Research Article

Benchmarking in Data Envelopment Analysis: An Approach Based on Genetic Algorithms and Parallel Programming

Juan Aparicio, Jose J. Lopez-Espin, Raul Martinez-Moreno, and Jesus T. Pastor

Center of Operations Research (CIO), University Miguel Hernandez of Elche, Avenida de la Universidad s/n, 03202 Elche (Alicante), Spain

Correspondence should be addressed to Juan Aparicio; j.aparicio@umh.es

Received 20 October 2013; Accepted 28 December 2013; Published 13 February 2014

Academic Editor: Shangyao Yan

Copyright © 2014 Juan Aparicio et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Data Envelopment Analysis (DEA) is a nonparametric technique to estimate the current level of efficiency of a set of entities. DEA also provides information on how to remove inefficiency through the determination of benchmarking information. This paper is devoted to study DEA models based on closest efficient targets, which are related to the shortest projection to the production frontier and allow inefficient firms to find the easiest way to improve their performance. Usually, these models have been solved by means of unsatisfactory methods since all of them are related in some sense to a combinatorial NP-hard problem. In this paper, the problem is approached by genetic algorithms and parallel programming. In addition, to produce reasonable solutions, a particular metaheuristic is proposed and checked through some numerical instances.

1. Introduction

In the literature, technologies have been estimated using many different approaches over the past 50 years [1]. The two principal methods are stochastic frontiers, which resort to econometric techniques, and Data Envelopment Analysis (DEA), which involves mathematical programming models. In particular, Data Envelopment Analysis (DEA) is a nonparametric technique based on mathematical programming for the evaluation of technical efficiency of a set of decision making units (DMUs) that consumes inputs to produce outputs [2]. In contrast to other efficiency methodologies, for example, stochastic frontiers, DEA provides simultaneously both an efficiency score and benchmarking information through efficient targets. These two pieces of information are usually inseparable in DEA. Indeed, the efficiency score is obtained from the distance between the assessed DMU and a point on the frontier of the technology, which serves as efficient target for the assessed DMU. Information on targets can play a relevant role since they indicate keys for inefficient units to improve their performance.

Traditional DEA measures of technical efficiency yield targets that are associated with the furthest efficient projection to the evaluated DMU [3]. However, several authors

(see [3, 4] to name but a few) argue that the distance to the efficient projection point should be minimized, instead of maximized, in order for the resulting targets to be as similar as possible to the observed inputs and outputs of the evaluated DMU. The cornerstone of this approach is that closer targets suggest directions of improvement for inputs and outputs that lead to the efficiency with less effort than other alternatives.

In general, closest targets are easily attainable and provide the most relevant solution to remove inefficiency from the process of production. The determination of closest targets has attracted an increasing interest of researchers in recent DEA literature [4–7]. In particular, revising the literature we can find that Frei and Harker [8] determined closest targets by minimizing the Euclidean distance to the efficient frontier. Gonzalez and Alvarez [9] proposed to minimize the sum of input contractions required to reach the boundary of the technology. Portela et al. [4] used the software QHull, which allows determining all facets of a polyhedron, with the aim of getting closest targets. More recently, Baek and Lee [10], Pastor and Aparicio [7], Amirteimoori and Kordrostami [5], and Aparicio and Pastor [6] focused their corresponding analysis on a weighted version of the Euclidean distance, while Jahanshahloo et al. [11] provided methods for both obtaining the minimum distance from DMUs to the frontier

and evaluating group performance of DMUs. On the other hand, Briec [12], Briec and Lesourd [13], and Briec and Lemaire [14] obtained the minimum distance to the frontier using Hölder norms, introducing in this way the Hölder distance functions in the literature related to efficiency measurement. Coelli [15] proposed an alternative to the second stage of the process for solving radial models based on closest targets. Cherchye and Van Puyenbroeck [16] maximized the cosine of the angle between the input vectors of the evaluated DMU and its corresponding projection on the boundary. Finally, other related papers are those by Lozano and Villa [17], who determined a sequence of targets to be achieved in successive leaps which finally converge to the efficient frontier, and by Charnes et al. [18] and Takeda and Nishino [19] who use techniques for assessing sensitivity in efficiency classification in DEA based on minimizing distances.

Regarding papers that have studied the computational aspects of DEA models associated with the determination of closest targets, we may cite several references: Aparicio et al. [3] and Jahanshahloo et al. [11, 20–22]. Overall, some of these approaches are based on Mixed Integer Linear Programming or Bilevel Linear Programming while others are derived from algorithms that allow the determination of all the facets of a polyhedron. As we will argue in detail in Section 2, all these approaches present some strong points and weaknesses and, consequently, currently there is no approach accepted as the best solution to the problem.

In view of this discussion, it seems that determining closest targets has been one of the important issues in the DEA literature (see Cook and Seiford [23], for a recent survey on DEA). Nevertheless, from a computational point of view, the problem, that is, the determination of closest targets, is difficult enough and this fact justifies the effort to apply new methods in order to overcome it.

In this paper, we use several genetic algorithms with the aim of determining closest targets in DEA. We apply this type of methodology to solve the aforementioned problem. In particular, we resort to the approach by Aparicio et al. [3], which is based on Mixed Integer Linear Programming, and try to find feasible solutions of the model that these authors introduced by means of genetic algorithms. The difficulty of the problem requires us to study the genetic algorithm by parts, incorporating restrictions while the results are analyzed.

