18.906	0.40	16.4	7.203	0.42	8.9	5.009	0.42	8.2
15	7	73.134	0.29	37.8	58.651	0.29	33.5	17.342	0.31	18.0	6.892	0.31	17.2
20	7	142.477	0.10	48.7	127.912	0.10	21.6	23.201	0.10	21.6	11.281	0.10	16.5
10	8	28.105	0.00	10.0	18.513	0.00	2.0	10.240	0.01	0	5.321	0.01	0
15	8	82.828	0.01	30.2	61.398	0.01	22.6	23.420	0.01	5.3	6.746	0.01	7.5
20	8	143.567	0.20	55.4	122.804	0.20	36.9	27.293	0.21	13.8	10.386	0.21	9.2

dependent on the computational cost and, therefore, on the user's available computing time. For example, following our results, in a problem with sample size 10 and input size 5; if the user has an approximate time of 5 seconds, the most suitable numStopB is 9. If, on the other hand, the user has up to 30 seconds available, then the suitable value for numStopB would be 4, obtaining the lower R^2.

V. CONCLUSION AND FUTURE WORKS

The idea behind the heuristic approach and the potentiality of the backtracking are combined to raise the quality of the estimation of production functions in microeconomics and production engineering based on EAT (Esteve *et al.*, 2020) in a feasible time. To do that, two parameters were set as a condition of achievement for changing the development strategy of the decision tree. The heuristic growth parameter is determined by *numStopH* and the backtracking parameter by *numStopB*. The simulation experiments carried out, prove that backtracking obtains a higher accuracy than heuristics, although with a very high computational cost. On the other hand, in the approaches in which both techniques are joined, it can be seen that the first of them (based on first heuristics and then backtracking) achieves similar results to that of the heuristic algorithm; in the second of them (based first on backtracking and then heuristics), the results are closer to those obtained in backtracking, although in a more reasonable time frame. In this case, it is observed that the percentage of improvement with respect to the accuracy of the backtracking is close to 94% when *numStopB* takes the smallest value, and when it increases, the percentage of improvement decreases from 0.7% to 0%.

According to our computational results, the combination of the heuristic and the backtracking algorithm, in particular, that in which the tree growth starts with heuristics and ends with backtracking, has achieved an accuracy similar to that of backtracking and within a reasonable computational time. In practice, our results could be of special interest for industrial engineers interested in measuring efficiency and productivity of industrial processes in many sectors (manufacturing, energy, agri-food, etc.) since we provide a suitable algorithm for estimating technical efficiency based on a machine learning technique, as decision trees. Additionally, we suggest how to tune some key parameters related to the performance of the algorithm, depending on the problem size and the computing time available.

REFERENCES

[1] H. O. Fried, C. A. K. Lovell, and S. S. Schmidt, *The Measurement of Productive Efficiency*. Oxford, U.K.: Oxford Univ. Press, 2008.

[2] M. Arnaboldi, G. Azzone, and M. Giorgino, *Performance Measurement and Management for Engineers*. Cambridge, MA, USA: Academic, 2014.

[3] C. J. O'Donnell, *Productivity and Efficiency Analysis*. Singapore: Springer, 2018.

[4] M. J. Farrell, "The measurement of productive efficiency," *J. Roy. Stat. Soc. A, Gen.*, vol. 120, no. 3, pp. 253–281, 1957.

[5] D. J. Aigner and S. F. Chu, "On estimating the industry production function," *Amer. Econ. Rev.*, vol. 58, no. 4, pp. 826–839, 1968.

[6] S. N. Afriat, "Efficiency estimation of production functions," *Int. Econ. Rev.*, vol. 13, no. 3, pp. 568–598, Oct. 1972.

[7] D. Aigner, C. A. K. Lovell, and P. Schmidt, "Formulation and estimation of stochastic frontier production function models," *J. Econometrics*, vol. 6, no. 1, pp. 21–37, Jul. 1977.