The remainder of the paper is organized as follows. In Section 2, a brief introduction of the main notions associated with Data Envelopment Analysis is presented. Additionally, in this section the existing approaches for determining closest targets are outlined. In Section 3, the genetic algorithm is developed. After that, a random search is used to improve this algorithm, to obtain a hybrid metaheuristic. The idea is to use a random search method in some parts of the genetic algorithm to explore the space of feasible solutions better. To reduce the execution time, a parallel algorithm in shared memory is developed and studied. In Section 5, the results of some experiments are summarized. Finally, Section 6 concludes the paper.

FIGURE 1: Stochastic frontiers versus DEA.

2. Data Envelopment Analysis and Closest Targets

Recent years have seen a great variety of applications of DEA for the use in evaluating the efficiency of many different types of entities [2]. DEA models are based on mathematical programming. While other methods, as stochastic frontiers, need the specification of a functional form for the production function (such as the Cobb-Douglas form), DEA is a nonparametric technique that estimates a piecewise-linear convex technology without this requirement, constructed such that no observation of the sample of data lies above it (refer to Figure 1). In Figure 1, we represent a sample of several firms (DMUs) that use one input (x) to produce one output (y). Each firm is represented by a point in the figure. We also show both the Cobb-Douglas function estimated from the data, if the analyst assumes such functional form, and the piecewise-linear convex estimation of the frontier of the technology resorting to DEA techniques.

DEA involves the use of Linear Programming to construct the nonparametric piecewise surface over the data. Technical efficiency measures associated with the performance of each DMU are then calculated relative to this surface, as a distance to it. Although Farrell [24] was the first to introduce these ideas in the literature, the method did not receive wide attention until the paper by Charnes et al. [25], in which the term Data Envelopment Analysis was coined. Since then there have been a large number of papers which have extended, adapted, and applied this methodology in the fields of economics, operations research, and management science.

Before continuing, we need to define some notations. Assume that there are data on m inputs and s outputs for each of n DMUs. For the j th DMU these are represented by $x_{ij} \geq 0$, $i = 1, \dots, m$, and $y_{rj} \geq 0$, $r = 1, \dots, s$, respectively.

There are a lot of DEA technical efficiency measures in the literature. The basic DEA models are the CCR [25] and the BCC [26]. Both models are based on radial projections to the production frontier. However, many other approaches

give freedom to the projection so that the final efficient targets do not conserve the mix of inputs and outputs. In particular the enhanced Russel Graph measure [27] or slacks-based measure [28] may be calculated for DMU k , $k = 1, \dots, n$, as follows:

$$\begin{aligned}
 \min \quad & \frac{1 - (1/m) \sum_{i=1}^m (s_{ik}^- / x_{ik})}{1 + (1/s) \sum_{r=1}^s (s_{rk}^+ / y_{rk})} \\
 \text{s.t.} \quad & \sum_{j=1}^n \lambda_{jk} x_{ij} = x_{ik} - s_{ik}^-, \quad \forall i, \\
 & \sum_{j=1}^n \lambda_{jk} y_{rj} = y_{rk} + s_{rk}^+, \quad \forall r, \\
 & \lambda_{jk}, s_{ik}^-, s_{rk}^+ \geq 0, \quad \forall j, i, r.
 \end{aligned} \tag{1}$$

Equation (1) is a linear fractional program that can be easily transformed into a linear program by a change of variables as described in Pastor et al. [27]. Specifically, described equation (1) is equivalent to the following linear program:

$$\begin{aligned}
 \min \quad & \beta_k - \frac{1}{m} \sum_{i=1}^m \frac{t_{ik}^-}{x_{ik}} \\
 \text{s.t.} \quad & \beta_k + \frac{1}{s} \sum_{r=1}^s \frac{t_{rk}^+}{y_{rk}} = 1, \\
 & -\beta_k x_{ik} + \sum_{j=1}^n \alpha_{jk} x_{ij} + t_{ik}^- = 0, \quad \forall i, \\
 & -\beta_k y_{rk} + \sum_{j=1}^n \alpha_{jk} y_{rj} - t_{rk}^+ = 0, \quad \forall r, \\
 & \beta_k, \alpha_{jk}, t_{ik}^-, t_{rk}^+ \geq 0, \quad \forall j, i, r.
 \end{aligned} \tag{2}$$

The Enhanced Russell Graph measure, defined as the optimal value of the above model, satisfies several interesting properties from a mathematical and economic point of view. However, it presents a weakness. In particular, (1), or equivalently (2), yields efficient targets, points onto the piecewise frontier, which are far away from the inputs and outputs of the evaluated DMU k . To illustrate this idea, note that the targets for this model are defined as $\sum_{j=1}^n \lambda_{jk} x_{ij} = (x_{ik} - s_{ik}^-)$ for inputs and $\sum_{j=1}^n \lambda_{jk} y_{rj} = (y_{rk} + s_{rk}^+)$ for outputs and that the slacks, s_{ik}^- and s_{rk}^+ , appear in the objective function. Consequently, and since we are minimizing, the model seeks slacks as large as possible and, therefore, (1) yields the furthest targets for DMU k with original inputs and outputs $(x_{1k}, \dots, x_{mk}, y_{1k}, \dots, y_{sk})$. In order to determine closest targets instead of furthest targets, it seems enough to change “minimizing” by “maximizing.” However, it is not true. In this case, we could show that the targets generated by the model would be not technically efficient but inefficient [3]. And, therefore, they could not serve as benchmark for the assessed DMU.

This problem was the main reason behind the introduction of different approaches in the literature to determine closest targets. On the one hand, a group of the researchers [20, 21] focus their work on finding all the faces of the polyhedron that defines a technology estimated by DEA. For example, in Figure 1, we show five of these faces. The computing time of these algorithms increases significantly as the problem size grows $(n + m + s)$ since this issue is closely related to a combinatorial NP-hard problem. On the other hand, a second group proposes to determine closest targets through mathematical programming [3, 22]. In particular, Aparicio et al. [3] introduced the following Mixed Integer Linear Program to overcome the problem for DMU k :

$$\begin{aligned}
 \max \quad & \beta_k - \frac{1}{m} \sum_{i=1}^m \frac{t_{ik}^-}{x_{ik}} \\
 \text{s.t.} \quad & \beta_k + \frac{1}{s} \sum_{r=1}^s \frac{t_{rk}^+}{y_{rk}} = 1, \quad (\text{c.1}) \\
 & -\beta_k x_{ik} + \sum_{j=1}^n \alpha_{jk} x_{ij} + t_{ik}^- = 0, \quad \forall i, \quad (\text{c.2}) \\
 & -\beta_k y_{rk} + \sum_{j=1}^n \alpha_{jk} y_{rj} - t_{rk}^+ = 0, \quad \forall r, \quad (\text{c.3}) \\
 & -\sum_{i=1}^m v_{ik} x_{ij} + \sum_{r=1}^s \mu_{rk} y_{rj} + d_{jk} = 0, \quad \forall j, \quad (\text{c.4}) \\
 & v_{ik} \geq 1, \quad \forall i, \quad (\text{c.5}) \\
 & \mu_{rk} \geq 1, \quad \forall r, \quad (\text{c.6}) \\
 & d_{jk} \leq M b_{jk}, \quad \forall j, \quad (\text{c.7}) \\
 & \alpha_{jk} \leq M (1 - b_{jk}), \quad \forall j, \quad (\text{c.8}) \\
 & b_{jk} = 0, 1, \quad (\text{c.9}) \\
 & \beta_k \geq 0, \quad (\text{c.10}) \\
 & t_{ik}^- \geq 0, \quad \forall i, \quad (\text{c.11}) \\
 & t_{rk}^+ \geq 0, \quad \forall r, \quad (\text{c.12}) \\
 & d_{jk} \geq 0, \quad \forall j, \quad (\text{c.13}) \\
 & \alpha_{jk} \geq 0, \quad \forall j. \quad (\text{c.14})
 \end{aligned} \tag{3}$$

Regarding (3), some comments are in order. First, note that (c.1)–(c.3) are the same constraints as those used in (2) and it implies that we are considering the same set of feasible points. However, note also that the objective function is maximized in this case instead of minimized. Second, with (c.4)–(c.6) we are considering supporting hyperplanes such that all the points of the estimated technology (a polyhedron) lie on or below these hyperplanes. Finally, (c.7) and (c.8) are the key conditions that connect the two previous sets of constraints. Specifically, they avoid that the targets correspond to interior points of the estimated technology. Finally, (c.9) defines b_{jk} as a binary variable and (c.10)–(c.14) are the usual nonnegative constraints.

```

(1) Initialize (S)
(2) while NotEndConditions (S) do
(3)   Evaluate (S)
(4)   SS1 = Select the best ranking of S
(5)   SS2 = Crossover and Mutation (SS1)
(6)   S = IncludeSolutions (SS2)
(7) end while

```

ALGORITHM 1: Scheme of a genetic algorithm.

A weakness of the approach in Aparicio et al. [3] is that it uses a “big M ” to model the key constraints (c.7) and (c.8). Specifically, it allows to link d_{jk} to α_{jk} by means of the binary variable b_{jk} . However, the value of M may be calculated if and only if we previously calculate all the faces that define the technology. Accordingly, this alternative is again associated with a combinatorial NP-hard problem. In the same manner, Jahanshahloo et al. [22] resorted to Lineal Bilevel Programming and a big M to determine closest targets and, therefore, their approach presents a similar drawback.

$$\beta_k \quad b_{1k} \quad \cdots \quad b_{nk} \quad t_{1k}^- \quad \cdots \quad t_{mk}^- \quad t_{1k}^+ \quad \cdots \quad t_{sk}^+ \quad \alpha_{1k} \quad \cdots \quad \alpha_{nk}. \quad (4)$$

Algorithm 1 shows the scheme of a genetic algorithm. The function in the scheme and the values of some parameters for a basic genetic algorithm are shown in Algorithm 1.

3.1. Defining a Valid Chromosome. A valid chromosome has to satisfy at least the following constraints in (3): (c.2), (c.3), (c.8), (c.9), (c.10), (c.11), (c.12), and (c.14). Four methods to generate the initial population of chromosomes have been tested. Methods 1, 2, and 4 are different and Method 3 is an extension of Method 2.

Method 1. The parameters of the chromosomes are generated randomly.