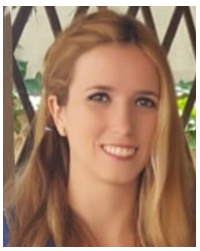
[8] W. Meeusen and J. van Den Broeck, "Efficiency estimation from Cobb-Douglas production functions with composed error," *Int. Econ. Rev.*, vol. 18, no. 2, pp. 435–444, Jun. 1977.

[9] A. Charnes, W. W. Cooper, and E. Rhodes, "Measuring the efficiency of decision making units," *Eur. J. Oper. Res.*, vol. 2, no. 6, pp. 429–444, Nov. 1978.

[10] R. D. Banker, A. Charnes, and W. W. Cooper, "Some models for estimating technical and scale inefficiencies in data envelopment analysis," *Manage. Sci.*, vol. 30, no. 9, pp. 1078–1092, Sep. 1984.

[11] D. Deprins, L. Simar, and H. Tulkens, "Measuring labor-efficiency in post offices," in *Public Goods, Environmental Externalities and Fiscal Competition*, M. Marchand, P. Pestieau, and H. Tulkens, Eds. Amsterdam, The Netherlands, 1984, pp. 243–267.

[12] P. Du, C. F. Parmeter, and J. S. Racine, "Nonparametric kernel regression with multiple predictors and multiple shape constraints," *Statistica Sinica*, vol. 23, no. 3, pp. 1347–1371, Jul. 2013.
[13] P. Hall and L. S. Huang, "Nonparametric kernel regression subject to monotonicity constraints," *Ann. Statist.*, vol. 29, no. 3, pp. 624–647, Jun. 2001.
[14] C. F. Parmeter, K. Sun, D. J. Henderson, and S. C. Kumbhakar, "Estimation and inference under economic restrictions," *J. Productiv. Anal.*, vol. 41, no. 1, pp. 111–129, Feb. 2014.
[15] D. J. Henderson and C. F. Parmeter, "Imposing economic constraints in nonparametric regression: Survey, implementation, and extension," *Adv. Econometrics*, vol. 25, pp. 69–433, Mar. 2009.
[16] L. Simar and P. W. Wilson, "Sensitivity analysis of efficiency scores: How to bootstrap in nonparametric frontier models," *Manage. Sci.*, vol. 44, no. 1, pp. 49–61, Jan. 1998.
[17] L. Simar and P. W. Wilson, "A general methodology for bootstrapping in non-parametric frontier models," *J. Appl. Statist.*, vol. 27, no. 6, pp. 779–802, Aug. 2000.
[18] L. Simar and P. W. Wilson, "Statistical inference in nonparametric frontier models: The state of the art," *J. Productiv. Anal.*, vol. 13, no. 1, pp. 49–78, 2000.
[19] B. Efron, "Bootstrap methods: Another look at the jackknife," *Ann. Statist.*, vol. 7, no. 1, pp. 1–26, Jan. 1979.
[20] T. Kuosmanen and A. L. Johnson, "Data envelopment analysis as nonparametric least-squares regression," *Oper. Res.*, vol. 58, no. 1, pp. 149–160, Feb. 2010.
[21] T. Kuosmanen and A. Johnson, "Modeling joint production of multiple outputs in StoNED: Directional distance function approach," *Eur. J. Oper. Res.*, vol. 262, no. 2, pp. 792–801, Oct. 2017.
[22] M. Esteve, J. Aparicio, A. Rabasa, and J. J. Rodriguez-Sala, "Efficiency analysis trees: A new methodology for estimating production frontiers through decision trees," *Expert Syst. Appl.*, vol. 162, Dec. 2020, Art. no. 113783, doi: 10.1016/j.eswa.2020.113783.
[23] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, 1st ed. New York, NY, USA: Taylor & Francis, 1984.
[24] J. Zhu, "DEA under big data: Data enabled analytics and network data envelopment analysis," *Ann. Oper. Res.*, vol. 23, no. 1, pp. 1–23, Aug. 2019.
[25] V. Charles, J. Aparicio, and J. Zhu, *Data Science and Productivity Analytics*, 1st ed. Heidelberg, Germany: Springer, 2020.
[26] D. Khezrimotlagh, J. Zhu, W. D. Cook, and M. Toloo, "Data envelopment analysis and big data," *Eur. J. Oper. Res.*, vol. 274, no. 3, pp. 1047–1054, 2019.
[27] S. Baase, *Computer Algorithms: Introduction to Design and Analysis*, 3rd ed. New Delhi, India: Pearson, 2009.
[28] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, 1st ed. Annapolis, MD, USA: Computer Science Press, 1978.
[29] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, 1st ed. Boston, MA, USA: Addision Wesley, 1984.
[30] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, Jun. 1972.
[31] J. H. Dulá and R. M. Thrall, "A computational framework for accelerating DEA," *J. Productiv. Anal.*, vol. 16, no. 1, pp. 63–78, 2001.
[32] J. H. Dulá, "A computational study of DEA with massive data sets," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1191–1203, Apr. 2008.
[33] A. Emrouznejad and A. L. Anouze, "Data envelopment analysis with classification and regression tree–a case of banking efficiency," *Expert Syst.*, vol. 27, no. 4, pp. 231–246, Aug. 2010.
[34] S. Rebai, F. Ben Yahia, and H. Essid, "A graphically based machine learning approach to predict secondary schools performance in tunisia," *Socio-Econ. Planning Sci.*, vol. 70, Jun. 2020, Art. no. 100724.
[35] C.-W. Yang and P.-S. Chen, "Applying data envelopment analysis to evaluate the operation performance of Taiwan's TFT-LCD industry after post-global financial crisis: A longitudinal study," *IEEE Access*, vol. 8, pp. 145171–145181, 2020.
[36] M. Olfati, W. Yuan, A. Khan, and S. H. Nasseri, "A new approach to solve fuzzy data envelopment analysis model based on uncertainty," *IEEE Access*, vol. 8, pp. 167300–167307, 2020.
[37] A. A. Shah, D. D. Wu, V. Korotkov, and G. Jabeen, "Do commercial banks benefited from the belt and road initiative? A three-stage DEA-Tobit-NN analysis," *IEEE Access*, vol. 7, pp. 37936–37949, 2019.
[38] R. W. Shephard, *Theory of Cost and Production Function*, 1st ed. Princeton, NJ, USA: Princeton Univ. Press 1970.
[39] P. C. Douglas and C. W. Cobb, "A theory of production," *Amer. Econ. Rev.*, vol. 18, no. 1, pp. 139–165, 1928.