Method 2. The process starts obtaining a random β_k and a set of α_{jk} values using Algorithm 2. Next the b_{jk} values are generated considering α_{jk} in order to satisfy (c.8). The values t_{rk}^+ and t_{ik}^- are deduced from inputs and outputs matrices and the previously generated β_k and α_{jk} using (c.2) and (c.3). In case of obtaining a valid solution the algorithm ends, if not, Algorithm 3 is used.

Algorithm 3 decreases and increases β_k in order to obtain the minimal β_k that satisfy (c.11). Two parameters are used: a factor number q and the maximal number of iterations of the process, which also determines the increasing or decreasing value of β_k . To increase β_k , a similar algorithm is used working in a similar way to decrease β_k , but doing the add operation in the first inner loop and using the constraint (c.12) as a condition instead of (c.11) for both inner loops.

In view of the preceding discussion, from a computational point of view, the determination of closest targets in DEA has not yet been satisfactorily solved and, consequently, it justifies the effort to apply new methods in order to overcome the problem. In this paper, we apply genetic algorithms to solve (3). In particular, and since the problem is really difficult, we will find valid solutions of (3) in the following sections.

3. Genetic Algorithms for Determining Closest Targets

Genetic algorithms [29] are used here to obtain a satisfactory solution of (3). A population of chromosomes representing particular valid solutions of (2) is explored. Each chromosome represents a candidate to be the best model and it is composed by $\beta_k, \alpha_{jk}, t_{ik}^-, t_{rk}^+ \in \mathbb{R}^+$ and $b_{jk} \in \{0, 1\}$ with $i = 1, \dots, m, j = 1, \dots, n, r = 1, \dots, s$. An evaluation of each chromosome is calculated by using the objective function of (3). Consider

Method 3. This method is an extension of Method 2 through adding a third tweak process for the α_{jk} (Algorithm 4).

Method 4. Method 4 consists in a hyperheuristic (metaheuristic that operates over some other metaheuristic) [30]. A genetic algorithm has been used to produce sets of α_{jk} and β_k for the initial population, which satisfy (c.2), (c.3), (c.11), (c.12), and (c.14). The evaluation function in the hyperheuristic is sum of the negative values of t_{rk}^+ and t_{ik}^- , with the chromosome with the higher punctuation being a better candidate. Moreover β_k and α_{jk} are considered, penalizing values close to 1.

3.2. Select the Best Ranking, Crossover, and Mutation. In each generation a proportion of the existing population is selected to breed a new generation. A comparison of the evaluations of all the chromosomes in the population is made in each generation, and only part of them (those which are in the best ranking) will survive. The number of chromosomes which survive in each population (called *SurvSize*) is preset. For each two new solutions to be produced (“son” and “daughter”), a pair of “parent” (“father” and “mother”) chromosomes is selected from the set of chromosomes.

Due to the number of constraints to satisfy, a traditional crossover method will produce an offspring with a higher rate of nonvalid chromosomes. With the aim of avoiding this, a crossover with different “levels” is introduced in Algorithm 5. The defined levels are 1 for β_k , 2 for α_{jk} , 3 for t_{ik}^- , and 4 for t_{rk}^+ . A level is chosen randomly, and values in that level are crossed and those in lower levels are deduced to satisfy (3).

Require: $X \in \mathbb{R}^{+m,n}, Y \in \mathbb{R}^{+s,n}$, DMU k .
Ensure: $\alpha_{1k}, \dots, \alpha_{nk} \in \mathbb{R}^+, \forall \alpha_{jk} \geq 0$.
 $V^x, V^y \in \mathbb{R}^{+n}$ such as $V_j^x = \frac{1}{m} \sum_{i=1}^m X_{i,j}$ and $V_j^y = \frac{1}{s} \sum_{i=1}^s Y_{i,j}$
Sort V^x and V^y in decreasing.
for $j := 1, \dots, n$ **do**
 if $V_j^x < \lfloor n/2 \rfloor$ **and** $V_j^y \geq \lfloor n/2 \rfloor$ **then**
 $\alpha_{jk} \leftarrow 0$
 else if $V_j^x \geq \lfloor n/2 \rfloor$ **and** $V_j^y < \lfloor n/2 \rfloor$ **then**
 $\alpha_{jk} \leftarrow$ Generate $0.5 \leq \alpha_{jk} \leq 1$ randomly.
 else if $(V_j^x \geq \text{BoundPoint and } V_j^y \geq \text{BoundPoint})$ **or** $(V_j^x < \text{BoundPoint and } V_j^y < \text{BoundPoint})$ **then**
 $\alpha_{jk} \leftarrow$ Generate $0 \leq \alpha_{jk} \leq 0.25$ randomly.
 end if
end for
Find the minimum α_{jk} and modify its value in order to satisfy (3)

ALGORITHM 2: Generate alpha.

Require: $\beta_k, q \in \mathbb{Z}, \text{MaxIter} \in \mathbb{Z}$.
Ensure: A minimal β_k for the chromosome that satisfy constraint 11: $\forall t_{ik}^- \geq 0$.
 $d \leftarrow 0$
while $d \leq \text{MaxIter}$ **do**
 while $\forall t_{ik}^- \geq 0$ **do**
 {Decrease β_k while still satisfies (c.11)}
 $\beta_k \leftarrow \beta_k - \frac{q}{d}$
 Generate t_{ik}^- using expression (2).
 end while
 {At this point (c.11) is not satisfied because of the decrease of β_k . Increase β_k value until it is satisfied again}
 repeat
 $\beta_k \leftarrow \beta_k + \frac{q}{d}$
 Generate t_{ik}^- using (3).
 until $\forall t_{ik}^- \geq 0$
 $d \leftarrow d + 1$
end while

ALGORITHM 3: Decrease β_k .

In each iteration a chromosome is randomly chosen to be mutated. The procedure is similar to crossover, considering the dependences between variables and defining different levels of mutation.

4. Parallel Algorithms

The algorithms proposed for GA are very time-consuming when the problem size and *PopSize* increase. Thus, we propose parallel algorithms which aim to reduce the execution time. The parallelization is carried out on a shared-memory model but a design for distributed memory could be similarly generated. Thus, we assume a shared-memory computer with p processors. The parallel programming environment that allows the use of this computer is OpenMP [31]. There are multithread implementations of the basic linear algebra library BLAS [32], which can be used by higher level libraries

(LAPACK [33]) to take advantage of this architecture by using multithreading techniques. For example, the Intel MKL library and the ATLAS [34] package have efficient multithread versions of BLAS.

The costliest parts of the genetic algorithm are *Evaluate*, *Crossover*, and *Mutate* and have been paralleled simply by assigning some chromosomes to each processor. The population is initialized only once, but it has a high cost and is also paralleled as above.

5. Experimental Results

Experiments have been carried out in a computer with Itanium-2 dual-core. The code has been written in C. We used the 11.1 version of Intel compiler and the Intel MKL library, version 10.2.2.025, which contains a multithread implementation of LAPACK.

Require: $X \in \mathbb{R}^{+m,n}$, $Y \in \mathbb{R}^{+s,n}$, $q \in \mathbb{R}$, DMU k , $MaxIter \in \mathbb{Z}$
Ensure: Chromosome c .

```

repeat
  Generate  $0 \leq \beta_k \leq 1$  randomly.
  Generate  $\alpha_{1k}, \dots, \alpha_{nk}$  using Algorithm 2 and  $t_{1k}^-, \dots, t_{mk}^-$  and  $t_{1k}^+, \dots, t_{sk}^+$  using expression (2).
  while  $\exists t_{ik}^- \leq 0$  and  $\exists t_{rk}^+ \leq 0$  and number of  $\alpha_{jk} \neq 0 > 2$  do
    if  $\exists t_{ik}^- \leq 0$  and  $\forall t_{rk}^+ \geq 0$  then
      Increase  $\beta_k$ 
    end if
    if  $\forall t_{ik}^- \geq 0$  and  $\exists t_{rk}^+ \leq 0$  then
      Decrease  $\beta_k$ 
    end if
    if  $\exists t_{ik}^- \leq 0$  or  $\exists t_{rk}^+ \leq 0$  then
      Choose  $\alpha_{jk}$  no null randomly and make it equal to zero, decreasing the rest  $\alpha_{jk}$ 
      in  $p$  and find the minimum  $\alpha_{jk}$  and modify its value in order to satisfy (3)
    end if
  end while
until A valid chromosome is obtained or  $Iterations \geq MaxIter$  or the number of  $\alpha_{jk}$ 
no null is lower than  $n - 2$ 

```

ALGORITHM 4: Generate a chromosome (Method 3).

Require: Chromosome c_1 , Chromosome c_2 .
Ensure: Chromosome c_3 , Chromosome c_4

```

Generate  $0 \leq \gamma \leq 1$  randomly.
Generate  $0 \leq \phi \leq 4$  randomly.
if  $\phi = 0$  then
  {Crossing  $\beta_k$ }
   $\beta_k^3 \leftarrow \beta_k^1 * \gamma + (1 - \gamma) * \beta_k^2$ 
   $\beta_k^4 \leftarrow \beta_k^2 * \gamma + (1 - \gamma) * \beta_k^1$ 
else if  $\phi \leq 1$  then
  {Crossing  $\alpha_{jk}$ }
  for  $j := 1, \dots, n$  do
     $\alpha_{jk}^3 \leftarrow \alpha_{jk}^1 * \gamma + (1 - \gamma) * \alpha_{jk}^2$ 
     $\alpha_{jk}^4 \leftarrow \alpha_{jk}^2 * \gamma + (1 - \gamma) * \alpha_{jk}^1$ 
  end for
else if  $\phi \leq 2$  then
  {Crossing  $t_{ik}^-$ }
  for  $i := 1, \dots, m$  do
     $t_{ik}^{-,3} \leftarrow t_{ik}^{-,1} * \gamma + (1 - \gamma) * t_{ik}^{-,2}$ 
     $t_{ik}^{-,4} \leftarrow t_{ik}^{-,2} * \gamma + (1 - \gamma) * t_{ik}^{-,1}$ 
  end for
else if  $\phi \leq 3$  then
  {Crossing  $t_{rk}^+$ }
  for  $r := 1, \dots, s$  do
     $t_{rk}^{+,3} \leftarrow t_{rk}^{+,1} * \gamma + (1 - \gamma) * t_{rk}^{+,2}$ 
     $t_{rk}^{+,4} \leftarrow t_{rk}^{+,2} * \gamma + (1 - \gamma) * t_{rk}^{+,1}$ 
  end for
end for
Deduce  $\alpha_{jk}$ ,  $b_{jk}$ ,  $t_{ik}^-$  and  $t_{rk}^+$  for  $c_1$  and  $c_2$  in order to satisfy (3)

```

ALGORITHM 5: Crossover.