MIRIAM ESTEVE is currently pursuing the Ph.D. degree in statistics, optimization, and applied mathematics with Miguel Hernández University of Elche, Elche, Spain. Her thesis is supported by the Spanish Ministry of Science, Innovation and Universities. She is also a Researcher with the Center of Operations Research. Her research interests include efficiency and productivity analysis, machine learning, and computer programming.

JESÚS JAVIER RODRÍGUEZ-SALA received the Ph.D. degree in computer science from the Miguel Hernández University of Elche, Spain. He is currently a Computer Engineer with the Miguel Hernández University of Elche. He is also a Researcher with the Center of Operations Research. His research interests include data mining and data science. His work focuses on the design and implementation of algorithms for data preprocessing and data analysis. He has applied his research to sectors, such as tourism, health, agronomy, or geography, among others.

JOSÉ JUAN LÓPEZ-ESPÍN received the Ph.D. degree in computer science from University of Murcia, Spain. He specializes in computer science and mathematics. He is currently a Researcher with the Center of Operations Research. His research interest includes computational econometrics, specifically the resolution and finding of simultaneous equation models (SEMs) in high-performance systems and their application in modeling problems in health sciences. He has published more than 26 contributions in JCR and Core A.

JUAN APARICIO received the Ph.D. degree in statistics from the Miguel Hernández University of Elche (UMH), Elche, Spain, in 2007. He is currently a Full Professor in statistics and operations research with the Miguel Hernández University of Elche. He is also the Director of the Center of Operations Research. He is the Co-Chair (with Knox Lovell) of the Santander Chair on Efficiency and Productivity. His research interest includes efficiency and productivity analysis. He has published and co-edited several books focusing on performance evaluation and benchmarking using data envelopment analysis and he has published more than 80 contributions in different international journals.