TABLE 1: Execution cost and % of valid chromosomes when the four methods of initiation are used, all when varying the problem size.

m	Size		Method 1		Method 2		Method 3		Method 4	
	n	s	Time	% val.	Time	% val.	Time	% val.	Time	% val.
2	15	1	0.003 _{0.004}	0.75 _{1.71}	0.008 _{0.005}	50.83 _{41.92}	26.423 _{51.440}	82.08 _{38.58}	17.244 _{3.566}	10.58 _{3.12}
3	25	2	0.004 _{0.004}	0.00 _{0.00}	0.010 _{0.006}	33.55 _{38.24}	6.722 _{16.025}	90.05 _{30.46}	21.283 _{0.801}	0.80 _{0.83}
4	30	2	0.004 _{0.005}	0.00 _{0.00}	0.022 _{0.008}	26.87 _{29.09}	0.223 _{0.584}	100.00 _{0.00}	29.521 _{10.540}	1.57 _{1.20}
5	40	3	0.004 _{0.003}	0.00 _{0.00}	0.019 _{0.003}	13.90 _{23.90}	13.125 _{20.640}	73.90 _{43.40}	18.187 _{1.209}	0.00 _{0.00}
6	60	4	0.006 _{0.000}	0.00 _{0.00}	0.032 _{0.001}	0.03 _{0.16}	2.066 _{1.132}	34.74 _{44.07}	103.698 _{0.185}	0.18 _{0.46}
10	100	10	0.011 _{0.000}	0.00 _{0.00}	0.092 _{0.002}	0.00 _{0.00}	8.426 _{3.235}	32.65 _{41.44}	302.316 _{0.713}	0.00 _{0.00}

FIGURE 2: A comparison between the percentage of valid chromosomes obtained with each generation method.

Some experiments have been carried out to tune certain parameters of the algorithm. In all of them, $Population = 100$, $SurvSize = PopSize/2$, and $MaxIter = 1000$. Moreover, for all the experiments, the data have been simulated.

The experiments have three objectives. The first is to compare the four methods of creating valid chromosomes, the second is to study the genetic algorithm, and the third is to study the execution cost when parallel algorithm is used, all when varying the problem size.

Table 1 shows both the averaged execution time (in seconds) and the averaged percentage of valid chromosomes associated with the four aforementioned methods. Additionally, the corresponding standard deviations are shown as subscripts. Method 1 works very poorly for all the six instances. Method 2 performs better than Method 1. However, the averaged percentage of valid chromosomes decreases as the problem size increases, reaching a value of zero for the biggest numerical example. The performance of Method 4 is also worse than that corresponding to the second method, even with respect to the execution cost. On the other hand, Method 3 is clearly the best approach for determining valid chromosomes, although the size of the problem adversely affects its performance. Specifically, Figure 2 illustrates the superiority of the third method compared to the other three possibilities.

TABLE 2: Execution cost, percentage of valid chromosomes, and solution value of the genetic algorithm (using Method 3), all when varying the problem size.

m	n	s	Time	% val.	Sol.
2	15	1	0.207 _{0.302}	70.00 _{41.68}	0.840 _{0.100}
3	25	2	0.297 _{56.900}	56.90 _{40.99}	0.806 _{0.101}
4	30	2	0.173 _{0.174}	84.22 _{24.79}	0.731 _{0.100}
5	40	3	0.773 _{0.536}	46.52 _{41.26}	0.708 _{0.120}
6	60	4	2.109 _{1.138}	30.84 _{41.47}	0.721 _{0.062}
10	100	10	8.516 _{3.325}	30.51 _{38.67}	0.660 _{0.077}

As for the genetic algorithm, Table 2 shows the averaged execution cost, the averaged percentage of valid chromosomes, and the averaged best solution (the objective function in (3)) when the genetic algorithm is used together with Method 3. The utilization of Method 3 is justified by its superiority compared to Methods 1, 2, and 4. Regarding the percentage of valid chromosomes, the algorithm generates about 84% in the best case ($m = 4$, $n = 30$, and $s = 2$) and about 30% in the worst case (for the two more complex instances). Although there is not a clear trend with respect to the size of the analyzed problems, the results show that the averaged percentage of valid chromosomes is smaller for $m = 6$, $n = 60$, and $s = 4$ and $m = 10$, $n = 100$, and $s = 10$. Moreover, though the last instance ($m = 10$, $n = 100$, and $s = 10$) is considerably more complex than the remaining simulations, the genetic algorithm does not work worse than in the preceding numerical example.

Table 3 shows the execution time and speedup for the parallel version of genetic algorithm. In general, the speedup values are satisfactory being better when the problem size increases. In small problems, the speedup when using 16 processors is lower than that obtained when using 8 or less. Therefore the selection of the number of processors is essential for resource optimization.

6. Conclusions and Future Works

Determining benchmarking information through closest efficient targets is one of the relevant topics in the recent Data Envelopment Analysis literature. However, from a computational point of view, it has been solved by unsatisfactory approaches since all of the existing methods, as was argued, are related to a combinatorial NP-hard problem.

TABLE 3: Execution time and speedup of parallel version of the genetic algorithm when varying the problem size. The average and standard deviation of the values of DMUs are shown. In each case, p is the number of processors used, time is the execution time in seconds, and sp is the speedup.

m	n	s	p	Time	sp
2	15	1	1	0.21 _{0.30}	—
			2	0.14 _{0.19}	1.24 _{1.24}
			4	0.09 _{0.12}	1.85 _{0.59}
			8	0.08 _{0.10}	2.03 _{1.23}
			16	0.12 _{0.17}	1.53 _{0.50}
3	25	2	1	0.30 _{0.27}	—
			2	0.18 _{0.15}	1.55 _{0.22}
			4	0.11 _{0.09}	2.46 _{0.67}
			8	0.08 _{0.07}	3.22 _{1.37}
			16	0.10 _{0.09}	2.59 _{0.86}
4	30	2	1	0.17 _{0.17}	—
			2	0.10 _{0.10}	1.65 _{0.19}
			4	0.06 _{0.05}	2.59 _{0.46}
			8	0.05 _{0.05}	3.17 _{0.85}
			16	0.06 _{0.05}	2.62 _{0.87}
5	40	4	1	0.81 _{0.55}	—
			2	0.46 _{0.30}	1.72 _{0.21}
			4	0.25 _{0.17}	3.04 _{0.44}
			8	0.19 _{0.12}	4.19 _{0.93}
			16	0.17 _{0.12}	4.71 _{1.44}
6	60	4	1	2.11 _{1.14}	—
			2	1.13 _{0.60}	1.85 _{0.10}
			4	0.61 _{0.32}	3.36 _{0.33}
			8	0.38 _{0.20}	5.38 _{0.96}
			16	0.29 _{0.16}	7.11 _{1.84}
10	100	10	1	0.66 _{0.08}	—
			2	4.37 _{1.68}	1.94 _{0.13}
			4	2.32 _{0.83}	3.59 _{0.37}
			8	1.31 _{0.46}	6.33 _{0.70}
			16	0.80 _{0.27}	10.27 _{1.55}

In this paper, for the first time, the determination of closest targets has been approached by means of genetic algorithms and parallel programming. Specifically, a first step has been carried out to solve the problem obtaining valid solutions for the mathematical programming model proposed by Aparicio et al. [3]. At this respect, four different methods to generate the initial population of chromosomes were used and tested by means of some simulated experiments. The third method, based on three algorithms to yield the suitable parameters of the problem, clearly presented the best performance. In a subsequent phase, this third method was used in the generation of valid chromosomes in the genetic algorithm. Finally, the execution time and speedup for the parallel version of the genetic algorithm were shown, demonstrating in particular that the speedup was better when the problem size increased.

In this paper, the introduced approach only takes into account eight constraints on a total of fourteen in (3).

Although the complexity of the problem justifies this first step, the development of a method considering the remaining restrictions may be a good avenue for further follow-up research. In particular, improving the method to generate the initial population of chromosomes and testing different heuristics could help to achieve this specific objective. A deeper study is also in mind of the relation between the different initial parameters and the size of the problem with the computing time required and the effectiveness of the algorithm. Finally, it is worth mentioning that (3) was associated with the Enhanced Russell Graph measure [27]. However, there are a lot of measures in DEA that can be used for measuring technical efficiency through closest targets. Therefore, programming the approach based on the genetic algorithm to solve all of them can be seen as a suitable future work.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors gratefully acknowledge the computer resources, technical expertise, and assistance provided by the Parallel Computing Group of the Murcia University. Additionally, Juan Aparicio and Jose Lopez-Espin are grateful to the Generalitat Valenciana for supporting this research with Grant no. GV/2013/112.

References

- [1] T. Coelli, D. S. P. Rao, and G. E. Battese, *An Introduction to Efficiency and Productivity Analysis*, Kluwer Academic, 1998.
- [2] W. W. Cooper, L. M. Seiford, and K. Tone, *Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver software*, Kluwer Academic, Boston, Mass, USA, 2000.
- [3] J. Aparicio, J. L. Ruiz, and I. Sirvent, "Closest targets and minimum distance to the Pareto-efficient frontier in DEA," *Journal of Productivity Analysis*, vol. 28, pp. 209–218, 2007.
- [4] M. C. A. S. Portela, P. C. Borges, and E. Thanassoulis, "Finding closest targets in non-oriented DEA models: the case of convex and non-convex technologies," *Journal of Productivity Analysis*, vol. 19, pp. 251–269, 2003.
- [5] A. Amirteimoori and S. Kordrostami, "A Euclidean distance-based measure of efficiency in data envelopment analysis," *Optimization*, vol. 59, no. 7, pp. 985–996, 2010.
- [6] J. Aparicio and J. T. Pastor, "On how to properly calculate the Euclidean distance-based measure in DEA," *Optimization*, 2012.
- [7] J. T. Pastor and J. Aparicio, "The relevance of DEA benchmarking information and the least-distance measure: comment," *Mathematical and Computer Modelling*, vol. 52, no. 1-2, pp. 397–399, 2010.
- [8] F. X. Frei and P. T. Harker, "Projections onto efficient frontiers: theoretical and computational extensions to DEA," *Journal of Productivity Analysis*, vol. 11, pp. 275–300, 1999.
- [9] E. Gonzalez and A. Alvarez, "From efficiency measurement to efficiency improvement: the choice of a relevant benchmark,"

- European Journal of Operational Research*, vol. 133, pp. 512–520, 2001.
- [10] C. Baek and J.-d. Lee, “The relevance of DEA benchmarking information and the least-distance measure,” *Mathematical and Computer Modelling*, vol. 49, no. 1-2, pp. 265–275, 2009.
 - [11] G. R. Jahanshahloo, J. Vakili, and S. M. Mirdehghan, “Using the minimum distance of DMUs from the frontier of the PPS for evaluating group performance of DMUs in DEA,” *Asia-Pacific Journal of Operational Research*, vol. 29, no. 2, Article ID 1250010, 25 pages, 2012.
 - [12] W. Briec, “Hölder distance functions and measurement of technical efficiency,” *Journal of Productivity Analysis*, vol. 11, pp. 111–131, 1998.
 - [13] W. Briec and J. B. Lesourd, “Metric distance function and profit: some duality results,” *Journal of Optimization Theory and Applications*, vol. 101, no. 1, pp. 15–33, 1999.
 - [14] W. Briec and B. Lemaire, “Technical efficiency and distance to a reverse convex set,” *European Journal of Operational Research*, vol. 114, pp. 178–187, 1999.
 - [15] T. Coelli, “A multi-stage methodology for the solution of orientated DEA models,” *Operations Research Letters*, vol. 23, no. 3–5, pp. 143–149, 1998.
 - [16] L. Cherchye and T. Van Puyenbroeck, “A comment on multi-stage DEA methodology,” *Operations Research Letters*, vol. 28, no. 2, pp. 93–98, 2001.
 - [17] S. Lozano and G. Villa, “Determining a sequence of targets in DEA,” *Journal of Operational Research Society*, vol. 56, pp. 1439–1447, 2005.
 - [18] A. Charnes, J. J. Rousseau, and J. H. Semple, “Sensitivity and stability of efficiency classifications in data envelopment analysis,” *Journal of Productivity Analysis*, vol. 7, pp. 5–18, 1996.
 - [19] A. Takeda and H. Nishino, “On measuring the inefficiency with the inner-product norm in data envelopment analysis,” *European Journal of Operational Research*, vol. 133, no. 2, pp. 377–393, 2001.
 - [20] G. R. Jahanshahloo, F. Hosseinzadeh Lotfi, and M. Zohrehbandian, “Finding the piecewise linear frontier production function in data envelopment analysis,” *Applied Mathematics and Computation*, vol. 163, no. 1, pp. 483–488, 2005.
 - [21] G. R. Jahanshahloo, F. Hosseinzadeh Lotfi, H. Zhiani Rezai, and F. Rezai Balf, “Finding strong defining hyperplanes of production possibility set,” *European Journal of Operational Research*, vol. 177, no. 1, pp. 42–54, 2007.
 - [22] G. R. Jahanshahloo, J. Vakili, and M. Zarepisheh, “A linear bilevel programming problem for obtaining the closest targets and minimum distance of a unit from the strong efficient frontier,” *Asia-Pacific Journal of Operational Research*, vol. 29, no. 2, Article ID 1250011, 19 pages, 2012.
 - [23] W. D. Cook and L. M. Seiford, “Data envelopment analysis (DEA)—thirty years on,” *European Journal of Operational Research*, vol. 192, no. 1, pp. 1–17, 2009.
 - [24] M. J. Farrell, “The measurement of productive efficiency,” *Journal of the Royal Statistical Society A*, vol. 120, pp. 253–281, 1957.
 - [25] A. Charnes, W. W. Cooper, and E. Rhodes, “Measuring the efficiency of decision making units,” *European Journal of Operational Research*, vol. 2, no. 6, pp. 429–444, 1978.
 - [26] R. D. Banker, A. Charnes, and W. W. Cooper, “Some models for estimating technical and scale inefficiencies in data envelopment analysis,” *Management Science*, vol. 30, pp. 1078–1092, 1984.
 - [27] J. T. Pastor, J. L. Ruiz, and I. Sirvent, “An enhanced DEA russell graph efficiency measure,” *European Journal of Operational Research*, vol. 115, pp. 596–607, 1999.
 - [28] K. Tone, “A slacks-based measure of efficiency in data envelopment analysis,” *European Journal of Operational Research*, vol. 130, no. 3, pp. 498–509, 2001.
 - [29] M. Mitchell, *An Introduction to Genetic Algorithm*, MIT Press, 1998.
 - [30] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, “Hyperheuristics: an emerging direction in modern search technology,” *KHandbook of Metaheuristics*, vol. 16, pp. 457–474, 2003.
 - [31] R. Chandra, R. Menon, and L. Dagum, *Parallel Programming in OpenMP*, Operations Morgan Kaufman, 2001.
 - [32] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, “An extended set of FORTRAN basic linear algebra subroutines,” *ACM Transactions on Mathematical Software*, vol. 14, pp. 1–17, 1988.
 - [33] E. Anderson, Z. Bai, C. Bischof, J. Demmel, and J. J. Dongarra, *LAPACK User's Guide*, Society for Industrial and Applied Mathematics, 1995.
 - [34] R. Clinton Whaley, A. Petitet, and J. Dongarra, “Automated empirical optimizations of software and the ATLAS project,” *Parallel Computing*, vol. 27, pp. 3–35, 2001.